Hierarchical Anomaly-Based Detection of Distributed DNS Attacks on Enterprise Networks

Minzhao Lyu, Hassan Habibi Gharakheili, Craig Russell, and Vijay Sivaraman

***Abstract*—Domain Name System (DNS) is a critical service for enterprise operations, and is often made openly accessible across firewalls. Malicious actors use this fact to attack organizational DNS servers, or use them as reflectors to attack other victims. Further, attackers can operate with little resources, can hide behind open recursive resolvers, and can amplify their attack volume manifold. The rising frequency and effectiveness of DNS-based DDoS attacks make this a growing concern for organizations. Solutions available today, such as firewalls and intrusion detection systems, use combinations of black-lists of malicious sources and thresholds on DNS traffic volumes to detect and defend against volumetric attacks, which are not robust to attack sources that morph their identity or adapt their rates to evade detection.**

We propose a method for detecting distributed DNS attacks that uses a hierarchical graph structure to track DNS traffic at three levels of host, subnet, and autonomous system (AS), combined with machine learning that identifies anomalous behaviors at various levels of the hierarchy. Our method can detect distributed attacks even with low rates and stealthy patterns. Our contributions are three-fold: (1) We analyze real DNS traffic over a week (nearly 400M packets) from the edges of two large enterprise networks to highlight various types of incoming DNS queries and the behavior of malicious entities generating query scans and floods; (2) We develop a hierarchical graph structure to monitor DNS activity, identify key attributes, and train/tune/evaluate anomaly detection models for various levels of the hierarchy, yielding more than 99% accuracy at each level; and (3) We apply our scheme to a month's worth of DNS data from the two enterprises and compare the results against blacklists and firewall logs to demonstrate its ability in detecting distributed attacks that might be missed by legacy methods while maintaining a decent real-time performance.

***Index Terms*—Distributed attack, DNS, network security, anomaly detection.**

I. Introduction

The critical role of DNS infrastructure in large enterprises makes it a popular target for cyber-criminals. In recent years, distributed denial-of-service (DDoS) attacks based on DNS have risen in frequency, volume, and sophistication [1], and it is likely to worsen further as the attack surface expands with bring-your-own devices (BYOD) and Internet-of-Things (IoT) appliances [40], [34]. As an example, more than 100K compromised IoT devices were enslaved in 2016 for a global-scale DDoS attack on Dyn's DNS infrastructure [21] which prevented Internet users from accessing more than 1.2K web services such as Netflix, Spotify, and Twitter. According to EfficientIP [16], in 2020, nearly 79% enterprises suffered from DNS attacks which cost on average $924K per attack.

In spite of these growing risks, organizations today have huge DNS "blind spots" [10] that leave them exposed to DNS-based attacks. Large enterprises have many departments, each with their own information technology (IT) personnel, independently managing DNS servers/caches (this is particularly true in loosely-federated organizations like Universities), so it becomes very challenging to track and lock-down internal DNS infrastructure at a central firewall. Small businesses often rely on their Internet Service Provider (ISP) for security, and the ISP is quite likely to allow all DNS traffic through as they may not have visibility of the DNS infrastructure of the business, which can be dynamic. Further, even for organizations that do try to restrict incoming DNS traffic, malicious entities can spoof well-crafted DNS queries to reach and exploit their internal DNS services. New methods are therefore needed that are robust to dynamic DNS infrastructures and morphing attacks.

DNS-based attacks are broadly categorized into three groups: (1) query floods (also known as DNS flooding), mainly sourced from botnets that directly bombard victim servers (mainly authoritative name servers or recursive resolvers) with a large number of queries to exhaust the victim's resources, (2) DNS reflection/amplification attacks that utilize open DNS resolvers as proxies [12] by sending spoofed source address queries (using the intended victim's IP address), and (3) DNS scans [7], [23] that actively probe a target network to identify potential victims for future DNS floods and/or reflections. As emphasized earlier, current intrusion detection/prevention and firewall systems rely on static configurations, and are not robust enough to detect and block these attacks in the presence of dynamic DNS infrastructures and morphing attacks.

Most security solutions, both software-based tools and hardware-based appliances, typically use static *signatures* of known attacks. However, signature-based detection approaches are difficult to scale cost-effectively and require regular updates since attack vectors evolve rapidly [26]. Existing commercial intrusion detection systems often use threshold methods [11], [19], [42], [48] whereby they search for recurring patterns in traffic by counting the number of certain events occurring within a "defined period", and take action if the configured "threshold values" are exceeded.

Existing methods require IT departments to configure static policies and set threshold values for counting periods. Determining suitable thresholds is a non-trivial task since it is

M. Lyu is with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia, and CSIRO's Data61, Sydney, NSW 2015, Australia (e-mail: minzhao.lyu@unsw.edu.au).

H. Habibi Gharakheili, C. Russell and V. Sivaraman are with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia (e-mails: h.habibi@unsw.edu.au, craig.russell@unsw.edu.au, vijay@unsw.edu.au).

not obvious what the optimal values for effective defensive policies should be applied – high thresholds allow attacks to go undetected while small thresholds can result in a large number of false positives, incurring a high cost of investigation. If thresholds are specifically configured by administrators (*e.g.,* 120% of empirical peak load for a critical server as suggested by firewall vendors [42]), then existing firewalls can protect specific servers (or IP subnets) from being overwhelmed by an excessive rate of inbound packets affecting their normal service operation. However, they are still unable to identify and defend against distributed attacks (*i.e.,* identifying attack sources), since each source generates malicious traffic at low rates [15] (and may appear legitimate). We note that low-rate attacks (*e.g.,* CPU-exhaustion DoS attacks [37]) are still important to detect since they may be powerful enough to adversely affect servers with less resources. Also, distributed attacks often begin at a low rate before causing serious disruption or damage [49]. They often originate from bot devices under a subnet or autonomous system (AS) [52] that can only be detected by maintaining information for external entities at multiple levels of aggregation.

To address these shortcomings, we develop, implement, and evaluate an anomaly-based detection system, incorporating a dynamic and hierarchical graph structure of well-selected attributes, to capture real-time volumetric behavior of external hosts and detect external anomalous entities at various levels of aggregation. There exist prior works which detect attacks at destination networks, but their primary focus is to identify "victims". To the best of our knowledge, we are the first to propose a victim-side method for isolating " attack sources" in distributed DNS attacks (*i.e.,* DDoS floods and reconnaissance scans) depending on their nature of distribution across hosts, subnets, or AS.

Our key contributions are summarized as follows. **Firstly**, in §III, we highlight the characteristics of malicious Internet hosts that launch DNS volumteric scans and flooding attacks on enterprise networks by analyzing datasets of real DNS traffic, consisting of approximately 400 million DNS queries and responses, collected from two enterprises over a week. **Secondly**, in §IV, we develop a hierarchical graph structure with dynamic nodes and edges for monitoring the DNS query behavior of external entities at various levels of aggregation (namely host-, subnet-, and AS-level), identify attributes that can be computed cost-effectively in real-time, generate anomaly detection models using benign traffic only, and evaluate them using benign traffic as well as synthetically generated attack traffic. **Finally**, in §V, we demonstrate the efficacy of our scheme especially in detecting low-rate DNS-based attacks by replaying a month's worth of DNS traffic to our prototype, and validate our results by checking flagged external hosts against public blacklists of malicious IP addresses, as well as against logs from a commercial firewall.

II. Related Work

In this section, we survey related literature for the scopes our work falls in (*i.e.,* DNS security and defense of distributed network attacks) and highlight the research gaps that are addressed in this paper.

A. DNS Security

DNS security has been an attractive topic for both industry and academia, especially on integrity of DNS records [45], [27], [28], and vulnerabilities of DNS infrastructure to volumetric attacks [33], [44], [24].

As for the integrity of domain names, malicious users on the Internet exploit DNS protocol to signal and control malicious network infrastructures such as DDoS botnet. Authors of [56] analyzed DNS responses to detect unusual behaviors (anomaly detection) related to domain names such as typo squatter domains and fast flux domains. *Kopis* [2] can accurately detect malware-related domains by using statistical features (such as distribution of requesters) at top-level domain servers. In [3], authors point out that attackers use domain generation algorithm to bypass detection systems, and they come up with a detection approach using clustering and classification algorithms for domain names and their requesters. Furthermore, as discussed in [8], malicious entities are able to launch resource-exhaustion attacks on DNS infrastructures by using disposable domains. Integrity problems can also arise during DNS lookups from legitimate users. Unsecured DNS communication can be easily hijacked and manipulated by third parties [27]. To address this problem, secured extensions such as DNSSEC [4] and DNS-over-HTTPS [22] have been proposed. However, Chung et al. revealed that the adoption of DNSSEC is still in early stages [9].

Researchers have also investigated vulnerabilities of DNS services and infrastructure to volumetric attacks including DDoS and reconnaissance scans. DNSSEC is seen more attractive by DNS amplification attackers. Rijswijk-Deij et al. [51] showed that DNSSEC can be mis-used for larger amplifications (in reflection attacks) compared to standard DNS. Work in [24] actively probed DNS resolvers available on the Internet and quantified their reflective capabilities.

In this work, we develop data-driven models, trained by real enterprise data, to detect and identify external anomalous sources (at three hierarchical levels including host, subnet, and AS) that attempt to discover, attack, or exploit enterprise DNS infrastructure, even when they reduce their activity profile by getting distributed and reducing their traffic rates.

B. Defense of Distributed Network Attacks

Efficient defense of distributed network attacks have been widely-studied by many researchers. Works can be classified based on where they are deployed regarding the anatomy of attacks, *i.e.,* at-source, at-destination, or at-network [55]. At-source methods are deployed at the place where attacks originated, *D-WARD* [38] looks for suspicious traffic patterns (*e.g.,* source IP spoofing on incoming traffic) and applies rate-limiting to corresponding hosts at the network edge. *ShadowNet* [6] measure traffic attributes (*e.g.,* rate of HTTP GET requests) from edge routers serving IoT devices to detect IoT botnets. For at-network solutions, *LADS* [46] uses lightweight SNMP and NetFlow statistics collected from an ISP's backbone routers, and *Bohatei* [18] uses SDN to reroute suspicious traffic (*e.g.,* excessive number of TCP SYN packets) to IDS middleboxes on ISP networks. *PSI* [54] is an example

Table I: Summary of dataset during 1-7 May 2018.

	Query In	Response Out	Unanswered Qry.	Invalid Qry.	`NameError`	`Serv.Failure`	`Refused`
University Campus	139,136,237	97,008,082	42,127,345	5,335,019	4,935,385	264,100	135,404
Research Institute	99,847,262	61,164,416	35,332,068	9,335,751	4,154,238	490,045	4,640,829

of at-destination approach. It is deployed at the edge of an enterprise network which dynamically applies security rules of firewalls or IDS to traffic of interest (*e.g.,* certain protocols) for detecting attacks towards enterprise hosts with optimized accuracy and cost. With emerging SDNs, denial-of-service attacks on controllers start ramping up. *SWGuard* [58] defenses against control-plane reflection attacks by monitoring victim's down-link control messages.

Note that both at-source and at-network systems are effective in both detection and mitigation but they require automatic coordination (signaling) between network operators on the path between source and destination, while legacy at-destination methods can well isolate victims but not source of attacks. Our work is the first to develop a novel hierarchical graph structure for detecting attacks at-destination that can precisely identify external attack sources at IP-level to subnet and AS levels, even the sophisticated distributed attackers which keep their individual attack traffic rates so low, bypassing network firewalls. Comparing our work with systems against volumetric distributed attacks in particular, existing methods primarily aim to detect potential victim servers (instead of identifying attack sources) [54], [57], [25], [31]. Moreover, their objective upon detection of an attack is to either isolate or rate-limit the victim, affecting malicious and benign traffic sources alike. Our work, instead, aims to detect distributed attacks and identify their sources which can be precisely blocked (automatically or manually) for remedial action without affecting benign/legitimate sources.

C. Summary of Research Gaps

we have identified three research gaps in relevant existing works. First, no prior work systematically characterized the behavior of malicious entities that generate distributed DNS attacks on enterprise networks. In §III, we highlight the anatomy of distributed DNS attacks on enterprise assets. Second, no prior solution considered the context of attack sources in terms of their subnets and ASes. We, instead, develop a hierarchical data structure (§IV) to track the behavior of attackers at various aggregate levels. Third, existing enterprise-side defense systems predominately aim to detect victim internal servers which are under attack (*i.e.,* identifying "destination" of attacks), and hence do not distinguish between malicious and benign external sources (*i.e.,* unable to identify "source" of attacks, especially when they are distributed). We, instead, detect distributed attack sources (§V) even those which keep their traffic rates relatively low to bypass security appliances.

III. Profile of DNS Volumetric Queries to Enterprise Networks

In this section, we analyze the DNS traffic collected from the border of two enterprise networks, a large University campus and a medium-size research institute to profile incoming queries to enterprise hosts. Our data analysis primarily focuses on the behavior of external source entities (*i.e.,* hosts, subnets, and ASes outside the protected enterprise network) that may contact or attack DNS servers in any enterprise settings. Note that the two studied networks have rich DNS facilities (*i.e.,* authoritative DNS servers, recursive resolvers and public-facing servers with assigned domain names) which are frequently targeted by distributed DNS attacks. Smaller organizations with fewer DNS servers can become target of similar inbound attacks, perhaps at relatively lower frequency.

In both instances the IT department of the enterprise provisioned a full mirror (both inbound and outbound) of their Internet traffic (each on a 10 Gbps interface) to our data collection system from their border routers (**outside** of the firewall), and we obtained appropriate ethics clearances[1] for this study. We extracted all DNS packets from each of the enterprise Internet traffic streams in real-time by configuring OpenFlow match rules for incoming/outgoing IPv4 port 53 packets on an SDN switch. Our dataset was collected during the full month May 2018, though our analysis in this section focuses on a one week period (1-7 May 2018), consisting of 139.1M and 99.8M incoming DNS queries to, along with 97M and 61.2M outgoing DNS responses from, hosts of the university and research networks respectively – a brief summary of the dataset for the first week of May 2018 is shown in Table I.

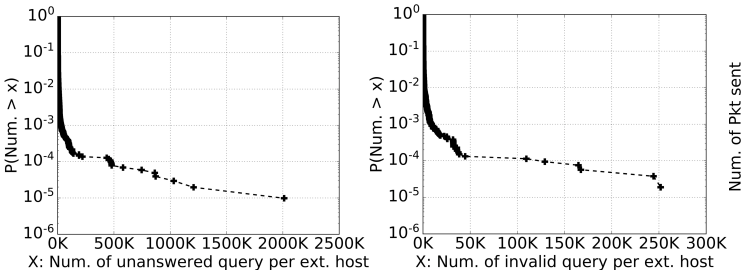
A. Incoming DNS Queries

A benign incoming DNS query to an enterprise network typically targets an authoritative name server and the corresponding server responds with a `NoError` flag. There are, however, other types of incoming queries observed in real networks: **unanswered queries** (*i.e.,* DNS queries with no response) and **invalid queries**, those answered with flags other than `NoError`, for various reasons such as the query packet is malformed or corrupted, the query name does not exist or is not relevant for the organization (and thus does not resolve to any IP address), or an unintended recursive resolver is queried.

Unanswered Queries: During the first week of May 2018, we found a total of 42M and 35M unanswered incoming queries to the networks of the university and the research institute respectively – approximately one third of the total incoming queries in both organizations. We have verified that these queries typically targeted non-DNS servers inside the organization (by performing reverse DNS lookups) or IP addresses that are not active/present in the network to receive the query, and therefore were not answered.

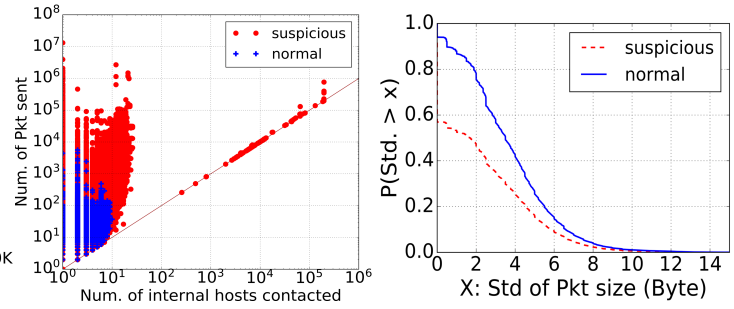
Invalid Queries: We also found a total of 5.3M and 9.3M invalid incoming queries to the networks of the university and the research institute respectively – accounting for 4%

[1]UNSW Human Research Ethics Advisory Panel approval number HC17499, and CSIRO Data61 Ethics approval number 115/17.



(a) Unanswered queries. (b) Invalid queries.

Figure 1: CCDF of: (a) unanswered, and (b) invalid, query counts per each external host for a duration of one week (to university network).



(a) Pkt count vs queried hosts count. (b) Variance of query packet size.

Figure 2: Query behavior of external hosts during a week: (a) number of packets sent versus number of internal hosts contacted per individual external host, and (b) CCDF of query packet size (Bytes) per external host – normal and suspicious external hosts are highlighted in blue and red respectively.

and 9% of the total incoming queries. For the university network, 92.5% of invalid queries were answered with a `NameError` flag, indicating that the DNS server was not able to resolve the queried name. Also, 5.3% of responses had an error code of `ServerFailure` and 2.5% an error code of `Refused`. Additionally, a tiny fraction of responses had `NotImplemented` and `FormatError` flags set (118 and 12 invalid queries respectively) – these error codes indicate incorrect messages contained in the queries or incorrect destination name servers (*e.g.,* a wrong domain name for the authoritative name server of the organization).

For the research institute, the distribution of error codes was as follows: 44.4% with `NameError`, 5.2% with `ServerFailure`, 49.6% with `Refused`, 0.4% with `FormatError` and only 117 instances of `NotImplemented` with additional codes including `NotAuth` and `NXRRSet` seen in 31,540 and 24 of responses to invalid queries. The last two error codes correspond to wrong destinations or messages in the queries.

B. External Hosts Sending Unwanted DNS Queries

Considering all external hosts sending DNS queries to the university network[2], we counted a total of 168,538 unique hosts (*i.e.,* IP addresses) in our dataset for the first week of May 2018. These external hosts come from 46,729 distinct subnets and 16,775 unique autonomous systems. We found a total of 112,704 external hosts sent some form of unwanted query: 41,893 external hosts sent both unanswered and invalid queries; 59,907 sent only unanswered queries and 10,904 sent only invalid queries.

Interestingly, we observe that only a tiny fraction of external hosts are very active in sending unwanted queries. In Fig. 1, we show CCDF plots of unwanted queries counts per each external host. It can be seen that 29 external hosts each generated more than 100K unanswered queries, as shown in Fig. 1(a), with three hosts each generating more than a million unanswered queries over the week. Similarly, only 6 external hosts each sent more than 100K invalid queries, as shown in Fig. 1(b). We will see later in this section that these heavy hosts are indeed involved in volumetric-based DNS attacks in the form of host scanning and/or query flooding.

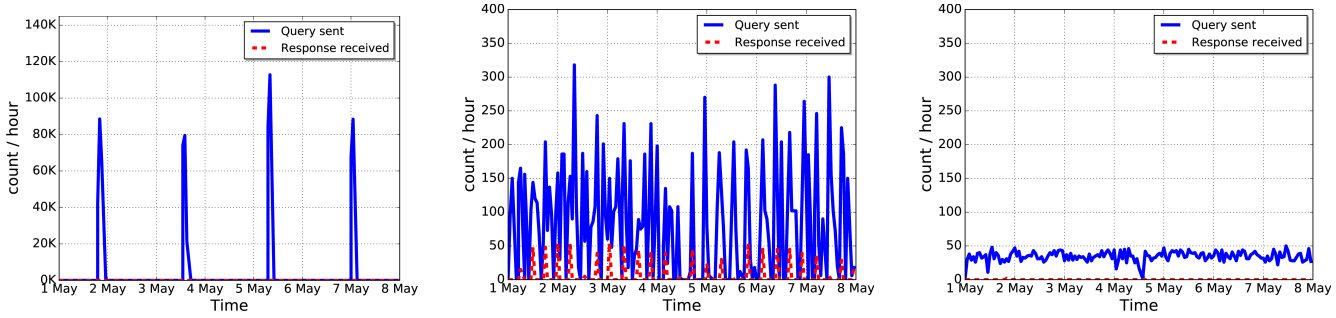
[2]In this subsection, we have omitted analysis of the research institute data so as to concentrate on insights. Similar observations were made for both organizations.

We now divide the external hosts into two groups: hosts with "normal" and hosts with "suspicious" behavior in their DNS queries – normal hosts do not send any unwanted DNS queries to the network. Note that some of the suspicious hosts may just have typographical errors in their DNS queries. Suspicious hosts are distributed across 30,295 subnets of 11,539 autonomous systems. We found that 73 ASes each with more than 100 hosts account for 59% of all suspicious IP addresses, of which the top 5 heavy ASes contain 19% of all suspicious hosts. This is not surprising as approximately half of all cyber-attacks originate from compromised devices in a small number of countries with insecure infrastructure [52]. On the other hand, 55,834 normal hosts are distributed across 10,706 unique ASes – the number of normal hosts for each AS is evenly distributed.

Fig. 2 illustrates the difference between the query behavior of normal and suspicious external hosts. Starting with the scatter plot of query packets count versus number of unique internal hosts queried for each external host, shown in Fig. 2(a), a cluster of normal hosts (shown by blue cross markers) is clearly visible at the bottom left corner of the plot – a normal external host does not send more than 10,000 queries in a week. On the other hand, suspicious hosts (shown by red circle markers) display two clusters (*i.e.,* suspicious hosts lying on the line $y = x$ that send one query to a large number of internal hosts, and suspicious hosts clustered on the left that send a large number of queries to a limited set of internal hosts).

Moving to the variance of the queries, we show in Fig. 2(b) the CCDF plot of the standard deviation of packet size sent by each of the normal and suspicious external hosts. It can be clearly seen that suspicious hosts (as shown by dashed red lines) display a smaller variation in the size of their query packets compared to normal hosts, highlighting a set of repopulated query names were automatically sent by suspicious hosts (*i.e.,* a bot). More importantly, we observe that a large fraction (*i.e.,* 42%) of suspicious hosts had identical packet size (*i.e.,* zero variation) for their queries during the week.

We also analyzed the payload of queries sent by each external host. Among suspicious hosts (excluding 4,436 hosts with just one query), we identified 19,551 hosts each used



(a) Periodic (an external host from Russia). (b) Focused (an external host from China). (c) Low-rate (an external host from US).

Figure 3: Weekly time-trace of various types of DNS scans: (a) periodic, (b) focused, and (c) low-rate; on enterprise hosts from the Internet.

identical query names for all queries during the week, with many of these hosts having a large number of queries sent to the university network (*e.g.,* an external IP address sent 13,144,130 identical query packets to one internal IP address). We also observed that most of these suspicious query names are not relevant to the services provided by the enterprise. For example, one IP address (located in Russia) sent 763K queries with the query name "`com`" to the campus network, and another IP (located in Lithuania) sent 397K queries for "`nrc.gov`" during the week.

It is important to note that using identical query names is also seen for normal hosts. We note that 8,836 normal external hosts each queried only one domain name, but each of these hosts generated only a small number of queries during the week. Another observation for these specific normal hosts is that the DNS ID (*i.e.,* a 16-bit identifier in the DNS header) varies over time, whereas for suspicious hosts only one DNS ID was consistently used for successive DNS queries.

Next, we analyze in more detail the properties of the two types of unwanted DNS queries.

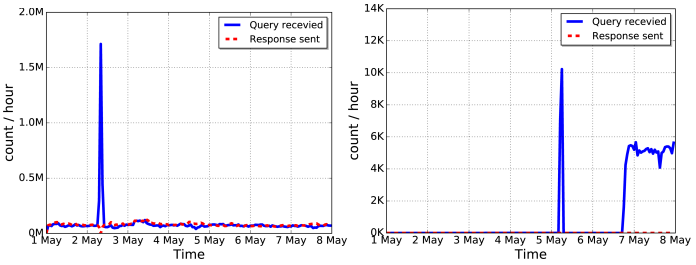
C. DNS Query Scans

The first cluster of suspicious hosts shown in Fig. 2(a) corresponds to scanners that sent one query packet (*i.e.,* probe) to a large number of (potentially all) internal IP addresses. We note that the university owns three "`/16`" IPv4 address prefixes, which represents more than 196K unique IP addresses for internal hosts, as indicated by the largest number of hosts contacted in Fig. 2(a). Scans are typically performed by malicious entities to make a list of available DNS servers inside enterprises that could subsequently be used as reflectors for DDoS attacks [23]. Also, there are a number of "white hat" researchers who conduct DNS query probing to only detect (not attack) vulnerable DNS servers available on the Internet [20], [24], [14]. Scan queries are typically crafted packets with the query name field most likely not relevant for the enterprise network. If this query reaches an operational DNS server, a response may or may not be returned (depending on the particular name server configuration). Scanners can choose various strategies (in terms of the query rate) to perform reconnaissance tasks. Next, we consider three types of scans – a representative of each type is shown in Fig. 3.

1) Periodic Heavy Scans: Some scanners choose to scan the network on a periodic basis. Fig. 3(a) shows a scanner from Russia that conducted 4 large-scale scans during the week by sending queries to a total of 196,865 IP addresses (*i.e.,* almost the full three `/16` prefixes) inside the university campus. Each scan lasted for about 2 hours and the query name "`com`" was repeatedly used in all query packets. Forty one DNS servers (*i.e.,* recursive resolvers and authoritative name servers) inside the organization responded to this scanner – a majority of servers were not able to respond with a `NoError` flag to these queries: 24 servers responded with error code `Refused`, 7 with `ServerFailure`, and 2 with `NameError`. Surprisingly, 8 DNS resolvers (verified by reverse lookups) inside the enterprise network responded with `NoError` flag to the scanner, resolving the queried name "`com`". We note that the answers to this top-level domain name are fairly large, resulting in an attractive amplification factor (*i.e.,* ratio of response size to query size) of up to 43. We will see later in §III-D2 that some of these servers were used as reflectors.

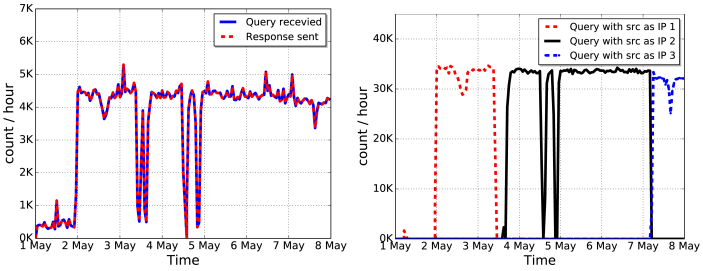
2) Focused Scans: Instead of blindly sweeping the entire IP range of a target network, some scanners (those with prior knowledge) may focus on selected IP addresses in their dictionary. Focused scans aim to validate the availability of potential DNS servers that can be used as reflectors/victims – some vulnerable DNS servers may subsequently be secured by a change of configuration or may no longer be operational.

A sample of focused scans is shown in Fig 3(b). This scanner (located in China) targeted 18 IP addresses from 2 subnets of the university network by periodically looping over a static list of hosts. Each scan round consisted of sending one customized query with the name "`d.c.b.a.in-addr.arpa`" to each IP address "`a.b.c.d`" – this reverse lookup query causes the victim (*i.e.,* potential DNS server) to return its own DNS name inside the organization. We observed that the same loop was repeated 760 times during the week. We verified (by manual reverse lookup) that 2 of these IP addresses are the university's main recursive resolvers, 3 of which are resolvers at Faculty/Department-level, and 13 IP addresses are WiFi gateways inside the enterprise network. Among these 18 internal hosts, only one department-level DNS resolver responded with the "`NoError`" flag, revealing its identity. Other DNS resolvers were securely configured and did not respond to this scanner – enterprise resolvers are meant to



(a) Targetting an internal name server. (b) Targetting an internal web server.

Figure 4: Weekly time-trace of DNS flooding on enterprise hosts from the Internet, targeting an internal (a) name server, and (b) web server.



(a) Behavior of an internal resolver. (b) Aggregate queries from 3 victims to 5 internal DNS servers.

Figure 5: Weekly time-trace of a reflective DNS flooding attack: (a) one attack reflector (an internal DNS resolver), and (b) three external victims.

serve only internal hosts.

3) Slow-Rate and Distributed Scans: As heavy scans (*i.e.,* high rate queries) can be easily detected by current firewalls and intrusion detection systems, sophisticated scanners may choose to go under the radar by lowering their query rate or distributing the probing task across a number of hosts. Fig. 3(c) shows an example of a low-rate scan. The external host (located in US) consistently sent approximately 40 queries per hour with the question name "`VERSION.BIND`" – in total 5831 queries were sent to 5806 IP addresses during the week (*i.e.,* on average one query per internal host). We note that this query (specific to the most widely deployed DNS server application BIND) asks the server's version. We found that only one host (*i.e.,* a department-level DNS resolver) replied to this query with error code `Refused`. Interestingly, right after this response, the scanner sent another query with question name "`direct.shodan.io`" and the DNS resolver successfully responded with `NoError`.

Impatient low-rate scanners tend to distribute their task across multiple sources (more likely from a subnet/AS under their control). In our dataset from the university network, we found (via manual analysis) 6 distributed scans each originating from a distinct subnet. For each of these scans we observed a similar scan pattern (time-series, total count, query name) across all hosts involved. For example, in one of these distributed scans, 16 external hosts from a /22 prefix (located in US) each sent about 75K queries (one query per internal host contacted) asking for "`dnsscan.shadowserver.org`".

We also found a distributed scan sourced from multiple prefixes within a distinct AS – manually finding this type of distributed attack is a non-trivial task. In this scan, only three external hosts each from a different subnet (*i.e.,* one in /17, one in /12, and one in another /12 subnet) all associated with AS4134 (located in China), each generated about 130K queries to 65K internal hosts (*i.e.,* two queries per internal host) asking two domain names "`www.163.com`" and "`version.bind`".

D. DNS Query Floods

We now consider the second cluster of hosts generating unwanted queries, shown in Fig. 2(a). These external hosts flood (sending a large number of queries to) a small number of enterprise hosts. We note that some flooders aim to exhaust the resources of the enterprise host (primarily DNS servers) [13], [34] whereas others aim to use enterprise servers to reflect/amplify volumetric DNS traffic to third party victims [35]. In the latter scenario, the attacker spoofs the source IP address of the query by using the intended victim's address.

Similar to scanners (described in §III-C), flooders may use one or a list of query names which may or may not be relevant to the enterprise network. Due to the objective of flooders, we expect to see a higher rate of queries coming from external hosts to the enterprise network. Next, we analyze two types of DNS query floods with supporting examples from the two enterprise networks.

1) Flooding Enterprise Servers: DNS flooders may target a DNS server of an enterprise to exhaust its computational resources (by asking it to resolve an excessive number of queries) or a non-DNS server to consume its network resources. Fig. 4(a) shows the DNS traffic pattern (incoming queries and outgoing responses) for one of the main authoritative name servers inside the research institute. We can see that this server typically handled approximately 50K query packets per hour for domain names associated with the research organization – number of queries and responses are almost the same during normal operation (*i.e.,* except for the spike period of an attack on 2 May). However, between 06:48AM to 08:11AM on 2 May, this server received a surge of queries (*i.e.,* about 1.7M queries per hour) sourced from 29,614 external hosts, 4,053 of which kept sending repeated queries with the research organization domain name (the characters were randomly in capital or lower case, *e.g.,* "`reSeaRChInstituTe.OrG`" and "`ResEaRchINSTituTe.oRG`"). The highly suspicious external hosts were associated with 432 ASes (122 in U.S., 40 in Australia, 17 in Canada, and 17 in Brazil), where 26 ASes cover 3183 (78.5%) flooders. We note that the top two ASes (both in US) account for 1,211 and 510 flooders.

We can see that this large scale attack resulted in no query (legitimate or malicious) being responded to, as shown by red lines hitting zero during the attack period in Fig. 4(a). We note that this might be because either the server became non-operational, or the enterprise border firewall had detected the attack and possibly dropped all queries to protect the server.

Fig. 4(b) shows a sample time-trace of DNS traffic for the second type of victims (*i.e.,* a non-DNS host). The victim host

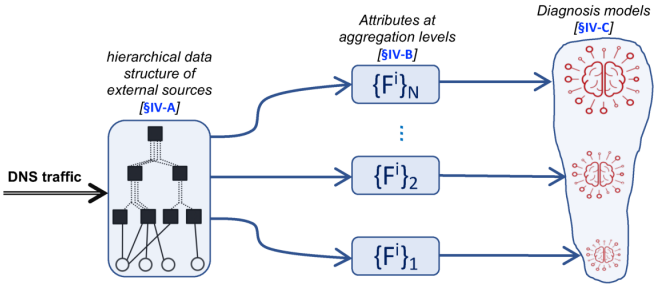


Figure 6: Architecture schema of our methodology.

is a department-level web server in the university network. We can see that this web server does not typically receive DNS traffic but two instances of query floods hit this server: spikes on 5 May and 6-7 May. During 04:33 to 05:51 on 5th May, 18 external IPs each generated about 800 queries with the question name of either "`qjnntx.eleximg.com`" or "`static.mobike.com`". These suspicious external hosts are associated with 4 ASes, of which 2 heavy ASes (AS20473 and AS36351 located in US) account for 10 and 6 attackers – it is a common practice for attackers to develop their botnets within a compromised subnet or AS [52].

For the second instance, between 19:03 on 6th May till the end of the week, 11 flooders from 2 ASes (7 IP addresses from AS36351 in the US and 4 IP addresses from AS132203 in China) each sent about 15K queries to the web server with irrelevant query names from the list of "`qjnntx.eleximg.com`", "`c.afekv.com`", and "`global-ldns.v3.apsv1.com`".

2) Reflective DNS Floods: As mentioned earlier, internal DNS servers of enterprises are targets for cyber-attackers for reflecting volumetric DNS traffic to third party victims on the Internet. Enterprise DNS servers are often discovered prior to this type of attack (as explained in §III-C). We note that the source IP address of queries in reflective DNS floods are spoofed using the intended victim's IP address, therefore external victims are perceived as external flooders by the border device of enterprise network.

We found one example of such an attack in our dataset for the university network that was well coordinated and persisted for almost a week by involving five internal DNS resolvers reflecting to three victims on the Internet, as shown in Fig. 5. 5 DNS resolvers (those that successfully responded to periodic scans) inside the university network simultaneously received a surge of DNS query packets (*i.e.,* around 4K-5K per hour) at around 11pm on 1st May with the question name "`ietf.org`". We note that the response size (in bytes) varies in the range of 15 to 45 times (*i.e.,* the amplification factor) the query size. Fig. 5(a) shows the query count (and the corresponding response count) for one of these DNS resolvers – others displayed almost the same pattern with a slight variation in their traffic rate.

Considering aggregated query traffic (with the victims' addresses as source) in Fig. 5(b), it is evident that the three victims were targeted consecutively (each shown by unique line color). We note that all of three victims are servers associated with AS49453 located in The Netherlands.

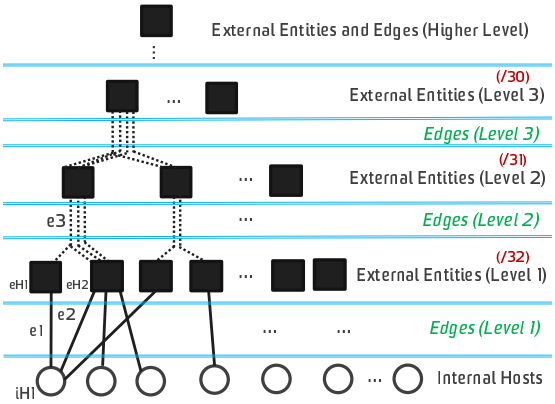


Figure 7: Our hierarchical data structure.

IV. Volumetric Profiling and Detection Scheme

In this section, we present our methodology in profiling and detecting distributed DNS attacks by developing a dynamic volumetric behavior model, and employing anomaly detection algorithms. Fig. 6 illustrates the schema of our method. We first develop (§IV-A) a binary-hierarchical attributed graph data structure to describe the volumetric traffic profile of external entities and mathematically show its efficacy in detecting attacks (especially distributed ones) by simple thresholding at multiple levels of the hierarchy. Our data structure is applicable to generic volumetric scans and attacks. In this paper, we only demonstrate its merits specifically to DNS-based attacks. We prove by mathematical analysis that our scheme is able to detect scans and floods of various forms (*e.g.,* distributed attacks are detected at an aggregated level) and varying rates (*e.g.,* low-rate attacks are detected within a guaranteed time period). Legacy threshold-based diagnosis methods only consider traffic rate as attribute for attack detection that makes it relatively easier for stealthy attackers to subvert the diagnosis systems. To address this shortcoming, we identify (§IV-B) key attributes of network traffic that are collectively able to distinguish benign versus malicious behavior of external sources. Using these attributes, we extend (§IV-C) our theoretical threshold-based hierarchy to a practical machine learning-based diagnosis system that employs anomaly detectors at three layers of host, subnet, and AS. We evaluate the performance of our scheme (profiling external sources using our novel data structure combined with ML-based models) in detecting distributed attacks with high accuracy, and highlight its superior detection ability in comparison with simple and hierarchical thresholding-based diagnosis methods.

A. Hierarchical Data Structure

In order to profile the volumetric behavior of an attacker, we employ a graph-like data structure to capture the relationship between external source entities and internal destination hosts of a given network. External source entities can be identified by a hierarchy of: hosts under subnets under ASes. As noted in §III, DNS query scanners and flooders are likely to be located within certain ASes and/or subnets. Therefore, it is important to develop a comprehensive model that covers traffic

Algorithm 1 Multi-thresholding Retention Duration

```
procedure ACTIONPKTIN
    while PktIn do
        for n from MaxLevel to 0 do
            if PktIn not match any node on level n then
                create corresponding node and edge with
                retentionDuration as 2^{n-1}R
            else
                update the matched node and edge
procedure ACTIONATTACKERDETECTED
    for n from maxLevel to 0 do
        if node_{n,i} on level n is anomaly then
            if n > 1 then
                set retention duration of its child nodes
                on level n − 1 to their parent level
```

activity of external sources at various levels of aggregation (as opposed to purely individual hosts level), enabling us to detect sophisticated attacks that get distributed across a number of hosts, subnets, or ASes, aiming to evade threshold-based diagnosis by lowering their traffic rate.

1) Key Design Rationales: External entities can be identified by a hierarchy of: hosts under subnets under ASes. As noted in §III-B, DNS query scanners and flooders are likely to be located within certain ASes or subnets. Therefore, it is important to consider a comprehensive model that covers traffic activity of external entities at various levels of aggregation, enabling us to detect sophisticated attacks that are distributed or go under the radar (by lowering their traffic rate). Since the set of external entities can be quite massive (*i.e.,* potentially the whole IPv4 space on the Internet), it becomes impractical to keep states forever. On the other hand, forgetting states too quickly may result in missing slow attackers. Therefore, we need an efficient retention policy to age out inactive entities in our data structure.

2) Theoretical Framework of Our Data Structure: Given the above requirements, we construct a binary hierarchical attributed graph model [43], as shown in Fig. 7. In this diagram, enterprise internal hosts are represented by circles at the bottom, connected via solid edges to external entities (shown by filled squares) which themselves get aggregated to upper-level entities via dashed edges. We apply aggregation to both nodes and edges in this model. Level-1 entities each represents an external host IPv4 address on the Internet (*i.e.,* /32), while level-2 entities are a group of external hosts created by masking one bit in the IPv4 address (*i.e.,* /31) – all the remaining levels work by incrementally masking the IP to get a larger subnet. To visualize the edge aggregation let us focus on two leftmost level-1 hosts (*i.e.,* $eH1$ and $eH2$) shown in Fig. 7. They both have a connection (*i.e.,* $e1$ and $e2$) to the first internal host (*i.e.,* $iH1$). These two level-1 edges are aggregated as a single edge $e3$ at level 2.

Dynamic retention policy: Nodes (*i.e.,* an external entity) and edges in our model each would have a set of attributes (explained in §IV-B), describing their profile, and they will be removed dynamically from the graph if they become inactive for longer than a corresponding retention duration defined in Algo. 1.

A new node (with an edge) is created at a level where an incoming packet does not match any existing node at that level in the graph (this needs to be checked for every level). Note that each node at level $n+1$ has two children nodes at level n because of the binary nature of subnetting operations. We, therefore, choose to initialize T_{n+1}, the retention period of nodes at level $n+1$, to the sum of retention periods of its children ($T_{n+1} = 2T_n$, where $T_0 = R$ a constant value set by an administrator). Upon creation of a node (with an edge), a default retention duration (*i.e.,* $T_n = 2^{n-1}R$) of its corresponding level n is assigned to the new node and edge. One may consider lager factors for the initial retention value (*e.g.,* $3^{n-1}R$) set for levels of the hierarchy. We note that longer retention period can result in improved visibility, but at much higher computing costs since a larger number of states need to be maintained. If the incoming packet matches an existing node and edge, then the corresponding retention duration is re-initialized.

Upon detection of an attacker node (at level n), the retention period of the two child nodes (*i.e.,* at level $n-1$) gets updated to the same value of the level n. We double the retention period of children for longer monitoring.

3) Detecting External Scanners & Flooders: We now show how our proposed model can detect scanners, especially those with low rate probing activity. Let's assume a simple threshold N is employed to detect scanners. A scanner node with retention duration T_n can be detected if it probes α internal hosts per epoch time and the condition (1) below is satisfied.

$$\alpha T_n \geq N \tag{1}$$

Detecting a slow scanner: A slow scanner can go undetected since its node/edge is removed from the graph every period of retention before hitting the threshold N. Instead, a higher level node with a larger (*i.e.,* power of two) retention duration T_{n+1} will be flagged as a scanner, and thereby the retention duration T_n of the child nodes gets updated. The time needed to detect the scanner child node is given by:

$$2\frac{N}{\alpha} - T_n \tag{2}$$

Proof of formula 2. It takes $\frac{N}{\alpha}$ for the retention policy of a scanner node to get updated to a higher value of its parents (*i.e.,* time taken to detect a parent/root node as attacker). Since the scanner node has already had αT_n edges with internal hosts, it needs to accumulate $N - \alpha T_n$ more edges before being detected. This takes $\frac{N}{\alpha} - T_n$ with the probing rate α. Thus, in total, it take $2\frac{N}{\alpha} - T_n$ for a successful detection at the level of scanner node.

This process will be sequentially passed to nodes at lower levels until a successful detection at the lowest level ($n = 1$) is achieved. Hence, given the simple detection criteria (*i.e.,* condition (1) above), an external scanner can be detected within a guaranteed time t_{detect}:

$$t_{detect} = \begin{cases} \frac{N}{\alpha}, & \text{if } T_1 \geq \frac{N}{\alpha} \\ n\frac{N}{\alpha} - \sum_{i=1}^{n-1} T_i, & \text{if } T_{n-1} < \frac{N}{\alpha} \& T_n \geq \frac{N}{\alpha} \end{cases} \tag{3}$$

Proof of formula 3. If the probing rate is high enough for the host being detected within its initial retention period, then the detection time is $\frac{N}{\alpha}$. Otherwise, t_{detect} is derived by aggregating multiple processes defined in formula 2 till the lowest level.

Detecting distributed scanners: If a scan is performed by k scanners each having rate α that cannot be detected at host-level by default T_1 (*i.e.,* $\alpha T_1 < N$), it is possible (under certain conditions) to detect them at host-level earlier than $n\frac{N}{\alpha} - \sum_{i=1}^{n-1} T_i$, as computed in formula (3). In the best-case scenario, if all scanners are immediate neighbors of each other in Fig. 7 (*i.e.,* quickly converge to one node at higher level), and a root node (a parent covering all scanners) at higher levels has a sufficiently large retention period, it takes t_{detect} given by:

$$t_{detect} = \frac{N}{\alpha} + \sum_{i=1}^{n-1}(\frac{N}{2^i\alpha} - T_i) \tag{4}$$

Proof of formula 4. Detection time t_{detect} is derived in a similar way as in formula 3, except that the scanning rate for a node at level n is $2^{n-1}\alpha$ instead of α due to aggregation.

Otherwise, if the root node cannot be detected within its default retention period, then the detection process takes longer since the root node needs to be first detected as a scanner (by updating the retention period from upper layers). In this scenario, the detection time t_{detect} is given by:

$$t_{detect} = \frac{N}{\alpha} + \sum_{i=1}^{-1+\log_2 k}(\frac{N}{2^i\alpha} - T_i) + \sum_{j=\log_2 k}^{n-1}(\frac{N}{2^{-1+\log_2 k}\alpha} - T_j) \tag{5}$$

Proof of formula 5. From level 2 to level $\log_2 k$, all children of a scanner parent are scanners, while from level $1 + \log_2 k$ to level n, each parent scanner has only one scanner child as explained in scenario for formula 3. Thus, the detection time is a combination of both processes.

To summarize, a scan can be detected at the host level within the duration t_{detect} given by formula 6, while the best-case is that all scanners are immediate neighbors with high scanning rate, and the worst-case is when scanners are located sparsely on the graph each with low scanning rate. In other words:

$$\frac{N}{\alpha} \leq t_{detect} \leq n\frac{N}{\alpha} - \sum_{i=1}^{n-1} Ti \tag{6}$$

where, n is the level at which the scan is first detected (*i.e.,* $\alpha T_n \geq N$ and $\alpha T_{n-1} < N$).

Proof of formula 6. The best-case scenario is when the scanning attack is detected (at the host level) within default retention period, while the worst-case is that all attackers can not be detected within default retention period and they are sparsely distributed.

Detecting distributed flooders at aggregation level: Our proposed model can detect flooders at the highest aggregation level (*i.e.,* root node) when a group of flooders is involved. This enables us to effectively detect (and mitigate) distributed attacks.

We define a simple threshold N for detecting flooder hosts, similar to that in condition (1) but for the rate of incoming queries. If an external host queries an internal node with more than N packets per time epoch, it gets detected. Considering aggregation, a node at level n is detected as a flooder if its query rate α exceeds $N * 2^{n-1}$. Hence, given k immediate neighbor flooder hosts with attack rate α, our data structure is able to detect distributed attacks at the highest aggregation level n_{detect} given by:

$$n_{detect} = 1 + \log_2 k \tag{7}$$

4) Practical Design Choices: We have demonstrated the efficacy of our mathematical-based model to detect challenging (low-rate and/or distributed) scans and floods. To realize and further improve this model for detecting volumetric attacks in real networks at scale, we consider three key design choices as follows.

First, instead of aggregating external source entities sequentially by subset mask, we track their activities in our model using a **three-level hierarchy** including AS level, subnet level (*i.e.,* registered subnets under the administration of each AS), and host level (*i.e.,* IP address of individual external source hosts). Second, since applying thresholds only on the traffic rate of an external host cannot isolate low-rate attackers from normal users, we use a **collection of attributes** to accurately model the behavior of external hosts individually as well as at aggregated levels (*i.e.,* subnets and autonomous systems), as explained in §IV-B. We replace the thresholding-based detection function with anomaly detection techniques. To enhance resilience against morphed attacks that deviate from known signatures [47], we train our models **only** with the behavior of benign external entities (*i.e.,* hosts, subnets, and ASes) and hence detect anomalies as described in §IV-C. Third, to scale our solution it is important to react quickly, and manage costs of memory access efficiently since a large amount of data needs to be processed in real-time. We, therefore, use online algorithms [32] to receive one data point at a time and use it to update a set of attributes – the required statistics (variance and average) are computed in a single pass when costs of memory access dominate those of computation. We compute the variance attribute using Welford's method [53] and the average attributes are computed by exponential averaging. After the update, the data point is discarded and only the updated attributes are kept in memory.

B. Identifying and Computing Attributes of DNS Traffic for External Entities

We analyzed the DNS query behavior of external hosts (with their subnets and autonomous systems) in §III-B. Given the insights obtained from real attackers, we now identify and compute important attributes (for each external entity) needed for our detection models to distinguish normal external entities from anomalous ones.

1) Attributes: We showed in §III-B, DNS-based attackers tend to craft query packets using a set of predefined domain names. We define our first attribute as ***varPktSize*** (*i.e.,* variance of packet size sent by the external entity), since the size of query packets sent by scanners and/or flooders are less variant compared to queries from normal (legitimate) hosts.

Table II: Summary of data cleansing for the university network (1 May 2018).

Reason of Removal	Unanswered	NameError	ServerFailure	Refused	NotImplemented	FormatError
Number of Pkts	9,970,082	689,422	40,247	29,155	21	18

Normal external users (including individual clients and recursive resolvers) may only query a limited number of internal hosts (*i.e.,* DNS servers of the organization). Scanners, on the other hand, contact a larger number of internal hosts (especially at relatively large time-scales). We therefore choose our second attribute as ***numHostQry*** (*i.e.,* number internal hosts queried by each external entity).

Both external flooders and heavy scanners send a much larger number of query packets (in total) to the enterprise network compared to normal external users. Flooders focus on one or a set of internal hosts, whereas scanners sweep over a wider range of internal hosts. So, we choose our third attribute as ***avgPktCountHost*** which is the average number of query packets sent to each internal host contacted.

For our last attribute (*i.e.,* ***varPktCountHost***), we compute the variation of query packet count sent to each internal host by an external entity. Note that the value of this attribute is smaller for scanners compared to flooders and legitimate users, since scanners tend to send an identical number of query packets (*e.g.,* one or two) to internal hosts of interest.

High-profile DNS query-based attacks can be quickly detected by either *numHostQry* (for heavy scanners) or *avgPktCountHost* (for heavy flooders). For relatively low-profile attacks (*i.e.,* distributed floods and/or slow scans), we need an enhanced visibility of the behavior of external entities using the collection of four attributes, aggregated-level models with dynamic retention policies. Next, we discuss how these attributes for each external entity (*i.e.,* node in the graph model) are computed in real-time.

2) Managing States for Edges: As explained in §IV-A, an external entity is related to an internal host using an edge in our graph model. We dynamically update the four attributes of each external entity every epoch time (*e.g.,* one minute) using a number of states maintained for the graph edges.

For an edge, we track three main states including total number of packets (denoted by N_p), total volume of packets (denoted by V_p), and variance of packet size (denoted by σ_p) during each epoch – arrival of a DNS query updates these three states for the corresponding edge. Edge states are exponentially averaged (with weighting factor of 0.9) at the end of each epoch – we employ the Welford's method [53] to compute σ_p in a single pass (*i.e.,* online algorithm).

Computing Attributes: Given edge states, the four attributes of an external node are computed as follows. *numHostQry* equals the number of edges associated with the node; *varPktSize* equals the weighted average of σ_p across all edges (*i.e.,* $\Sigma\frac{\sigma_p.N_p}{\Sigma N_p}$); *avgPktCountHost* equals the average N_p across all edges (*i.e.,* $\frac{\Sigma N_p}{numHostQry}$); and, lastly *varPktCountHost* can be derived by computing the standard deviation of N_p on all associated edges.

C. Anomaly Detection Model

We now train, tune and validate the accuracy of three anomaly detection models, namely host-level, subnet-level and AS-level for external source entities. We note that the intended pattern of inbound DNS traffic can vary across different enterprises, depending on the richness and size of their infrastructure, and their services offerings. Therefore, each network would have its own set of models trained by their own records of DNS traffic activity to achieve the best detection performance. To make the training process portable across enterprises, our engines for data cleansing, producing training set, and generating models are fully automated and consume DNS logs (*i.e.,* PCAP files) as inputs.

1) Dataset Preparation: Benign instances obtained from real DNS traffic of each enterprise are used for training anomaly detection models of the corresponding network. Additionally, we generate and collect data of DNS attacks including scans and floods of varying rates in our lab testbed. This attack dataset (together with the benign dataset) is used to evaluate the accuracy of our host-level anomaly detection.

Benign dataset: We clean our raw dataset (of 1st May) by removing unanswered and invalid queries, and use it for generating benign instances. We acknowledge that our cleaned dataset can still contain "not purely benign" instances, thus we tune a hyper-parameter called "contamination level" during model training to reduce the effect of outliers in the dataset. For example, we extracted 3.6M DNS queries (as benign) after removing 10.7M queries from the university dataset[3] on 1 May – details of removed queries are shown in Table II. Using cleaned data, we generated 32.4M, 24.5M, 11.1M benign instances (of 1-minute granularity) for the host-, subnet-, and AS-level models respectively.

Attack dataset: We set up an isolated testbed in our lab to emulate an enterprise network communicating with external hosts via a border router. The internal and external networks were configured with a subnet from a /16 IPv4 address prefix. We configured a DNS server (running BIND 9) and one regular host inside the enterprise zone, and 3 DNS servers and 2 attacker machines (running a customized script using the Python Scapy library) on the external zone – each machine running Ubuntu 16.04.4 and equipped with a 2.1GHz CPU and 8GB RAM.

Our attack script running on the two machines (*i.e.,* M1 and M2) generated query scans (from M1) to the entire IP range and query floods (from M2) on the DNS server of the internal zone (enterprise network). We generated a diversity of attack patterns by varying three parameters for each attack type. For query scans, we varied parameters as follows: the query rate from 1 to 72K (in 12 steps) packets-per-hour; query names selected randomly from a varying size (1 to 10) of a pre-populated list of the university sub-domains; and, the

[3]We omitted results of data cleansing and model training for the research institute. Fairly similar observations were made in both organizations.

Table III: Model tuning (host-level).

Cont. Level	Accuracy	TN	TP	AUC
0.0001	99.39%	99.98%	65.35%	76.94%
0.0002	99.42%	99.96%	68.41%	89.34%
0.0005	99.54%	99.89%	79.45%	97.05%
0.0010	**99.76%**	**99.81%**	**97.21%**	**99.96%**
0.0020	99.57%	99.61%	97.43%	99.91%
0.0050	99.03%	99.04%	98.524%	99.74%
0.0100	98.03%	98.03%	98.55%	98.86%
0.0200	96.09%	96.05%	98.74%	97.05%
0.0500	90.16%	90.01%	99.13%	61.34%

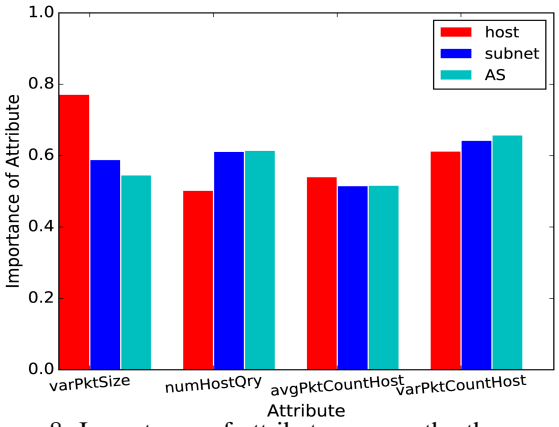


Figure 8: Importance of attributes across the three models.

number of queries per internal host from 1 to 4. For query floods, the rate varied from 1 to 300 (in 12 steps) packets-per-second; query names were selected similar to query scan attacks; and, the query types were chosen from four common types (*i.e.,* `A`, `AAAA`, `ANY`, `PTR`). In total, we generated 480 scans and 480 floods with each attack lasting for 3 hours. Note that multiple attacks, each sourced from a unique crafted IP address, were scheduled in parallel. We collected data of the attacks on the testbed, computed attributes of external attackers during their activity, and generated 556,258 instances at host-level and 13,904 instances for both subnet- and AS-level – all instances associated with one subnet and one AS.

2) Attribute Analysis: We now employ "information gain" (IG) metric to quantify the impact of our attributes in distinguishing benign and anomalous instances. This metric measures the correlation between each of the attributes and the predicted output of decision trees (valued between 0 and 1, where 0 indicates an irrelevant attribute and 1 highlights an important attribute). This method calculates the reduction of entropy values by excluding a certain attribute from prediction. We note that training a model with attributes of low information-gain can lead to issues like overfitting, since the classifier gets trained by noise or less-relevant information.

We show in Fig. 8 the importance of our four attributes for host-level, subnet-level, and AS-level models. It can be seen that the importance all attributes (across the three models of our hierarchy) is larger than 0.5, and hence carrying a significant amount of information in differentiating benign and malicious entities. In addition, we observe that ***varPktSize*** (with importance value equal to 0.772) is a fairly important attribute for the host-level model – this is because an anomalous external host will likely craft DNS packets of the same size, and hence resulting in a low (close to zero) variance for their packet size. Also, it is seen that the importance of ***numHostQry*** and ***varPktCountHost*** slightly increases from the host-level model to subnet-level and AS-level models, highlighting that these attributes become more influential at aggregate levels.

Table IV: Model tuning (subnet-level and AS-level).

Cont. Level	TN of subnet model	TN of AS model
0.0001	99.98%	99.98%
0.0002	99.97%	99.96%
0.0005	99.96%	99.90%
0.0010	**99.91%**	**99.82%**
0.0020	99.80%	99.63%
0.0050	99.02%	98.97%
0.0100	98.02%	97.94%
0.0200	96.08%	95.87%
0.0500	89.99%	89.91%

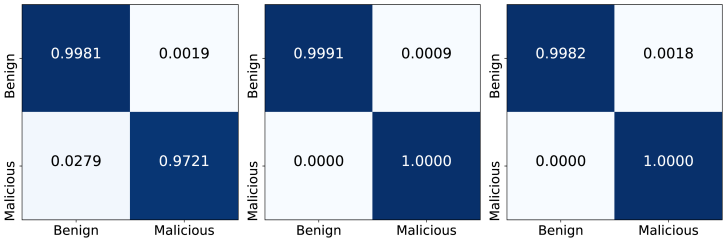
3) Model Tuning and Evaluation: We considered two widely-used anomaly detection algorithms, namely *isolation Forest* [29] (decision-tree based) and *one-class SVM* [36] (high dimensional distribution-based). We evaluated the accuracy of these algorithms using 10-fold cross validation on the benign dataset (obtaining the True Positive rate) and testing on the attack dataset (obtaining the True Negative rate). Note that True Positives indicate benign instances that are correctly classified as benign, and True Negatives are attack instances that are correctly labeled as attack.

To be more specific, we trained and evaluated our model for each algorithm by varying the contamination-level parameter[4] for the isolation Forest, and kernel functions (*i.e.,* linear, Gaussian, polynomial, sigmod, and RBF) for the one-class SVM. For each set of tuning parameters or functions, the training dataset (benign only) is randomly partitioned into ten equal size subsets. Of the ten subsets, nine are used as training data, and the remaining subset is retained as the validation data for testing the model. During the testing phase, benign instances from the single subset of validation is used to compute the rate of True Positive (TP) while the entire attack dataset is used to compute the rate of True Negative (TN). We found that the isolation Forest model outperforms the one-class SVM model by both TP and TN rates. Given the multi-centric distribution of our benign dataset (instances are geometrically located in several clusters [17]), the SVM model at best yields an overall accuracy of 63.6% (TN rate of 63.5% and TP rate of 74.6%).

We tuned and evaluated[5] the host-level model of isolation Forest considering both TP and TN rates, as shown in Table III. We can see that increasing the contamination-level causes the TP rate to fall monotonically (from 99.98% to 90.01%), since the algorithm excludes more training instances at the boundary of benign clusters when the contamination-level gets larger. On the other hand, the TN rate is positively correlated with the contamination-level, as low-profile attackers become

[4]A value between 0 and 1 that indicates the fraction of anomalies (*i.e.,* not benign instances) in the training dataset.

[5]In our evaluation, we used a mixed dataset (of labeled benign and attack instances) to compute overall accuracy, true negative rate and true positive rate.



(a) The host model. (b) The subnet model. (c) The AS model.

Figure 9: Confusion matrix of best-performing selected models at: (a) host, (b) subnet, and (c) AS, levels – columns correspond to true labels and rows correspond to predicted labels.

similar to benign hosts in their attributes. We, therefore, set the contamination-level to 0.001, resulting in the best overall accuracy among all models and a reasonably high rate of both TP and TN.

By tuning and evaluating models of subnet-level and AS-level, we observe a similar impact of the contamination-level on TN rate (as shown in Table IV) – the TP rate would be consistently 100% (except for the contamination-level at 0.0001) since our attack instances at subnet-level and AS-level represent a network of high-profile attackers. Given the optimal value of the contamination-level (obtained from the host model), we achieve TN rates of 99.91% and 99.82%, and AUC values of 99.95% and 99.98% for the subnet-level and AS-level models, respectively. The three best-performing models trained with a contamination-level equal to 0.0001 are selected for our university dataset, and their performance is summarized by confusion matrices shown in Fig. 9.

By applying our best-performing models on the proposed graph data structure to the synthetic attack dataset, all external attackers are detected at the host level – 91.37% are detected immediately (within the first minute of their commencement); 3.45% (low-rate scans and floods) are detected in 2 minutes; and only 1.72% of very low-rate scans (with the rate of 1 packet per second) are detected after 8 minutes of their commencement.

Impact of Training Data Composition on Detection Performance: We have so far generated an almost perfect isolation forest model (accuracy of 99.76%) by training it on the purified data (benign-only). It has been shown in [30] that fine-tuning the contamination-level (during training) may make no or little difference to the model performance in certain situations where the training data consists of well-distinguished malicious and benign instances (mix of "black and white" instances) with ground-truth labels. We, therefore, quantify the impact of impurified data (raw DNS data, consisting of benign and malicious instances) on the performance of detection for the host-level model. We tune the contamination-level ranging from 0 to 0.2 is steps size of 0.005. We observe that the model gives its best overall accuracy of 93.33% (true positive 86.94%, and true negative 99.72%) at the contamination-level set to 0.085. This mis-detection of attack instances (low true positive) highlights the fact that inclusion of anomalies in the training data can be detrimental to the performance of isolation forest one-class classifier (the boundary of the model becomes loose), especially in absence of ground-truth labels

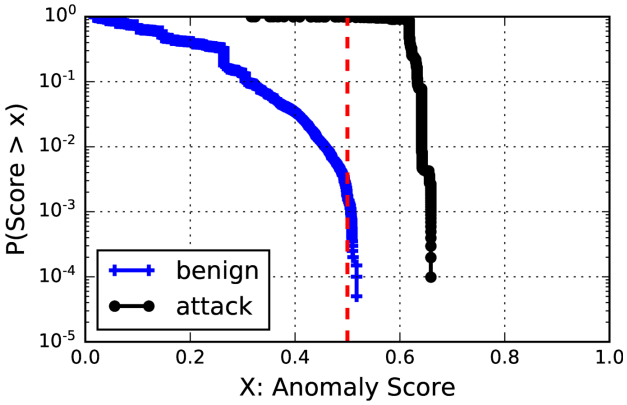


Figure 10: CCDF of anomaly score instances in evaluation dataset (host-level model).

for relatively uncertain benign and malicious instances ("gray" instances).

Understanding False Alarms of Our Model: The isolation Forest algorithm outputs a score of anomaly (*i.e.,* a value between 0 and 1) for a given instance – where 0 score means purely normal and 1 indicates a definite anomaly. We use the anomaly score to quantify the severity of malicious behaviors for external entities – a high anomaly score indicates that the detected external entity strongly displays the behavior of an attacker (*e.g.,* high rates attack traffic or highly repetitive packet contents), whereas lower anomaly scores suggest that the anomalous external entity has slightly deviated from the expected normal behaviors (*e.g.,* a slow-rate scanner). We show in Fig. 10 the CCDF of anomaly score for our ground-truth evaluation instances (benign and attack). It can be seen that about 90% of benign instances (shown by blue lines and cross markers) have a low score of less than 0.3. We also observe that the anomaly score for only a small faction of benign instances (*i.e.,* 0.2%) exceeds the red boundary line (*i.e.,* 0.5) in Fig. 10, separating benign and malicious entities. We found that the reason for benign instances getting larger anomaly scores was their *avgPktCountHost* attribute which was slightly higher, compared to other benign instances. For example, an external host (in the benign dataset) had sent 9,492 query packets (all responded with `NoError` flag) to 12 internal hosts in an hour – we are not able to verify if this host (and it's behavior) was illegitimate or not.

Moving to the distribution of anomaly score for attackers (shown by black lines and circle markers in Fig. 10), about 85% of attack instances result a score of more than 0.6. We also see a tiny fraction (2.8%) of attacks (floods and scans) have a score of less than 0.5 generating False Positive alarms – these instances mainly correspond to (a) the beginning of low-rate scans depending on the traffic rate, and (b) low rate floods with a diversified set of query names (*e.g.,* 10 query names). We note that these low-profile attacks may not be detected by the host model in a short timescale, but they ultimately get detected by the aggregated-level models (as explained in §IV-A).

Limitation of Our Evaluation: Note that our training dataset (benign) was obtained from a "real" production network while the attack traffic traces were collected from a lab

Table V: Summary of in-the-wild dataset – May 2018.

	# Qry. in	# Resp. out	# Resp. in	# Qry. out	# ext. host	# ext. subnet	# ext. AS
University Campus	520,229,825	219,095,708	629,058,944	709,593,940	374,348	81,778	25,263
Research Institute	351,733,547	126,321,729	178,292,559	212,671,912	179,568	51,389	18,819

testbed ("synthetic"), and hence the overall evaluation mix may not necessarily represent realistic traffic instances. For future studies, one may want to consider a more comprehensive attack dataset, covering a wider set of volumetric DNS attacks (direct and/or distributed) that are collected from real networks.

4) Testing the Dataset with a Commercial Firewall: We replayed our mixed dataset through a typical enterprise next-generation-firewall (Palo Alto Networks firewall appliance PA-3020 [41]) which was configured using the vendor's official user manual [42]. This firewall generates *Host Sweep* and *UDP Flood* alerts for DNS scans and floods, respectively. We found that the firewall missed 83% of scanners and 16% of flooders (*i.e.,* in total 51% **true positives**) in our dataset, according to its threat logs – note that all missed attackers had a low rate (*i.e.,* below 20 hosts every 10 seconds for scanners and 100 packet-per-second for flooders) of DNS traffic. Also, the firewall generated no alert for benign instances (*i.e.,* 100% **true negatives**) .

Comparing these results obtained from the firewall with those in Table III, it can be seen that our anomaly detection scheme performs much better than the firewall by considering the TP metric (*i.e.,* 97.21% in Table III versus 51% highlighted in the above paragraph). In terms of the TN metric, the performance of our scheme is very close to that of the firewall (*i.e.,* 99.81% in Table III versus 100% stated in the above paragraph). Note that our scheme raises malicious alarms for a tiny fraction (*i.e.,* 0.18%) of benign instances due to impurities considered for the training dataset.

5) Testing the Dataset with Pure Thresholding of Our Hierarchical Structure: In order to evaluate the efficacy of our hierarchical structure (with appropriate parameters R, α, N from §IV-A3), we applied thresholds on the packets rate in our dataset (mixed benign and malicious). Our thresholds are chosen according to industry best practices [42] (*i.e.,* 120% of maximum normal value). For detecting flooders, we set host-level, subnet-level, and AS-level thresholds on ***avgPktCountHost*** equal to 51, 64, and 82, respectively. Also, for detecting scanners, thresholds on ***numHostQry*** are set to 8, 13, and 15 at respective levels of the hierarchy.

Our evaluation results show that all of the attacks (100%) are detected by AS-level and subnet-level thresholding, while 89.42% of scans and 64.75% of floods are detected at the host-level, resulting in an overall true positive rate of 77.08%. Also, considering the true negative rates, we observe 98.24%, 98.57%, and 99.18% at AS-level, subnet-level and host-level, respectively.

In summary, applying the thresholding method of our hierarchical structure yields 99.18% TN and 77.08% TP at the host level, while our anomaly detection model gives 99.81% TN and 97.21% TP at the host level (Table III), and the legacy thresholding employed by a commercial firewall results in 100% TN and 51% TP (§IV-C4). This shows that our hierarchical data structure significantly improves the performance of thresholding methods, but it is less performant when compared with our anomaly-based approach which models more comprehensive dynamic behavioral profiles.

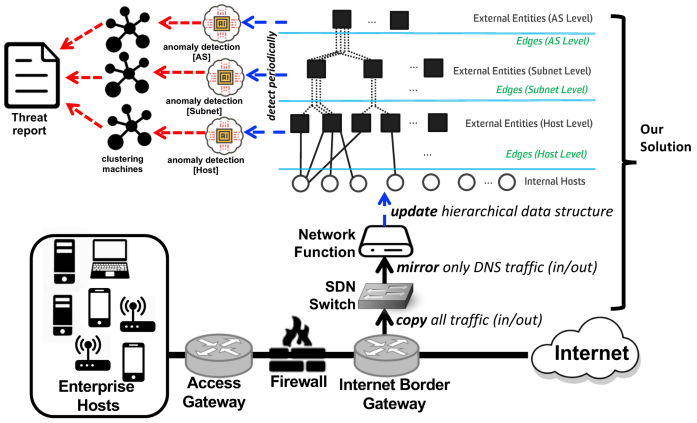


Figure 11: Prototype implementation.

V. Implementation and In-the-wild Detection

In this section, firstly, we demonstrate the efficacy of our detection method using a replay of DNS traffic collected from the two enterprises over a month. We then draw insights into the severity of attacks detected with a closer look at examples of low-profile scans and distributed floods. Next, we compare the output of our system using a public blacklist from Symantec and a commercial firewall from Palo Alto Networks. Finally, we quantify the performance of our system and show how we can detect DNS attacks in real-time with acceptable memory and CPU footprints for a one month period with real traffic of a large enterprise.

A. Prototype Implementation

Fig. 11 depicts the prototype implementation of our real-time system that is deployed in our lab processing full DNS traffic copied from the border of both enterprise networks. We use an SDN switch (NoviFlow 2122 [39]) that takes a full copy of enterprise Internet traffic (both inbound and outbound) and selectively mirrors only DNS traffic (source or destination port number equal to 53) to a compute node (*i.e.,* network function). All software modules are implemented on a generic server (equipped with 16 2.10GHz CPUs and 64GB RAM) running Ubuntu 16.04.4. Our packet processing module (*i.e.,* network function) written in the Golang language using the Data Plane Development Kit (DPDK) and the Intel NFF-Go library. It extracts necessary attributes (*i.e.,* source/destination IP/port and packet size) from incoming query packets, and passes them to update our hierarchical data structure also written in Golang. We use IPASN data files (*i.e.,* ".dat" files)

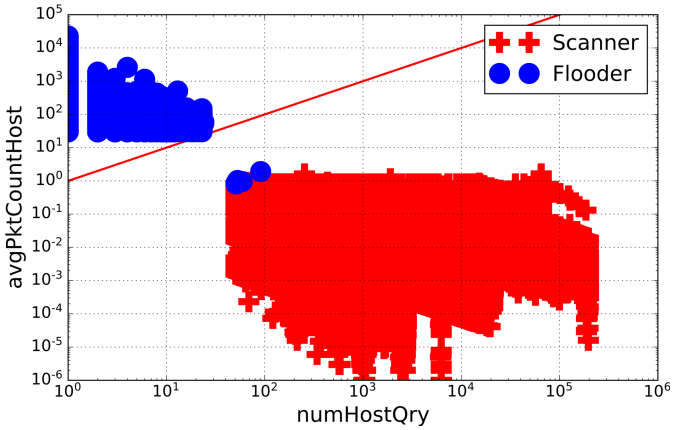


Figure 12: Clustering external anomalous hosts.

to map external IPs to their corresponding subnets and ASes, which are generated from MRT/RIB BGP archives [5]. Our proposed detection modules are implemented using Python3 to be compatible with machine learning utilities. For decision-making functions, we use the best-performing anomaly detection models (from §IV-C) trained with the contamination-level equal to 0.001 (at all three levels: host, subnet, and AS) – they perform reasonably well by both TP and TN metrics.

B. Summary of Dataset

A summary of our in-the-wild dataset collected over the month May 2018 is shown in Table. V. For the university network, we have a total of 2.0B DNS packets including 520M incoming queries, 219M outgoing responses, 709M outgoing queries and 629M incoming responses. Focusing on external entities who sent DNS queries to the network, we see a total of 374K unique hosts associated with 81K subnets and 25K ASes. For the research institute, there were 867M DNS packets of which 351M packets were incoming queries sourced from 179M external hosts associated with 51K subnets and 18K ASes on the Internet.

We replayed the dataset of each organization on our system with the corresponding models. Instances were created at runtime for trained models to predict whether they are normal or anomalous. We made a log of all anomalous instances detected by our system for post-analysis and drawing further insights. After evaluating the university dataset, we found 14785 external hosts (*i.e.,* 3% of total hosts), 7403 subnets, and 2415 ASes flagged as anomalous. Also, for the research institute, 4332 external hosts, 2121 subnets, and 922 ASes were detected as anomalous entities.

C. Clustering Anomalous Entities

In order to distinguish scanners from flooders (at all three levels), we applied an unsupervised clustering algorithm, *i.e.,* expectation-maximum (EM), to those entities detected as anomalous. For the clustering model, we used two of our previously identified attributes (in §IV-B) namely *avgPktCountHost* and *NumHostQry* as they collectively distinguish flooders from scanners.

As a result of clustering for the university network: 14171 flooders and 621 scanners were found at the host-level; 7493 flooders and 107 scanners were identified at the subnet-level; and, at the AS -level, 2430 and 47 flooders and scanners were found. For the research institute, 4332 hosts, 2122 subnets, and 921 ASes were identified as flooder entities, and also 490 hosts, 63 subnets, and 36 ASes were found as scanners.

Table VI: Top scanner ASes.

AS ID	Loc.	Subnet	Host
42570	CH	1	242
60781	NL	1	48
36375	US	1	32

Table VII: Top flooder ASes.

AS ID	Loc.	Subnet	Host
32934	US	11	865
16509	US	87	707
14618	US	59	495

Fig. 12 shows the scatter plot of two attributes, clearly separating flooders from scanners. As expected, flooders (shown by blue circle markers) are primarily located in the top left region of the plot while all scanners (shown by red cross markers) are grouped on the lower right region. Interestingly, we observe that several blue circles (clustered as flooders) are located very close to scanners group – this is because of their *avgPktCountHost* attribute value was higher than other scanners, and thus are classified as flooders.

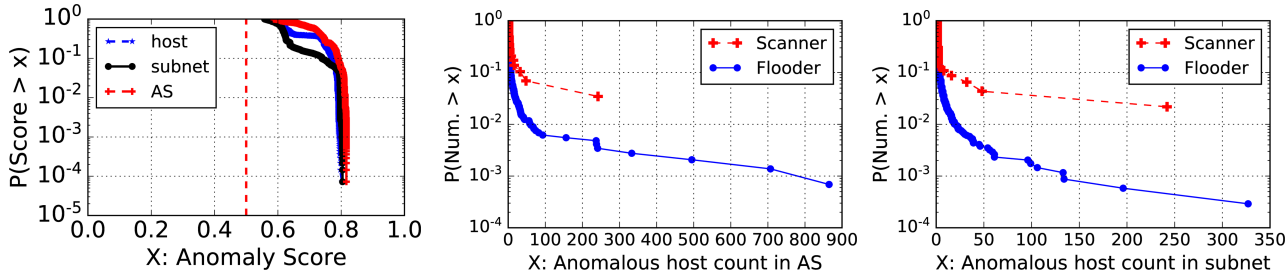
D. Anomaly Scores

We now consider attack profiles by checking the anomaly score as well as the distribution of flooders and scanners at various levels of aggregation. Higher anomaly scores indicate a larger deviation from normal values of attributes (*e.g.,* large packet rates, highly repeated DNS query size, or numerous internal host contacted). Fig. 13(a) shows the score of anomalous entities detected by our isolation Forest models for hosts, subnets, and ASes. The first observation is that the anomaly score of detected attackers from real networks are relatively larger (*i.e.,* well above the border line 0.5) compared to attacks generated in our lab. We also see that at least 10% of instances (in all three models) have the score value greater than 0.7, highlighting the confidence of our models in detecting these anomalous entities in the wild.

We show in Figures 13(b) and 13(c) the distribution of clustered anomalous entities. Fig. 13(b) shows the CCDF plot of the number of anomalous hosts in anomalous ASes. The majority of ASes cover less than 100 anomalous hosts (both flooder and scanners). We observe that one AS has about 250 scanners but 7 ASes have relatively a large number of flooders (possibly distributed). The same observation is made for anomalous hosts of subnets, as shown in Fig. 13(c). We verified that the tail of curves in Figures 13(b) and 13(c) correspond to large anomaly scores (*i.e.,* between 0.75 and 0.80). The top distributed scanner hosts are located in one subnet, while the top distributed flooder hosts are spread across many subnets (under the administration of one AS). We list top scanner ASes and flooder ASes in Tables VI and VII respectively, based on their count of anomalous hosts. It can be seen that the top distributed scanner hosts are located in one subnet, while the top distributed flooder hosts are spread across many subnets (under the administration of one AS).

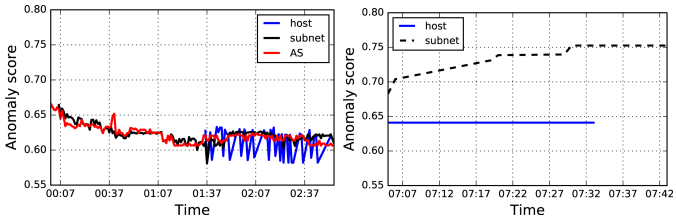
E. Two Representative Attacks

In order to demonstrate the efficacy of our scheme, we focus on two representative attacks (*i.e.,* a low rate scan and



(a) Anomaly score of anomalous instances. (b) Anomalous host count in anomalous ASes. (c) Anomalous host count in anomalous subnets.

Figure 13: Severity of attacks in the university network: (a) anomaly score, and (b, c) distribution of clustered anomalous hosts in anomalous subnets and ASes.



(a) A low-rate scan. (b) A distributed flood.

Figure 14: Anomaly detection examples: (a) low-rate scan, and (b) distributed flood.

a distributed flood) that are typically missed by traditional solutions.

Fig. 14(a) shows the time-trace of anomaly score at all three levels, tracking the evolution of our detection. This attack is the low-rate scan which we manually identified in §III-C, shown in Fig. 3(c). As mentioned earlier, 37 hosts from a /16 subnet performed a scan to the university network simultaneously each with the rate of less than 1 packet-per-hour. We observe that this attack is first detected at the AS-level (shown by red lines in Fig. 14(a)). It takes 7 minutes for the subnet model to raise an anomaly alarm (shown by black lines in Fig. 14(a)), and the host model starts detecting attackers after more than an hour and half (we plot the score for only one of 37 scanners as highlighted by blue lines in Fig. 14(a)). This detection was successfully achieved because of our hierarchical aggregation and dynamic retention policy (explained in §IV-A) – detection of the attack at AS-level increased the retention duration of child subnets, leading to a detection at the subnet-level (with some delay) which in turn elongated the retention policy of child hosts, enabling the host model to detect the scanner hosts.

For our second example of attack, we show in Fig. 14(b) the time-trace of anomaly score for one /16 subnet consisting of 7 flooders which participated in a widely distributed flood that we described in §III-D, shown in Fig. 4(a). We note that this subnet is the only subnet under its AS that has anomalous behavior – the AS looks normal otherwise. We can see in Fig. 14(b) that right from the beginning of this flood (*i.e.,* around 7:07AM on 2nd May), the subnet was detected as a heavily anomalous entity, while its AS was classified as normal. We observe that the flooder host (one of 7) is consistently flagged as anomalous with the score 0.64 due to its repeated flooding pattern, while the anomaly score of the subnet rises in time as more external hosts join this distributed attack.

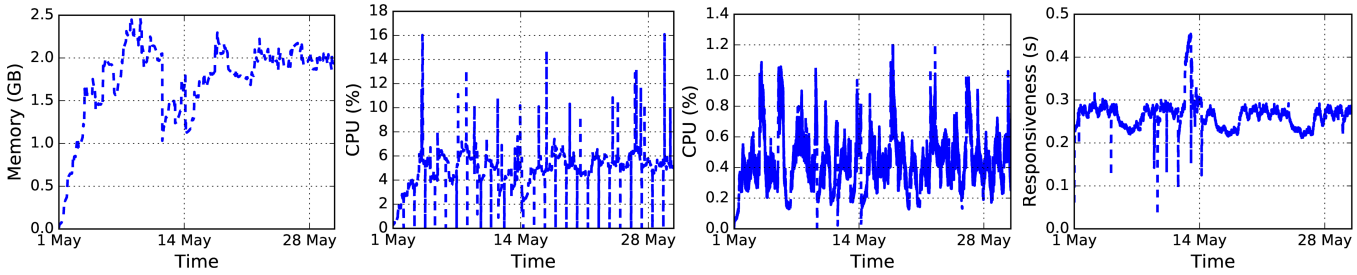
F. Comparison with Blacklist and Commercial Firewall

We selected 200 hosts[6] , those that are flagged during the entire month May 2018 – the top 100 hosts with the highest anomaly score and the bottom 100 with the lowest anomaly score (above the border line 0.5). We checked these hosts against an IP reputation repository (*i.e.,* blacklist) maintained by Symantec [50]. This web-based tool takes an IP address as input and reports if it was involved in malicious activities such as sending spam or spreading viruses. We found that the majority (*i.e.,* 63%) of hosts in our top 100 list are flagged as malicious IPs in the Symantec blacklist – 6 of them are scanners and 57 of them are flooders. Also, 58 hosts in our bottom 100 list are seen in the blacklist – all of these hosts are flooders.

Finally, to compare our system with a commercial firewall, we replayed our in-the-wild dataset through the Palo Alto Networks firewall appliance PA-3020 [41]. We extracted and analyzed the syslog file produced by the firewall during our traffic replay. The firewall detected 70 scanner IP addresses for the university dataset and 107 scanners for the research institute – this is a subset (11.3% for the university dataset and 21.8% for the research institute) of our detection results. Unsurprisingly, all of the scanners detected by the firewall had a high rate of probing, while none of low-rate scanners were flagged.

For the query floods, the firewall captured 5 distributed attacks in the university network and 2 attacks in the research institute, as those victim internal hosts received an excessive number of UDP packets within a short time interval. Although the firewall logged all external IPs that sent packets when the alarm was triggered, it was not possible to **precisely identify and locate attackers**. Our system, instead, not only detected all those distributed attacks flagged by the firewall, but also precisely captured the source (*i.e.,* external anomalous hosts, subnets or ASes). Besides, it is important to note that our system detected 9 flooding attacks (sourced from the bottom 100 list) for the university network, but none of them were alerted up by the firewall as the attack rate was relatively low. We manually checked these low-rate floods and found that they all sent repeated queries (with identical query name) to 6 non-DNS servers, 2 authoritative name servers, and 3 internal recursive resolvers.

[6]Automatic lookup of all flagged hosts in the blacklist is prevented by anti-robot image test.



(a) Total memory usage. (b) Updating the graph structure. (c) Extracting attributes & calling models. (d) Prediction responsiveness.

Figure 15: Real-time performance of our detection system under full load of the university campus network.

G. Real-Time Performance of Our Detection System

To demonstrate the practicality of our proposed scheme we quantify the performance of our system (explained in §V-A) with one month's worth of real traffic from the university network[7]. Fig. 15 shows the real-time utilization of memory and CPU in our prototype. The memory consumption is recorded every minute, as shown in Fig. 15(a). Also, we measure the CPU utilization for two separate processes, namely *Updating* the graph structure upon arrival of DNS packets (Fig. 15(b)), and *Extracting* attributes from the graph structure and *calling* models at three levels of hierarchy (Fig. 15(c)) – each process utilizes one CPU. Additionally we quantify the inference responsiveness (*i.e.,* time taken for computing attributes and obtaining results from models at the end of each epoch) as shown in Fig. 15(d). It is seen that all four metrics are bounded by reasonable values: the memory consumption is capped at 2.5GB, Updating and Processing respectively use up to 16% and 1.3% of CPU, and the system responds in less than 0.5s. Note that our system performance fluctuates (within a bounded region) due to the variation in the traffic. This demonstrates that our system can meet reasonable performance criteria typically required by enterprise network operators.

VI. Conclusion

Enterprise networks are the target of sophisticated DNS attacks in the form of query floods, reflection and amplification attacks, and scans. Existing security appliances are not well-equipped to protect network assets from dynamic attacks sourced from distributed and automated external hosts on the Internet. We have developed a method using anomaly-based detection models to automatically detect DNS floods and scans of varying rates sourced from one or a distributed set of external hosts. We highlighted the characteristics of malicious entities sending query-based attacks, developed a hierarchical and dynamic graph data structure for scalable monitoring and detection of scans and volumteric attacks, identified key attributes to effectively differentiate attacker entities versus normal external users, and employed anomaly detection models (trained/tuned using benign and attack traffic) in our dynamic data structure. Lastly, we demonstrated the efficacy of our scheme and compared our system with a public blacklist and a commercial firewall.

[7]We omit results of the research institute since its load was lower than the university network.

Acknowledgments

This work was completed in collaboration with the Australian Defence Science and Technology Group.

References

[1] Akamai Technologies, "Threat Advisory: Mirai Botnet," https://bit.ly/2UC1JfJ, 2017, accessed: 2017-12-3.
[2] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, II, and D. Dagon, "Detecting Malware Domains at the Upper DNS Hierarchy," in *Proc. USENIX Security*, Berkeley, CA, USA, Aug 2011.
[3] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From Throw-away Traffic to Bots: Detecting the Rise of DGA-based Malware," in *Proc. USENIX Security*, Bellevue, WA, USA, Aug 2012.
[4] R. Arends, R. Austein, D. M. M. Larson, and R.Rose, "DNS Security Introduction and Requirements," RFC 4033, Mar 2005. [Online]. Available: https://www.ietf.org/rfc/rfc4033.txt
[5] H. Asghari, "Offline IP address to Autonomous System Number lookup module," https://pypi.org/project/pyasn/, 2020, accessed: 2020-11-05.
[6] K. Bhardwaj *et al.*, "Towards IoT-DDoS Prevention Using Edge Computing," in *Proc. USENIX HotEdge*, Boston, MA, USA, Aug 2018.
[7] J. Bushart, "Optimizing Recurrent Pulsing Attacks using Application-Layer Amplification of Open DNS Resolvers," in *Proc. USENIX WOOT*, Baltimore, MD, USA, Aug 2018.
[8] Y. Chen *et al.*, "DNS Noise: Measuring the Pervasiveness of Disposable Domains in Modern DNS Traffic," in *Proc. IEEE/IFIP DSN*, Atlanta, Georgia, USA, Jun 2014.
[9] T. Chung *et al.*, "Understanding the Role of Registrars in DNSSEC Deployment," in *Proc. ACM IMC*, London, UK, Nov 2017.
[10] Cisco Blog, "Overcoming the DNS Blind Spot," https://blogs.cisco.com/security/overcoming-the-dns-blind-spot, 2016, accessed: 2019-05-15.
[11] Cisco Systems, "Protection Against Distributed Denial of Service Attacks," https://bit.ly/2WUbvvK, 2018, accessed: 2018-11-2.
[12] D. Dagon *et al.*, "Recursive DNS Architectures and Vulnerability Implications," in *Proc. NDSS*, San Diego, CA, USA, Feb 2009.
[13] N. Daswani and H. Garcia-Molina, "Query-flood DoS Attacks in Gnutella," in *Proc. ACM CCS*, Washington, DC, USA, Nov 2002.
[14] K. Du, H. Yang, Z. Li, H. Duan, and K. Zhang, "The Ever-Changing Labyrinth: A Large-Scale Analysis of Wildcard DNS Powered Blackhat SEO," in *Proc. USENIX Security*, Austin, TX, USA, Aug 2016.
[15] Z. Durumeric *et al.*, "ZMap: Fast Internet-wide Scanning and Its Security Applications," in *Proc. USENIX Security*, Washington, D.C. , USA, Aug 2013.
[16] EfficientIP, "EfficientIP and IDC: DNS Attacks Cost Nearly $1 Million Each, Increasingly Impacting the Cloud ," Global DNS Threat Report, 2020.
[17] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121 – 134, Apr 2016.
[18] S. K. Fayaz *et al.*, "Bohatei: Flexible and Elastic DDoS Defense," in *Proc. USENIX Security*, Washington, D.C., USA, Aug 2015.
[19] Fortinet, "FortiDDoS and Verisign DDoS Protection Service," https://bit.ly/2DsDObH, 2018, accessed: 2018-11-2.
[20] K. Fukuda, J. Heidemann, and A. Qadeer, "Detecting malicious activity with dns backscatter over time," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3203–3218, Oct 2017.

[21] T. Greene, "How the Dyn DDoS attack unfolded," https://www.networkworld.com/article/3134057/how-the-dyn-ddos-attack-unfolded.html, 2016, accessed: 2020-11-03.
[22] P. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," RFC 8484, Oct 2018. [Online]. Available: https://tools.ietf.org/html/rfc8484
[23] J. Krupp *et al.*, "Identifying the Scan and Attack Infrastructures Behind Amplification DDoS Attacks," in *Proc. ACM CCS*, Vienna, Austria, Oct 2016.
[24] M. Kührer *et al.*, "Going Wild: Large-Scale Classification of Open DNS Resolvers," in *Proc. ACM IMC*, Tokyo, Japan, Oct 2015.
[25] A. C. Lapolli, J. Adilson Marques, and L. P. Gaspary, "Offloading Real-time DDoS Attack Detection to Programmable Data Planes," in *Proc. IFIP/IEEE IM*, Arlington, VA, USA, Apr 2019.
[26] H. Li, H. Hu, G. Gu, G.-J. Ahn, and F. Zhang, "vNIDS: Towards Elastic Security with Safe and Efficient Virtualization of Network Intrusion Detection Systems," in *Proc. ACM CCS*, Toronto, Canada, Oct 2018.
[27] B. Liu *et al.*, "Who Is Answering My Queries: Understanding and Characterizing Interception of the DNS Resolution Path," in *Proc. USENIX Security*, Baltimore, MD, USA, Aug 2018.
[28] D. Liu, S. Hao, and H. Wang, "All Your DNS Records Point to Us: Understanding the Security Threats of Dangling DNS Records," in *Proc. ACM CCS*, Vienna, Austria, Oct 2016.
[29] F. T. Liu *et al.*, "Isolation Forest," in *Proc. IEEE ICDM*, Pisa, Italy, Dec 2008.
[30] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-Based Anomaly Detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, Mar. 2012.
[31] Z. Liu, H. Jin, Y.-C. Hu, and M. Bailey, "MiddlePolice: Toward Enforcing Destination-Defined Policies in the Middle of the Internet," in *Proc. ACM CCS*, Vienna, Austria, Oct 2016.
[32] J. Loveless, S. Stoikov, and R. Waeber, "Online algorithms in high-frequency trading," *ACM Queue*, vol. 11, no. 8, pp. 30:30–30:41, Aug. 2013.
[33] M. Lyu *et al.*, "Mapping an Enterprise Network by Analyzing DNS Traffic," in *Proc. Springer PAM*, Puerto Varas, Chile, Mar 2019.
[34] M. Antonakakis *et al.*, "Understanding the Mirai Botnet," in *Proc. USENIX Security*, Vancouver, BC, USA, Aug 2017.
[35] D. MacFarland *et al.*, "The best bang for the byte: Characterizing the potential of DNS amplification attacks," *Computer Networks*, vol. 116, pp. 12–21, Apr 2017.
[36] L. M. Manevitz and M. Yousef, "One-class Svms for Document Classification," *J. Mach. Learn. Res.*, vol. 2, pp. 139–154, Mar. 2002.
[37] W. Meng *et al.*, "Rampart: Protecting Web Applications from CPU-exhaustion Denial-of-service Attacks," in *Proc. USENIX Security*, Baltimore, MD, USA, Aug 2018.
[38] J. Mirkovic and P. Reiher, "D-WARD: A Source-End Defense Against Flooding Denial-of-Service Attacks," *IEEE Trans. Dependable Secur. Comput.*, vol. 2, no. 3, pp. 216–232, Jul. 2005.
[39] NoviFlow, "NoviSwitch^{TM} 2122 High Performance OpenFlow Switch," https://noviflow.com/wp-content/uploads/NoviSwitch-2122-Datasheet-1.pdf, 2018, accessed: 2018-28-1.
[40] Y. M. P. Pa *et al.*, "IoTPOT: Analysing the Rise of IoT Compromises," in *Proc. USENIX WOOT*, Washington, D.C., USA, Aug 2015.
[41] Palo AIto Networks, "PA-3000 Series Datasheet," https://bit.ly/2MPcsk2, 2018, accessed: 2018-28-1.
[42] Palo Alto Networks, "DoS and Zone Protection Best Practices," https://bit.ly/2HQOMwU, 2018, accessed: 2018-28-1.
[43] J. J. Pfeiffer III *et al.*, "Attributed Graph Models: Modeling Network Structure with Correlated Attributes," in *Proc. ACM WWW*, Seoul, Korea, Apr 2014.
[44] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," in *Proc. NDSS*, San Diego, CA, USA, Feb 2014.
[45] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI : Feature-based Automated NXDomain Classification and Intelligence," in *Proc. USENIX Security*, Baltimore, MD, USA, Aug 2018.
[46] V. Sekar *et al.*, "LADS: Large-scale Automated DDOS Detection System," in *Proc. USENIX ATC*, Boston, MA, USA, May 2006.
[47] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *Proc. IEEE S&P*, Oakland, California, USA, May 2010.
[48] Sophos Group, "Sophos XG Firewall: How to prevent DoS and DDoS attacks," https://bit.ly/2tcOPZY, 2018, accessed: 2018-11-2.
[49] Symantec, "The continued rise of DDoS attacks," https://bit.ly/2UFeWqK, 2019, accessed: 2019-4-2.
[50] Symantec Corporation, "IP Reputation Investigation," https://ipremoval.sms.symantec.com/, 2018, accessed: 2018-28-1.
[51] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study," in *Proc. ACM IMC*, Vancouver, BC, Canada, Nov 2014.
[52] A. Wang, A. Mohaisen, W. Chang, and S. Chen, "Delving into Internet DDoS Attacks by Botnets: Characterization and Analysis," in *Proc. IEEE/IFIP DSN*, Rio de Janeiro, Brazil, Jun 2015.
[53] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.
[54] T. Yu *et al.*, "PSI: Precise Security Instrumentation for Enterprise Networks," in *Proc. NDSS*, San Diego, CA, USA, Feb 2017.
[55] S. T. Zargar *et al.*, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, Mar 2013.
[56] B. Zdrnja, N. Brownlee, and D. Wessels, "Passive Monitoring of DNS Anomalies," in *Proc. DIMVA*, Lucerne, Switzerland, Jul 2007.
[57] M. Zhang *et al.*, "Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches," in *Proc. NDSS*, San Diego, CA, USA, Feb 2020.
[58] M. Zhang, G. Li, L. Xu, J. Bi, G. Gu, and J. Bai, "Control Plane Reflection Attacks in SDNs: New Attacks and Countermeasures," in *Proc. Springer RAID*, Heraklion, Crete, Greece, Sep 2018.

Minzhao Lyu received his B.Eng. degree (First Class Hons.) from the University of New South Wales, Sydney, Australia in 2017. He is currently pursuing Ph.D degree in the area of computer networks from the University of New South Wales and CSIRO's Data61. His research interests include programmable networks, network security and applied machine learning.

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 the University of New South Wales (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.