Classifying and Tracking Enterprise Assets via Dual-Grained Network Behavioral Analysis

Minzhao Lyu, Hassan Habibi Gharakheili, and Vijay Sivaraman

***Abstract*—Enterprise networks continue to grow in scale and complexity, encompassing a wide range of Internet-connected end-points including web servers/proxies, DNS/VPN/mail servers, and other special-purpose devices. Monitoring this dynamically evolving set of assets, for the purposes of ensuring operational efficiency and cyber security, poses a significant challenge for IT personnel. In this paper, we develop a system that automatically classifies enterprise Internet-connected assets in a continuous manner by analyzing their network activity, thereby reducing blind spots for organizational IT departments. Our contributions are three-fold: (1) We analyze over 3 billion packets from a large enterprise network to deduce network behavioral profiles of the popular asset types like website servers, DNS servers, and file storage systems and transport-layer patterns of less popular ones such as non-typical TCP/UDP servers, proxies, and NAT gateways; (2) We systematically develop host-level graph structure, identify a rich set of behavioral attributes, balance the computational cost against predictive power, train classifiers in a dual-grained classification scheme to categorize assets, and evaluate them via cross-fold validation as well as open set; and (3) We prototype our system on multiple 10Gbps Internet links of a campus network, and present insights over a month, such as the ability to identify hundreds of typical servers as well as thousands of non-typical assets, track their utilization, and highlight anomalous behaviors pertinent to possible cyber-threats. Our solution provides a dynamic and scalable way for IT personnel to effectively track enterprise assets.**

***Index Terms*—Host classification, network traffic analysis, machine learning, programmable networks.**

I. Introduction

Enterprise networks host a variety of Internet-connected devices ranging from website servers, web proxies, DNS proxies, mail servers, and VPN servers to remote computing platforms and desktops. Organizational IT departments struggle to keep track of their complex environment [6], [33], which continuously evolves as assets get decommissioned, and new ones are added. Consequently, it is common for organizations to be unaware of under-utilized and orphaned assets contributing to operational inefficiencies, as well as diverse device-specific vulnerabilities exposing the organization to cyber threats, as reported by AT&T Cybersecurity [11].

Industry guidelines for IT security and management recommend best practices to enforce a standard operating environment (SOE), such as controlled naming and address assignment, strict access control rules, and administrative permissions to onboard devices. However, first-hand experience has shown that the problem is particularly acute in less controlled enterprise networks such as Universities, wherein connected assets are managed in a loosely federated manner across departments, bespoke dynamic and complex network requirements of sub-departments limit the ability to enforce a strict SOE. Besides, the culture is attuned to staff and students connecting their own devices into the campus network. These blind spots have handicapped IT departments and exposed Universities to cyber-risks ranging from malware and botnets to DDoS attacks [30].

Large organizations often invest in a number of solid security tools, but they still report visibility gaps [14]. Maintaining an up-to-date inventory of Internet-connected assets[1] is very challenging in dynamic enterprise environments. Manually updated spreadsheets become obsolete quickly and are rarely synchronized across IT team members, let alone across the organization. For example, Facility Management in our University deploys security cameras and smart monitors, and IT only finds out later. Network managers have built home-grown tools to track assets by ingesting logs from various network services like DHCP servers, RADIUS authentication servers, DNS servers, firewalls, and web proxies; however, each of these only provides a narrow and independent view of the connected assets [9], [32], leaving many hosts undiscovered. This can be very problematic, given that network operators largely rely on device-specific rules (*e.g.,* access control lists) on their border security appliance (*e.g.,* firewall) to protect their assets, and having blind spots in their inventory can result in exposure to a large attack surface [5], [18], [33].

Prior research works on host network behavioral monitoring have relied on deducing end-to-end traffic graph patterns [46], [23], [49], [51], [19]. Statistical techniques such as machine learning (ML) are used to cluster/identify host types by correlations of networked entities [52], [12], [19], [21], [45]. With enterprise networked devices numbering tens of thousands and traffic rates growing to tens of Gbps, constructing a reliable graph structure of all network flows between internal hosts and external services can be expensive computationally, thus, impractical to be directly applied to a large network with high traffic rates.

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

[1]This paper focuses on those end-devices/hosts (each identified by their IP address) that communicate with remote endpoints on the Internet. Hence, our proposed network monitoring system is to be deployed at the edge of enterprise networks. There may exist certain hosts which merely communicate locally, and their traffic does not cross the edge of enterprise networks, and thus, are beyond the scope of this work.

In this paper, we propose a **dual-grained classification** method that leverages machine learning algorithms to classify and track the behavior of Internet-connected enterprise assets by passively analyzing their network traffic in real-time. Specifically, by analyzing the transport- and network-layer behavior of all enterprise hosts, our method: (1) classifies assets into fine-grained (specific) and/or coarse-grained (generic) types. A fine-grained class of an asset (host) is determined if it exhibits network patterns as one of ***N* popular types** (*e.g.,* web server, DNS server) specifically defined by the enterprise operator. Otherwise (*e.g.,* server operating on non-typical services), it is classified as one of **six coarse-grained types** that are generic for all enterprise networks; and (2) conducts a deeper investigation of only a small dynamic subset of hosts exhibiting unfamiliar behavior at both granularities to identify their non-standard activities (*e.g.,* generating attacks). Our approach has the advantage of being encryption-resistant, scalable, and easy to deploy without putting network operations at risk. We make three specific contributions elaborated below:

First, in §III, by analyzing a large representative traffic trace of over 3 billion packets collected from a University campus network, we summarize the typical network behavior of the ten most common host types in an enterprise, such as web server, mail server, and DNS server. We identify the presence of non-typical hosts such as servers that use less common protocols or non-standard services (ports), and hosts that serve multiple purposes. We further categorize enterprise hosts into six coarse-grained types based on their distinct behaviors at transport and network layers, including: TCP- and UDP-dominant public-facing servers that provide enterprise content to external users; TCP- and UDP-dominant proxies that act as relays (*e.g.,* for DNS and HTTPS); NAT gateways that represent internal clients with private IP addresses; and end-hosts that have unique public IP addresses.

Second, in §IV, we develop a dual-grained classification scheme that employs a rich set of host-level attributes along with supervised machine learning (ML) models to infer dynamic and diverse behaviors of enterprise hosts that may not be necessarily feasible with a deterministic and relatively static set of attributes. To this end, we first systematically profile host network behavior using a host-specific rooted-graph structure and identify descriptive attributes of network behaviors. We then optimize our data structures and behavioral attributes by balancing their predictive power against computational cost. We lastly develop ML models by tuning various algorithms, model parameters, input attribute sets, and retention periods. Our fine-grained model classifies enterprise hosts into N common specific types ($N = 10$ in our use-case), and the coarse-grained model classifies hosts into six generic types labeled by their dominant services, highlighting their functionality. Well-tuned models yield a fairly high accuracy (close to 99%) in cross-fold validation while providing cost-effectiveness (scalable and practical).

Third, in §V, we prototype our system using a commodity programmable switch and virtual network functions (VNF) on a generic server. We deploy it at the edge of our University network and highlight insights obtained over a one-month trial period. Our system was able to uncover over 300 web servers, along with several DNS servers, mail servers, NAT gateways, and proxies. Additionally, our system was able to detect unexpected behavior from several assets indicative of scans and malware. We also profile the performance of our solution in terms of CPU and memory usage, responsiveness, and inspection load, validating that it can easily scale to large enterprises.

The rest of this paper is organized as follows: prior relevant works and our key novelties are summarized in §II. §III describes insights from our traffic analysis of over 3 billion packets captured from the Internet border router of our university campus network. In §IV, we describe our dual-grained scheme for enterprise host classification, attribute extraction, ML model training, and evaluation. Our prototype design and campus deployment are described in §V. Limitations and future works are discussion in §VI. The paper is concluded in §VII.

II. Related Work

In this section, we discuss prior works and highlight the key novelty of our work.

A. Analysis of Network Traffic

Analysis of network traffic has been a hot topic for more than two decades. Port-based analysis methods [10], [22] that map traffic type by source/destination transport-layer port numbers are widely used in many commercial solutions, for its low computational cost and high accuracy in identifying certain application types and servers like website (HTTP/HTTPS) and name resolutions (DNS). However, with the complexity of modern applications and host roles in using a variety of transport services [3], purely port-based approaches fall short in effectively classify all hosts in an enterprise network. Analysis of network traffic using statistical methods such as machine learning is increasingly gaining interest from the research community [8], [40], [4]. Several works have been done in detecting and classifying various types of traffic such as video streaming [34], coflows from cloud computing [54], DNS assets [31], IoT devices [45], and email service [26]. In addition, a recent work [35] employed explainable AI techniques to analyze mobile traffic which becomes popular in this field.

B. ML-based Classification of Host Behaviors

There are many research works on classifying host types [25], [1], [19], [46], [23], [21], [17], [49] Authors of [25] classify user traffic flows such as bulk download, video, web, and interactive by characteristics of their network activities. Work in [1] classifies encrypted traffic of mobile applications. Work in [19] constructed networked graphs that represent interconnections between hosts; *G. Tan et al.* [46] proposed two effective algorithms in obtaining the similarity between host communication patterns to identify their social groups; and *BLINC* [23] classified host types using flow statistics to summarize host roles including attackers and victims in

cyber-crimes. Authors of [21] utilized a stochastic block model to identify pattern changes in the networked graph of host connections for potential anomalous changes. *Baywatch* [17] identified malware-infected hosts in an enterprise network by analyzing their beaconing behavior (the process for infected hosts communicating with remote Command-and-Control servers). *Beehive* [49] performed large-scale analysis on logs collected from key IT infrastructures such as DHCP servers, VPN gateways, and web proxies to detect suspicious host activities.

C. Systems with Programmable Networking

Programmable networking techniques (*i.e.,* SDN and NFV) have been employed to address various research problems such as dynamic telemetry and security enforcement [38]. Their use-cases range from measuring network-wide flow-level statistics elastically [48], [28], satisfying operators' dynamic needs for network telemetry via general-purpose query-driven system [15], identifying and fingerprinting specific network traffic like video streaming [34], [16] to detecting network attacks and threats reactively [13], [50], [27]. Inspired by prior works, to achieve scalable real-time monitoring along with reactive traffic isolation for less confidently classified enterprise hosts, we develop our system using virtual network functions on a generic server to process live traffic of an enterprise network (*i.e.,* up to 20Gbps), and leverage a commodity programmable switch supported by controllers and applications to reactively mirror traffic of selected hosts to an off-the-shelf packet inspection engine for further diagnosis.

D. Our Key Novelty

The novelty of our work can be summarized in three key aspects. **First**, we profile the network behavior of hosts by a comprehensive set of descriptive attributes computed from a rooted graph structure. They collectively capture packet and flow statistics associated with internal/external active hosts (IP addresses) and services (transport-layer port numbers). Machine learning algorithms can readily use them for accurate classification. **Second**, we systematically select cost-effective yet predictive attributes and reduce the graph complexity to achieve effective and scalable ML-based classification when analyzing network traffic of a large-sized enterprise. **Third**, we develop a robust data-driven method empowered by machine learning algorithms that can map all possible asset profiles (known and emerging). Given the dynamic nature of network traffic, deterministic approaches fall short of expectations in accurately modeling the behavior of diverse asset classes. Our dual-grained classification scheme offers an extensible model for inferring known specific host types (N fine-grained classes) as well as a generic model for predicting the behavior of emerging asset classes (six generic and coarse-grained classes). Our method empowers network operators to map all connected assets without leaving blind spots.

III. Understanding Network Behaviors of Enterprise Hosts

By analyzing one-hour traffic trace captured from two (one inbound and one outbound) 10 Gbps Internet border links, in this section, we: (a) highlight traffic statistics at the border of enterprise network in terms of distribution of inbound/outbound TCP/UDP traffic across the entire IP block of our organization and the variety of network services[2] (by transport-layer protocols and port numbers) on which internal hosts offer/access; (b) summarize network behavioral patterns of ten types of common enterprise networked assets (hosts) including website server, authoritative name server, VPN server, remote computing server, file storage server, email server, website proxy, recursive domain name resolver, and NAT gateway. In addition to these common types, there exist some non-typical assets with a diverse usage of custom transport-layer services. Therefore, we: (c) identify six aggregate classes of enterprise hosts based on their behavior regardless of the usage of transport services. These classes include TCP and UDP public-facing servers, TCP and UDP application proxies, NAT proxies, and end-hosts. Insights into the behavioral profile of various asset classes obtained in this section motivate our ML-based method (§IV) that precisely and automatically captures a comprehensive set of statistical (instead of deterministic) patterns found in our labeled dataset.

A. Overview of our PCAP Traces

To understand the behavioral profile of various enterprise hosts, we collected one-hour full traffic trace (inbound and outbound) from the two 10 Gbps Internet links outside the border firewall of our university campus network. Appropriate ethics clearances[3] were obtained for this study.

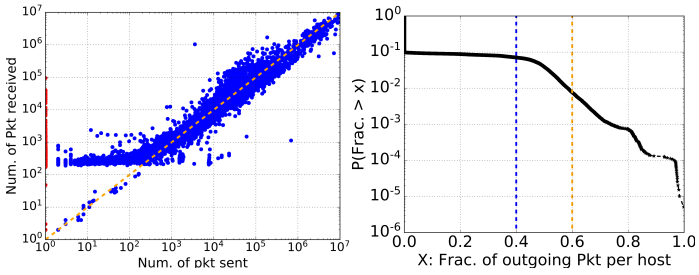
1) Basic Statistics of our Dataset: Our university owns three IPv4 blocks of size /16, giving a total of more than 196K public addresses. Using `tcpdump` tool, the first 96 bytes of all packets were recorded during the peak hour of a typical weekday (9-10am on 11 March 2019) – we verified that only 0.1% of packets were missed during this measurement. All headers of Ethernet, network, and transport layers are well-preserved, resulting in a total of 1.1 billion packets inbound and 1.6 billion packets outbound. The traffic rate of our collected data was about 10 Gigabits-per-second (Gbps) with an average 800K packets per second (pps).

2) Outgoing versus Incoming Traffic for Enterprise Hosts: Focusing on the hosts inside the enterprise network, as illustrated in Fig. 1(a), we observe that some hosts have almost equal amount of inbound and outbound packets (distributed across line y=x), whereas many other internal hosts display unbalanced behaviors. Also, there are "inactive" IP addresses (within the enterprise IP space) which receive packets from the Internet without sending any reply packet. For illustration purpose, we overlay inactive addresses by red dots along the y-axis in Fig. 1(a).

Fig. 1(b) shows the CCDF plot of the outgoing fraction of total packets per each internal host. In this plot, we can see three main regions: outgoing fraction (*i.e.,* x-axis) less than 0.4, between 0.4 and 0.6, and more than 0.6, partitioned

[2]We also use the term "transport service" to represent the combination of transport-layer protocol and port number, such as *TCP/443*.

[3]UNSW Human Research Ethics Advisory Panel approval number HC17499.



(a) in versus out pkts per host. (b) out fraction of pkts per host.

Fig. 1: Dynamics of bidirectional traffic volume across internal hosts of the enterprise network: (a) count of incoming versus outgoing packets per host, and (b) CCDF of outgoing fraction of packets per host.

(a) TCP port numbers. (b) UDP port numbers.

Fig. 2: Wordcloud of: (a) TCP, and (b) UDP, source port numbers in outgoing traffic of the enterprise network to the Internet.

by vertical dashed lines (blue and orange). The first region, accounting for about 90% of internal IP addresses, represents internal assets which are either completely inactive (zero outgoing packet), or have much less number of outgoing packets than incoming – they seem to be target of scans or DoS attacks. The second region (*i.e.,* $0.4 \leq outFrac \leq 0.6$) represents the majority of active internal hosts which have almost same amount of incoming and outgoing packets – such behavior is often normal and expected. Lastly, the third region represents those enterprise hosts, each having a majority of their packets leave the network (*i.e., outFrac* > 0.6) – seemingly participating in malicious activities (scans or DoS) which target external networks/hosts.

Note that a total of 217,708 internal IP addresses appeared in our dataset, where only 21,258 of these addresses are "active" hosts (*i.e.,* sending at least a packet to external entities) – most internal IP addresses did not respond to incoming packets from external entities (possibly subject to incoming scans from the Internet. For the rest of our analysis in this paper, we produce a "cleaned dataset" by only including packets of active hosts. Therefore, our study will be on those 21,258 enterprise hosts (IP addresses) with non-zero outgoing packets.

3) Diversity in Transport-Layer Behavior: We now analyze the headers of outgoing packets sent by internal hosts, and zoom into their transport-layer services and source port numbers. This enables us to identify the role (*i.e.,* client or server?) of individual enterprise hosts at a high level, and infer the type of services (*e.g.,* HTTPS, SSH, or QUIC) they may offer. Similar insights can also be obtained from incoming packets and their destination port numbers – we omit this analysis to avoid redundant explanations.

To better visualize the diversity of transport-layer services, we use their word-cloud representation in Fig. 2. A weight is associated to each internal service (TCP/UDP port numbers) using the count of packets sent per service port number. It can be seen that most frequently used port numbers are either occupied by well-known services (ranging from 0 to 1023 [47] like *TCP/443* for HTTPS, assigned by the Internet standard RFCs) or certain de-facto servers (*e.g., UDP/443* used for Cisco VPN application), or randomly selected client ports (*e.g., UDP/44247* and *TCP/56392*) for a variety of network applications.

Let us have a closer look at some of these transport-layer ports which dominate our traffic trace. Considering *TCP/443*, highlighted by dark green in Fig. 2(a), we observe that about 500 enterprise hosts serve close to 20000 unique external hosts (IP addresses) by this top most popular web service. We manually verified (by performing reverse DNS lookup within the enterprise network) that 382 of these enterprise hosts are public-facing servers (such as the main student web-portal of university, VPN server, or web servers of various faculties/schools). Interestingly, *UDP/443*, highlighted by yellow in Fig.2(b), is also among popular UDP services provided by enterprise hosts to the public Internet – this port number corresponds to Cisco AnyConnect VPN service, handling more than 200 remote users during the one-hour traffic capture. Note that QUIC (developed by Google and predominantly used by Google servers) also operates on *UDP/443* – our university network, however, does not have any QUIC servers in operation.

Port numbers greater than 1023 are typically used on the client side (randomly chosen by operating systems) when attempting to contact their TCP/UDP servers. Interestingly, we found that *TCP/5375*, highlighted by dark purple in Fig.2(a), is the source port in about 9 million packets – almost all of these packets are sourced from an internal NAT gateway, having replied by approximately equal number of packets, which is an expected behavior. We examined the headers of packets sent/received by this NAT gateway, found that the vast majority (98.6%) of the packets sent are 60-byte TCP ACK packets in response to packets (average size of 117 bytes) received from an HTTPS (*i.e., TCP/443*) server operated by Microsoft Azure Could Computing Platform – probably a large download requested by an enterprise host behind the NAT gateway.

Similarly, *UDP/44247* is highlighted by dark blue in Fig.2(b) – 7 million packets generating a total volume of $\approx$10 GB to a Google cache server (operating on QUIC via *UDP/443*). This outgoing traffic was sourced from an internal host while its Google server consistently replied by small packets (of size 85 bytes on average) during this interaction – most likely, automatic sync with Google drive or uploading a video onto YouTube.

As discussed above, the usage of network services (transport-layer port numbers) can be quite diverse, suggesting many types of roles (functionalities) across enterprise hosts. We, next, focus on popular host types that are commonly found in large typical enterprise networks.

TABLE I: Ten popular host types identified from DNS names and their abstract network behavior.

Type	# hosts	sample DNS name	# internal services	# external services	flow duration	pkt. size
Website srv	61	`www.unswlawjournal.unsw.edu.au`	small, fixed	large, random	short	medium
Authoritative name srv	15	`ns1.sdn.unsw.edu.au`	small, fixed	large, random	short	small
VPN srv	13	`securevpn.nida.edu.au`	small, fixed	large, random	long	medium
Remote computing srv	16	`analyticalcentre2.chem.unsw.edu.au`	medium, fixed	large, random	long	small
File storage srv	14	`files.be.unsw.edu.au`	small, fixed	large, random	medium	large
Mail srv	18	`smtp.garvan.unsw.edu.au`	medium, fixed	large, random	short	medium
DNS proxy	7	`ns6.unsw.edu.au`	large, random	small, fixed	short	small
Web proxy	4	`wwwproxy2.library.unsw.edu.au`	large, random	small, fixed	short	medium
NAT gateway	256	`uniwide-pat-pool-a-b-c-d.gw.unsw.edu.au`	large,random	large,random	medium	medium
End-host	1961	`minzhaos-macbook-pro.ad.unsw.edu.au`	medium, random	small, random	medium	medium

B. Fine-grained Behavioral Profile of Enterprise Hosts

There are diverse types of hosts in an enterprise network that are trivial and not practical to enumerate. We now discuss ten popular fine-grained host types which are quite common in a large enterprise network, highlight their typical network behaviors using a rooted graph, and illustrate the existence of non-typical host types and behaviors.

1) Ten Popular Fine-grained Types of Enterprise Hosts: **Website server** is one of the most commonly used asset types that can be found in enterprise networks. These networked assets serve contents to public users via HTTP (*TCP/80*) or HTTPS (*TCP/443*). To retrieve enterprise web contents, external users initiate short TCP connections to these servers, sourced from randomly selected transport-layer services (*i.e.,* port numbers).

The second essential network asset for most of large enterprise networks is an **authoritative name server** which maps the organizational domain names to their respective IP addresses configured by the network administrator. This type of servers often operate on *UDP/53*, answering DNS queries from many external entities which use random source port numbers.

To enable their employees and users (*e.g.,* staff and students in case of university networks) who need to remotely access protected IT resources, enterprises often set up **VPN servers** (virtual private network servers). These servers provide tunneling connections (usually with longer duration) between remote endpoints and their internal networked resources.

Remote computing servers such as workstations and virtual machines, that provide enterprise staff with powerful resources to execute computationally-intensive tasks, are commonly used by research groups and departments. These networked assets typically offer remote accessing services like SSH (*TCP/22*) and Telnet (*TCP/23*).

Enterprises often need to store and manage their business-critical information, and make it available to their trusted departments and individual employees. **File storage servers** are therefore configured centrally and/or by sub-departments, so that staff and trusted entities can upload or access such critical data via bulk-transfer protocols like FTP (*TCP/21*).

Large organizations like an university may host their own email domains. **Mail servers** which are authoritative for email domains and handle the delivery of emails, are another common host type in an enterprise. They obviously operate on Email-related protocols such as SMTP (*TCP/25*, *TCP/465*, *TCP/587*) and others.

To secure and facilitate DNS lookups from enterprise hosts to public resolvers, large enterprises often have central and/or department-level **DNS proxies** configured, that only send DNS queries (sourced from random UDP ports) to external resolvers that listening on *UDP/53*.

Similarly, **website proxies** may also get configured to perform web lookups on behalf of enterprise regular end-hosts. These proxies use random source TCP port numbers to retrieve contents using short connections from Internet-based website servers that offer HTTP (*TCP/80*) or HTTPS (*TCP/443*) services.

In addition to application-specific proxies, **NAT gateways** [29] are usually configured as agents to provide outbound Internet connectivity (TCP/UDP) for hosts (often WiFi-connected devices) without public IP addresses, protecting them from unsolicited incoming connections from the Internet.

Lastly, some enterprises may allocate certain **end-hosts** with their organizational public IP addresses, allowing them to directly communicate with the Internet. In our university network, machines connecting via Ethernet cable to wall ports in staff offices and certain labs will get public IP addresses.

In addition to these popular asset types (discussed above), we note that some other hosts like SNMP agents, video conferencing hubs, and Key management servers are not necessarily common in every enterprise network, hence not explicitly studied in this section. We will later in this section (§III-C) analyze the behavior of enterprise hosts by broadly considering their network activity over transport-layer services.

2) Inferring Host Types from Their DNS Name: DNS names associated with enterprise hosts can be helpful to some extent for inferencing their roles. In our 1-hour PCAP dataset, we extracted the domain name of each enterprise host (IP address) from outgoing DNS responses captured on the same day[4]. We managed to obtain the DNS name for 11,039 out of the 21,258 active enterprise hosts – more than 50% are found with a corresponding DNS name. However, by analyzing their DNS name (a combination of automatic string search and manual inspection), one may identify the role (type) of only 2,365 hosts (11%) – only those servers and proxies that are directly managed by our university central IT department, public servers operated by certain

[4]We cross-checked against a separate dataset of daily DNS packets (incoming/outgoing), recorded at the border of our university campus network.

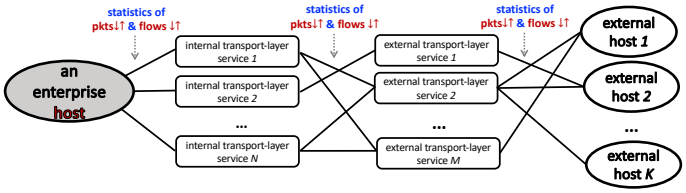


Fig. 3: Rooted graph structure containing four layers visualizes the network behavior of an enterprise host.

departments/divisions (*e.g.,* a journal website maintained by the Law School of our university, as shown in the top row of Table I), and NAT gateways or end-host devices that have names with identifiable patterns. For example, website servers usually use "`www`" in their name prefixes; some authoritative name servers in our university network have their names starting with "`ns`" followed by a number less than 5; the domain names of some DNS proxies contain "`ns`" followed by a number larger than 5; and NAT gateways configured by our IT department would have names with a certain pattern like "`uniwide-pat-pool-a-b-c-d.gw.unsw.edu.au`", given their static IP address is "`a.b.c.d`". The second and third columns of Table I summarize the count of hosts and a sample of their DNS names across the ten popular types.

We note that though DNS name may be used to label (identify the role of) some of the active hosts (mostly those which the central IT department manages), a large fraction of enterprise hosts (especially those which are managed by subdivisions or departments) do not have an identifiable name, hence remain unclassified. Therefore, a more comprehensive approach is required to profile the behavior of active hosts and identify their functionalities.

3) Profiling Network Behavior of Enterprise Hosts using Rooted Graph Structure: We profile the behavioral characteristics of each enterprise host by a rooted graph structure shown in Fig. 3, illustrating the communications between the enterprise host and external entities. As shown in this graph, the enterprise host (the leftmost node in Fig. 3) connects with external hosts (the rightmost nodes) via intermediate nodes (internal and external transport-layer services) and their corresponding edges. Edges on the graph represent statistical attributes (*i.e.,* count, size, and direction of packets; count, rate, volume, direction, and duration of flows) for unidirectional traffic exchanged across transport-layer services used by the enterprise host and corresponding external hosts. The direction of a flow is determined by the (internal or external) host which initiates the flow – for an "inbound" flow, the first packet is sent by an external host to the internal host. Thus, for every flow, we model inbound and outbound packets by separate edges.

We now generate and analyze the network behavioral profile of those popular host types (as ground-truth data) are identified by their DNS name. The last four columns in Table I briefly summarize the graph pattern (behavior) of each host type based on the features of nodes (*i.e.,* internal and external transport-layer services) and edges (*i.e.,* flow duration, and packet sizes).

Generally speaking, we observe both distinct and common behaviors across different host types. For example, servers like website server, authoritative name server, and remote computing server display a relatively focused set of internal transport services and fairly spread/broad set of random external transport services, while proxies (web and DNS) exhibit an opposite behavior – a wide range of random internal services and a narrow set of fixed external services). Furthermore, in terms of flow and packet characteristics, we observe some distinct patterns. For example, VPN servers typically maintain long flows with medium-size packets, while file storage servers generate medium-duration flows carrying large packets. To make our discussion more concrete, we next zoom into the behavioral profile of three representative host types, including website servers, web proxies, and NAT gateways.

Website servers: In summary, such servers are likely to offer a small set of internal TCP services to a wide range of external user TCP ports. The top internal services (by either packet or flow count) in both directions are likely to be *TCP/443* and *TCP/80*, while external port numbers are pretty random, distributed relatively evenly. Also, compared to other types of servers that predominantly operate over TCP services, website servers often maintain short flows with less than a few seconds.

Let us take a closer look at the behavior of an example website server (*i.e.,* student portal of our university), captured in our 1-hour PCAP trace. Starting with high-level observations on its packet-level characteristics, we found that the packet count is almost equal in both directions (*i.e.,* 917K for inbound and 913K for outbound). In contrast, the average size of outbound packets (245 Bytes) is more significant than that of inbound packets (62 Bytes). In terms of flow-level characteristics, the server received far more flows from the Internet (18K inbound flows) than it initiated towards external hosts (1K outbound flows). Also, the average duration of inbound flows is larger than that of outbound flows (*i.e.,* 1.4s and 0.2s, respectively). The server is found to use a total of 458 internal services (*i.e.,* ports), while about half of them (209 transport services) only appear in the inbound packets. Apparently, it is the victim of unsolicited traffic from the Internet. We further investigated those remaining 247 services (excluding unsolicited ones), and ranked them by their contribution to the number of packets and flows in each direction. Table II shows the top five services of the website server by four traffic characteristics (inbound/outbound packets and flows). The number in square brackets highlights the contribution of the corresponding service. Blue cells indicate services that are common among the top-5 across the four columns, while red cells indicate uncommon services which are not necessarily among the top-5 across all columns. It is clearly seen that a vast majority of packets and flows in both directions are contributed by a small set of services that collectively characterize the behavior of this website server. We also analyzed external transport-layer services communicated with this asset[5]. We observed a wide range (more than 12K) of TCP services utilized fairly evenly – none of the top-5 external services contributed more than 0.5% across the four metrics of inbound/outbound packets and flows. Further, by analyzing the

[5]We omitted the table of results for external transport services.

TABLE II: Utilization of top-5 internal transport-layer services of a representative website server.

In↓ packets	Out↑ packets	In↓ flows	Out↑ flows
TCP/443 [97.2%]	*TCP/443* [98.1%]	*TCP/443* [73.0%]	*TCP/443* [52.0%]
TCP/80 [2.7%]	*TCP/80* [1.8%]	*TCP/80* [24.9%]	*TCP/80* [24.3%]
TCP/0 [0.0%]	*TCP/50923* [0.0%]	*TCP/50923* [0.9%]	*TCP/50923* [0.2%]
TCP/23 [0.0%]	*TCP/21631* [0.0%]	*TCP/21631* [0.0%]	*TCP/21631* [0.2%]
TCP/137 [0.0%]	*TCP/8641* [0.0%]	*TCP/8641* [0.0%]	*TCP/8641* [0.2%]

TABLE III: Utilization of top-5 internal transport-layer services of a representative web proxy.

In↓ packets	Out↑ packets	In↓ flows	Out↑ flows
TCP/39762 [3.6%]	*TCP/33280* [3.5%]	*TCP/3128* [0.2%]	*TCP/42108* [0.4%]
TCP/40576 [3.5%]	*TCP/41801* [2.6%]	*TCP/51771* [0.1%]	*TCP/51769* [0.4%]
TCP/58355 [3.4%]	*TCP/8513* [2.6%]	*TCP/2869* [0.1%]	*TCP/15833* [0.3%]
TCP/44379 [3.3%]	*TCP/40576* [2.4%]	*TCP/7253* [0.1%]	*TCP/3985* [0.3%]
TCP/53718 [3.2%]	*TCP/39762* [2.4%]	*TCP/48782* [0.1%]	*TCP/48471* [0.3%]

behavior of other website servers, we observed very similar patterns with slight variations in their use of transport-layer services *TCP/80* and *TCP/443*. Some use a mix of HTTP and HTTPS. In contrast, others prefer one of these TCP services.

Additionally, we note that website servers share certain network behaviors with public-serving assets like authoritative name servers, VPN servers, remote computing servers, and file storage servers. They all have a small set of internal services communicated with a wide range of external transport services. That said, each of these individual asset types is differentiated by their unique transport-layer services (*e.g., UDP/53* for authoritative name servers, or *TCP/22*, *TCP/23* or *TCP/3389* for remote computing servers). Also, they exhibit different characteristics of packets and flows (*e.g.,* VPN servers often maintain longer flows with an average duration of about a minute, and file storage servers typically use larger packets of average size over 350 Bytes).

Web proxies: These hosts use a wide range of random internal TCP ports, accessing *TCP/443* and *TCP/80* services offered by Internet servers.

As an example, let us highlight the traffic profile of a web proxy in our university network. At a high level, this proxy exchanged an almost equal number of packets in each direction (1.2M outbound and 1.5M inbound), while it had far more outbound flows (36K) than inbound flows (3K). Table III provides relatively fine-grained insights into the behavior of this type of host by listing its internal top-5 transport-layer services along with their respective contribution to the host traffic at packet and flow levels. We can see that all cells in this table are highlighted by red color, suggesting uncommon services across the four metrics. Also, it is seen that these top services are utilized fairly evenly by individual columns of in/out packets and flows. As for external transport services (*i.e.,* the services on external hosts), instead, a vast majority of packets and flows (*i.e.,* more than 99%) are narrowly focused on *TCP/443* and *TCP/80*. We saw earlier this set of dominant services on the internal side of the website server.

Other application proxies in our dataset displayed similar behavioral patterns, except their usage of transport services – *e.g.,* DNS proxies primarily use UDP for their transport-layer protocols, with *UDP/53* dominating their external service.

NAT gateways: Unlike website servers and proxies described so far, NAT gateways use a much more random set as well as a broader range of transport services, both internally and externally, across TCP and/or UDP protocols. Unsurprisingly, they behave very much like clients that initialize flows towards external hosts.

For example, a WiFi access point (NAT gateway) on our campus network exchanged almost the same number of packets in each direction (*i.e.,* 2.2M inbound and 2.3M outbound) as captured by our 1-hour traffic trace. In terms of flows, we observed about 50K outbound flows and 30K inbound flows. This large count of incoming flows is unexpected. Further investigation revealed that 28K (91%) of the inbound flows are unsolicited and not responded. The responded incoming flows (remaining 2K) found during the first five minutes of our trace – probably they were initiated (as outbound) just before the commencement of our traffic capture. Note that we will discuss in §IV that why our final classifiers will be trained by attributes of outbound traffic of individual hosts.

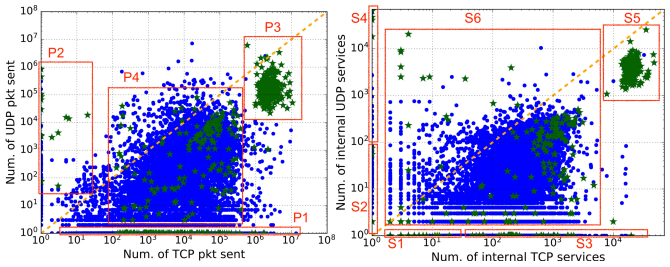
Analyzing the internal services of this network asset does not manifest any pattern (a wide range of seemingly random TCP/UDP port numbers). For external transport-layer services, instead, more than 99% of inbound/outbound packets and flows are contributed by top-5 services including HTTPS (*TCP/443*), QUIC (*UDP/443*), HTTP (*TCP/80*), DNS (*UDP/53*), and IMAPS (*TCP/993*), while strongly dominated (more than 90%) by HTTPS and HTTP.

Other NAT gateways are found to exhibit similar behavior with slight variation in their use of transport services. Note that end-hosts also share this behavior, but with lighter activity in traffic volumes and range of transport services.

4) Enterprise Hosts with Non-Standard Behavior: By further analysis of our PCAP dataset, we identified some other types of network assets that either fundamentally differ from typical hosts (Table I) or display significantly new patterns in addition to the expected behavior of typical hosts.

Non-typical host types: Some enterprise hosts utilize transport-layer services that are less popular in a typical enterprise IT infrastructure. For example, one host has two internal TCP services (*TCP/636* for LDAP and *TCP/389* for LDAPS) that together contribute to more than 97% of outbound/inbound packets and flows. Also, we found another server with a distinct internal service *TCP/11371* for HKP (used by Key management servers). We verified that this host is operated by a non-profit organization within our university campus by inspecting its DNS name.

Non-standard variants of typical host types: Among the hosts labeled (by their DNS name) as one of the ten typical types, some display distinct behaviors (by the use of additional services) compared to their respective cohort. A website server configured by a research group in an engineering department, has 5 distinct internal services, namely *TCP/443* and *TCP/80* (expected standard services), as well as *TCP/3306*, *TCP/2222*, and *TCP/23* that respectively correspond to MySQL, SFTP, and Telnet (non-standard services) – respectively, contributing to 28.1%, 24.6% 11.0%, 4.4%, and 3.6% of outbound packets. Indeed, running a server with multiple roles is not a best



(a) Packet sent. (b) Internal transport services.

Fig. 4: UDP versus TCP outgoing traffic per internal host: (a) count of packets sent, and (b) unique internal transport-layer service used.

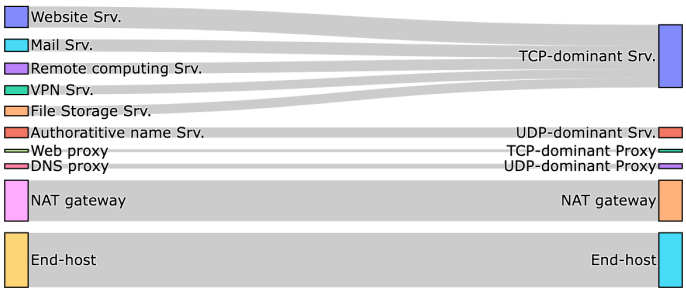


Fig. 5: Fine-grained classes (left) map to coarse-grained classes (right) for hosts with a ground-truth label from their DNS name.

practice [41], since various services demand specific policies (for their particular vulnerabilities and risks) to be enforced at the network. Another example is a name server managed by an engineering department (not the central IT) in our university. We found this host to function as both authoritative name server and DNS proxy – *UDP/53* is the most dominant in both internal and external services. In addition to a combined role, it is not a recommended security practice for an enterprise host to resolve DNS queries from the public Internet [2].

C. Coarse-grained Behavioral Profile of Enterprise Hosts at Transport-Layer

From what we have observed in our traffic traces, a growing set of profiles (classes) will exist for enterprise hosts, considering the specific transport-layer services ("fine-grained") they offer and/or consume – the role of many emerging hosts may not be known to enterprise network operators. This makes it practically challenging to capture and maintain those individual classes a priori for a real-time asset classification and monitoring task. However, it is possible to model host behaviors by aggregating their fine-grained transport services and considering their "coarse-grained" characteristics at the transport-layer. In what follows, we discuss how enterprise hosts can be categorized under six generic coarse-grained types, including TCP-dominant server, UDP-dominant server, TCP-dominant proxy, UDP-dominant proxy, NAT gateway, and end-host.

1) Six coarse-grained host types by transport services: Enterprise hosts that offer network services to users on the public Internet can be either a **TCP-dominant** or **UDP-dominant server**, depending on the distribution of transport-layer services (*i.e.,* TCP or UDP) in their network traffic. They expose a small set of internal transport-layer services (either TCP or UDP) contacted by a wide range of external transport services initiated by Internet clients. For instance, the website servers mentioned above primarily operate on HTTP (*TCP/80*) and/or HTTPS (*TCP/443*) and thus are TCP-dominant servers. An organizational VPN server can operate on both *TCP/443* and *UDP/443*. Still, a vast majority of packets and flows belong to TCP protocol, and thereby a TCP-dominant server.

Proxies that access certain services like Web (*TCP/443*) or DNS (*UDP/53*) on the Internet can be categorized as either **TCP-dominant proxy** or **UDP-dominant proxy**. They tend to use a highly diverse and random set of internal transport-layer services, either TCP or UDP. Also, proxies (depending upon their role) consume a narrow set of external transport services. For example, DNS proxies (as UDP-dominant proxies) use a wide range of internal UDP service numbers to send DNS query packets to external servers operating on the network service *UDP/53*.

Lastly, **NAT gateways** and **end-hosts** are relatively distinct by their use of a wide range of internal services, consuming a mix of external services over both TCP and UDP protocols.

2) Grouping Enterprise Hosts by Coarse-grained Behavior of Transport-layer: We now analyze the coarse-grained transport-layer behavior of all active enterprise hosts that appeared in our 1-hour cleaned dataset. We compute the number of packets, flows, internal transport services, and external transport services of both directions (TCP and UDP separately) for a given enterprise host. In what follows, we elaborate on the coarse-grained types (discussed above) by highlighting some of network behaviors for the hosts with ground-truth label.

Let us concentrate on the outgoing traffic of individual hosts and analyze their distribution of packets and services[6]. Fig. 4(a) is the scatter plot of UDP versus TCP packets per host. Fig. 4(b) displays UDP and TCP ports distributed across individual hosts. Each green star highlights a ground-truth host identified by their DNS names, and blue dots represent other hosts. Fig. 5 visualizes how fine-grained classes of hosts (those with a ground-truth label from their DNS name) map to coarse-grained classes. Note that the width of each link (connecting left boxes to right boxes) indicates their relative popularity instead of absolute quantity.

We observe that hosts (with ground-truth label) from various coarse-grained types display distinct behavior at least by the two distributions illustrated in Fig. 4. TCP-dominant servers appear on the narrow region (P1) in Fig. 4(a) and narrow (S1) in Fig. 4(b), indicating their heavy and concentrated activities via specific TCP services, that is, sending out many TCP packets through a limited set of TCP services. Similarly, UDP-dominant servers are expected to appear on regions (P2) in Fig. 4(a) and (S2) in Fig. 4(b). TCP-dominant proxies are located on (P1) and (S3) regions (large number of TCP packets, and heavily distributed in TCP services), while UDP-dominant

[6]We omit the insights obtained from inbound packets and flows for brevity.

proxies sit on the regions of (P2) and (S4) in the two scatter plots. NAT gateways, given they represent a large number of end-hosts, appear in the upper right corner of both figures (*i.e.,* (P3) and (S5)). Lastly, end-hosts mainly fall in two broader regions ((P4) and (S6)) for their diverse and less-concentrated use of TCP and UDP transport-layer services.

IV. Classifying Enterprise Hosts

In this section, we develop a dual-grained classification scheme that classifies enterprise hosts into ten popular fine-grained types (discussed in §III-B) and six aggregated coarse-grained types (discussed in §III-C) with highlights of their dominate services at transport-layer. We discuss host-specific traffic attributes used as inputs of our models, and quantify their importance, independence, and computational cost. Next, we train, tune, and validate two multi-class ML models (as our baseline models) for both fine-grained and coarse-grained classifications. We enhance the practicality of our method by judiciously selecting subsets of attributes and adjusting retention duration to enable it for real-time operation at scale. Their classification performance is compared with that of our baseline models. Lastly, we quantify the efficacy of our inference scheme by applying it to a fresh set of traffic instances not seen during the training phase of our ML classifiers.

A. Dual-Grained Classification Scheme

Ideally speaking, every enterprise host is expected to be labeled by their fine-grained type (*e.g.,* website server, authoritative name server, or web proxy). However, it becomes a challenging task in practice when many operational hosts may not display a narrowly identified behavior (using less-common network services or having mixed roles). Therefore, those hosts can at least be classified as one of the six coarse-grained types (*e.g.,* TCP-dominate server or TCP-dominate proxy) by aggregating transport-layer services.

Therefore, our classification scheme infers the type of enterprise hosts at two levels of granularity, namely fine-grained and coarse-grained. Fig. 6 illustrates the structure of our scheme where we take two groups of statistics computed from the rooted graph of each host (§III-B3) as inputs. The first group pertains to "numerical traffic attributes" that describe the activity behavior of the host without inclusion of specific transport-layer service name (*e.g.,* "*TCP/443*"), while the second group highlights "top transport-layer services" (internal and external) of the host. Our classification scheme contains three functional modules. The fine-grained model is a *N*-class classifier that receives both of the input groups, and predicts a specific class (*e.g.,* website server) with a confidence value for the input host. For the use-case of our campus network, we considered ten popular host types (*N=10*). We note that various enterprises may want to customize the number of fine-grained classes (extending our ten classes or choosing a subset of these ten classes) depending on their preference and the composition of their network. The other two modules are designed for coarse-grained inference. A six-class classifier model gives an intermediate prediction (*e.g.,* TCP-dominate server) with a confidence level by processing numerical traffic attributes

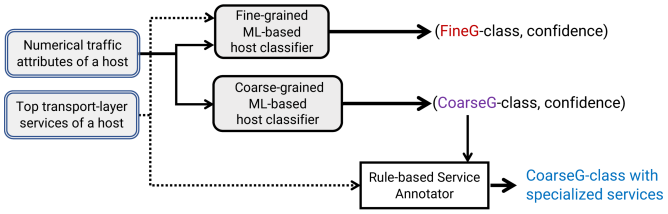


Fig. 6: Our dual-grained classification scheme.

of a host. A rule-based annotator will take the intermediate prediction as well as top transport-layer services of the host to generate the final coarse-grained host type (*e.g.,* "a TCP-dominate server by *TCP/443* and *TCP/80*"). The rule-based annotator uses our insights obtained in §III-C to annotate servers by their top internal services, proxies by their top external services. NAT gateways and end-hosts are not annotated. It is important to note that the coarse-grained inference model is generic, and hence applicable to any enterprise setting. When the prediction is annotated, it enables the network operator to discover emerging asset classes, potentially extending the fine-grained model by new classes (if desirable).

Each of the two predictions (fine-grained and coarse-grained) will contain a class label and confidence level, helping network operators better manage their assets by choosing certain inference results. A coarse-grained label is accompanied by top transport-layer services per host, providing additional information for further investigation post automatic inference.

Generalizability and Utility of our Classification Scheme: The data structure and host attributes (§IV-B), ML training methods (§IV-C), and system prototype (§V-A) are generic and applicable to any enterprise network. To apply the methods discussed in this paper to an enterprise, network operators will need to train the ML models on their own labeled traffic traces to capture behavioral nuances contextualized to their network. Also, they may want to customize (expand or shrink) the fine-grained model subject to their specific requirements.

B. Attributes of Host Network Behavior

We now discuss and evaluate our host-specific attributes required for the inputs of our dual-grained host behavior inference scheme.

1) Attributes: Having understood various host profiles in §III, we identify a set of attributes that are computed from the rooted graph of an enterprise host (shown in Fig. 3) to capture their comprehensive network behavior. Given nodes (enterprise host, internal transport-layer services, and external transport-layer services) in Fig. 3, we extract a total of 256 attributes, including 176 numerical traffic attributes and 80 categorical attributes indicating top transport-layer services. To better describe them, we choose the name of each attribute according to a pattern "*metricType-direction-resolution*", where "*metricType*" capture statistical measures of traffic volume and utilization of transport services, "*direction*" highlights inbound↓ vs. outbound↑, and "*resolution*" is packet-level or flow-level. In what follows, we discuss three groups of numerical attributes, namely aggregate host activity, utilization

of internal transport-layer services, and utilization of external transport-layer services), along with categorical attributes (*i.e.,* top transport-layer services).

Aggregate Host Activity: The leftmost node in Fig. 3 is the enterprise host with edges of inbound and outbound packets/flows representing its aggregate network activity. We use two metrics *AvgSize* and *VarSize* to highlight the average size and variance of the traffic units (packet-level or flow-level) for a given host.

Our analysis in §III revealed that various enterprise hosts could exhibit different traffic distribution in each direction (*i.e.,* inbound↓ or outbound↑) across resolutions (*i.e.,* packet or flow). Therefore, we compute the above two metrics for both directions and both resolutions, resulting in 8 attributes for this group. An example of these attributes is *AvgSize-↓-Pkt*, indicating the average size of inbound packets.

Utilization of Internal Transport Services: Moving to attributes of the second leftmost node in Fig. 3, we find various services via protocol type TCP or UDP, distributed across inbound and outbound directions. For this group of features, we identify 21 statistical metrics.

We start by two aggregate metrics, namely *DominTypeInSrv* (*i.e.,* the dominant protocol type of enterprise services by packet/flow count), and *FracMinorTypeInSrv* (*i.e.,* the fraction of packet/flow count associated with internal services of the protocol type at minority).

To capture the distribution of host traffic across internal services, we consider two ways, namely (i) packet/flow count, and (ii) unique external service count, for ranking their individual contributions. First, by considering the total count of packets/flows, we identify nine statistical metrics. Given a list of services ranked by their packet/flow count from largest to smallest, we compute: (a) traffic fraction of the top service (highest activity), the first quartile service, the second quartile service and the third quartile service, denoted by *FracTopInSrv*, *FracQ1InSrv*, *FracQ2InSrv*, and *FracQ3InSrv*, respectively; (b) the variance of traffic fraction across internal services denoted by *VarInSrv*; (c) fraction of internal services above average, above average plus one-sigma, above average plus two-sigma, and above average plus three-sigma, denoted by *FracAbvAvgInSrv*, *FracAbvAvg1SigInSrv*, *FracAbvAvg2SigInSrv*, and *FracAbvAvg3SigInSrv*, respectively. Second, by considering the count of corresponding unique external services, we identify ten statistical metrics. Nine of the metrics are computed in the same way as described above. Additionally, we use the ratio of internal service count and external service count, denoted by *RatioIntExtSrv*.

For this group, we identify a total of 21 metrics across two directions and two resolutions, resulting in 84 numerical attributes.

Utilization of External Transport Services: Similar to the characterization of internal services (discussed above), we capture the distribution of external services by 21 metrics (each with two directions and two resolutions), resulting in a total of 84 attributes. For brevity, we omit details of attributes.

Top Transport-layer Services: As discussed in §III-B, various host types may focus on certain transport-layer services, appeared as dominant internal and/or external services. We use the top five internal and external services ranked by certain properties of edges in the graph depicted in Fig. 3. Top-ranked internal services are determined by: (i) the volume of traffic (packet, flow) in each direction (outbound, inbound) on the left edges connecting them to the enterprise host, and (ii) the count of non-zero (inbound/outbound/ packet/flow) middle edges connecting them to the external services. Similarly, top-ranked external services are determined by: (i) the volume of traffic on the right edges connecting them to external hosts, and (ii) the count of non-zero middle edges connecting them to the internal services. For example, the attribute *TopInSrvVol-↑-Pkt* indicates the top internal transport service ranked by the volume of outbound packets, while *TopInSrvNZExSrv-↑-Pkt* is ranked by the count of external services with non-zero edges of outbound packets. As a result, 80 attributes are identified. We note that this group of attributes is categorical and can not be directly fed to numerical ML models. We, therefore, use a method called integer encoding that maps categorical attributes to numerical values, overcoming this challenge. This lightweight method is suitable for a wide range of categorical values [42] compared to its alternatives like one-hot encoding. Also, it can be handled well by tree-based classifiers.

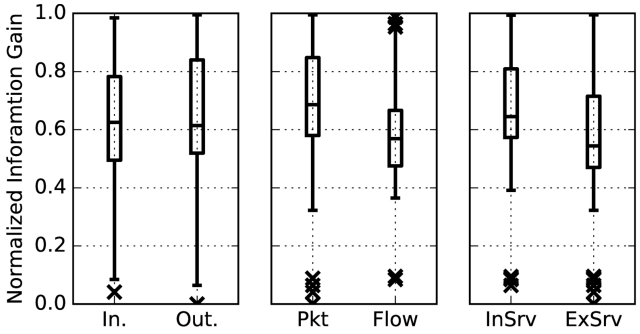


Fig. 7: The merit of attributes across different perspectives.

2) Dataset Preparation: We compute attributes of hosts with the ground-truth label (fine-grained and coarse-grained types) identified by DNS names in §III-B2. For each of those hosts, a run-time rooted graph is tracked with a retention duration of one hour (*i.e.,* edges that are inactive for more than one hour will get removed), and host attributes are calculated every minute. As a result, a dataset consisting of 928,946 records for 2,365 ground-truth hosts (identified by DNS names in §III-B2) is developed from 24 hours (between 11am on 31st May 2019 and 11am on 1st June 2019) worth of traffic traces.

3) Merit of Attributes: We now evaluate the merit of individual attributes (a total of 256 attributes: 8 for aggregate host activity, 84 for utilization of internal transport service, 84 for utilization of external transport service, and 80 for top transport-layer services) in predicting the type of their corresponding host. We also quantify the dependency between each pair of attributes and the cost for computing each attribute in real-time. The insights gained from this section will guide us (later in §IV-C3) to balance the cost against the accuracy of our ML classifiers.

Importance: To quantitatively justify the efficacy of our identified attributes, we use "information gain" to measure

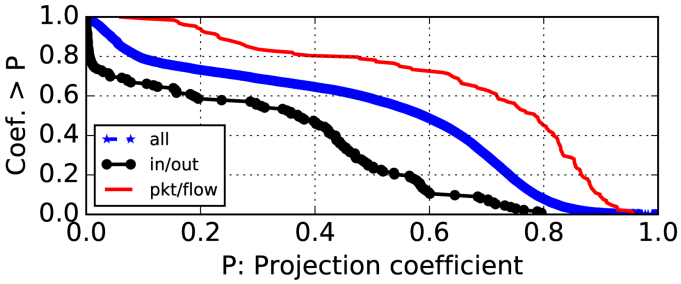


Fig. 8: Correlation of attribute pairs.

their importance. *InfoGain* of an attribute indicates how much information the attribute provides with respect to the classification goal. To be more specific, *InfoGain* [7] of attribute is derived as the difference between the entropy of original (unsorted) host type and the entropy of host type sorted by the attribute.

Normalized importance of each attribute is calculated and labeled by their direction (*i.e.,* inbound or outbound), resolution (*i.e.,* packets or flows), and associated node (*i.e.,* internal or external transport-layer services). Fig. 7 summarizes the importance of attributes across these three perspectives as box plots where each box highlights a range from the first quartile to the third quartile of merit values within their respective group. It can be seen that all attributes contain some information for predicting the type of hosts. Only a few attributes display a low importance (InfoGain values smaller than 0.2) – these attributes are primarily pertinent to the protocol type of transport layer (*e.g., DominTypeInSrv-↓-Pkt*). As indicated by the leftmost subplot in Fig. 7, outbound attributes are slightly more predictive than their inbound counterparts – possibly because inbound traffic may contain noises such as scans or unsolicited traffic, providing information not indicative of the role of internal hosts. Moving to the middle subplot, packet-level attributes (given their higher resolution) relatively outweigh flow-level ones. Lastly, the rightmost subplot shows that attributes pertinent to internal transport services yield a higher power in predicting the type of enterprise hosts than external services.

Independence: Certain attributes may positively or negatively correlate with others. To quantify the correlation between a pair of attributes, we use the projection coefficient p between the instances of two attribute arrays.

Fig. 8 shows the CCDF plot of the projection coefficient across all $\binom{256}{2} = 32640$ attribute pairs (shown by blue dots), 128 corresponding pairs in two directions like *FracTopInSrv-↓-Pkt* versus *FracTopInSrv-↑-Pkt* (shown by black dots), and 128 corresponding pairs in two resolutions like *FracTopInSrv-↓-Flow* versus *FracTopInSrv-↓-Pkt* (shown by red dots). We observe that packet-based attributes are largely correlated with their corresponding flow-based attributes – 70% of pairs on the red curve display a correlation value greater than 0.6. On the other hand, outbound attributes are loosely correlated with their inbound counterparts. Almost 80% of pairs on the black curve display a correlation value smaller than 0.5 – probably because inbound traffic is relatively polluted by unsolicited traffic.

Computational Cost: To compute the attributes of an enterprise host in real-time, we need to continuously maintain and update their data structure with run-time statistics. Therefore, we use the complexity of the data structure required to obtain an attribute as a proxy of its cost.

As illustrated by Fig. 3, a fine-grained graph contains four key layers including enterprise hosts, internal transport-layer services, external transport-layer services, and external hosts. Flow-based attributes (*e.g., AvgSize-↑-Flow*) require all four key layers of metadata (*i.e.,* enterprise host, internal transport-layer service, external transport-layer service, external hosts); hence, they are computationally heavier. Attributes like those that pertain to both internal and external transport services (*e.g., TopInSrvNZExSrv-↑-Pkt*) need three key layers of metadata (*i.e.,* enterprise host, internal transport-layer service, external transport-layer service). Lastly, packet-based attributes may only need one or two layers of metadata (*e.g.,* enterprise host for *AvgSize-↑-Pkt*; enterprise host and internal transport-layer service for *FracTopInSrv-↓-Pkt*); hence, they are computationally lighter.

We, therefore, associate qualitative costs of the low, medium, high, and ultra-high to attributes that require one, two, three, and four key layers of metadata, respectively. As a result, of the 256 attributes, 4 are low cost, 68 are medium cost, 56 are high cost, and 128 are ultra-high cost.

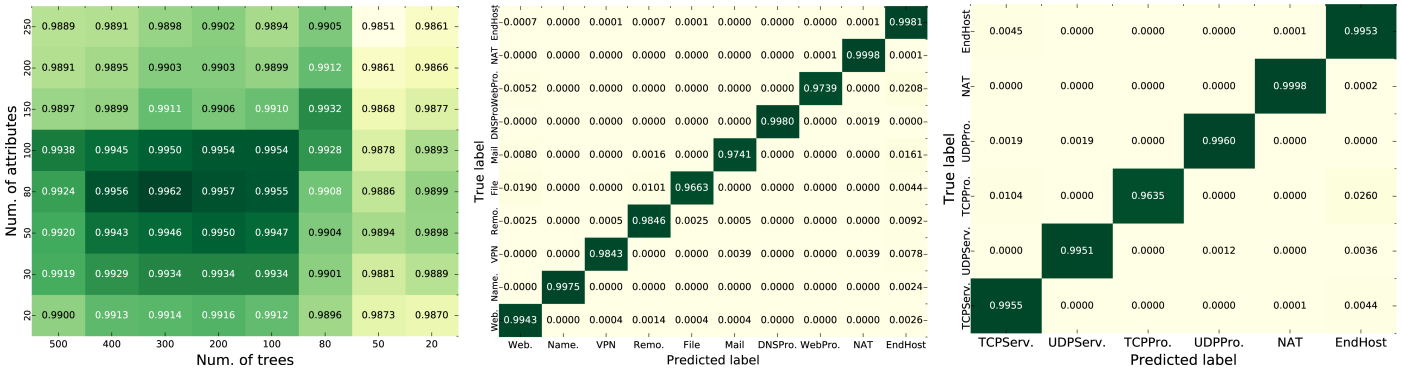
Maintaining the comprehensive four-layer graph structure (in Fig. 3) and compute all host attributes can be impractical for the high complexity of a large enterprise network. Therefore, one may focus on a subset of attributes computed from lightweight sub-graphs that give more predictive power, are independent, and incur a reasonable cost. For example, later in §IV-C3, we show 36 attributes of outbound packets that carry significant information and can be computed at low/medium cost from two optimized two-layer sub-graphs.

C. Training, Tuning, and Cross-Validating Classifiers

We train and evaluate our ML classifiers for both fine-grained and coarse-grained host types using two famous algorithms, namely multilayer perceptron (*i.e.,* MLP, a neural network algorithm) and Random Forest (a collection of decision trees). Our models are trained and cross-validated using our ground-truth dataset (discussed in §IV-B2). Parameters of each model are tuned to achieve their best performance. We also train models with subsets of attributes considering computing costs and compare their accuracy with that of best performing models.

1) Performance of Models: We now describe the training and validation of our classification models for predicting fine-grained and coarse-grained host types. Our classifiers are generated using two popular algorithms: random forest (RF) and multilayer perceptron (MLP).

Random forest (RF) algorithms are tuned by varying the number of trees and the number of attributes for each tree. MLP classifiers are tuned by varying the number of layers and the number of nodes (*i.e.,* neurons) in each layer. The neural network-based classifiers (*i.e.,* MLP) are trained with their batch size set to 200, a log loss function, an LBFGS optimizer,



(a) Tuning RF for fine-grained types.

(b) The best RF model for fine-grained types.

(c) The best RF model for coarse-grained types.

Fig. 9: Performance of classifiers: (a) tuning parameters of RF models for host types, (b) accuracy of the best RF model for fine-grained host classification, and (c) accuracy of the best RF model for coarse-grained host classification.

and a constant learning rate for the scheduler. Fig. 9(a) illustrates the tuning process of an RF model, highlighting the model accuracy (obtained by cross-validation) under each set of parameters. The MLP classifiers are tuned with the number of layers ranging from 2 to 20. In each layer, the number of nodes varies from 2 to 20. The best performance of the fine-grained classifiers is 84.69% (MLP) and 99.62% (RF). For the coarse-grained classifiers, the best performance for MLP and RF are 98.15% and 99.72%. It is clear that our RF models outperform the MLP models at both angularities. We, therefore, use the two best RF models as a baseline for the rest of this paper.

Let us now zoom into the accuracy of our baseline models in predicting the type of hosts. We observe in Fig. 9(b) that more than 96% of fine-grained instances are correctly classified according to our cross-validation. The performance is higher (more than 99%) in certain classes such as website servers and authoritative name servers. We note that 1.90% of file server instances are misclassified as website servers. Also, 2.08% of web proxy instances are misclassified as end-hosts. Moving to the coarse-grained classifier in Fig. 9(c), almost all classes receive an accuracy of more than 99%, while 2.60% of TCP proxy instances are misclassified as end-hosts.

2) Confidence Levels: Our models output a measure of confidence (a value between 0 and 1) with each prediction. Let us start with the fine-grained host type classification whereby every predicted instance (correct and incorrect) is accompanied by a confidence level of more than 0.60, while correctly classified instances come with fairly higher confidence greater than 0.85. A fraction (9%) of misclassified instances receive a confidence level of more than 0.80 (relatively high). By analyzing their attributes, we found that affected hosts indeed displayed a different network behavior other than their expected type. For example, a NAT gateway is classified as a web proxy with an 0.87 confidence. This host only had a small number of TCP flows destined to external services *TCP/443* and *TCP/80*, representing the typical behavior of a low-profile web proxy. The majority of misclassified fine-grained instances carry a confidence level lower than 0.80. They are likely to be associated with non-typical behavioral patterns; hence, they require further analysis such as inference by the coarse-grained classifier, annotation of their popular transport-layer services, or deeper pack-based investigation. As another example, a website server is classified as end-host with low confidence of 0.61. After looking into its attributes, we found that it holds mixed functionalities including DNS, remote accessing, file storage, and accessing external servers via an extensive range of internal transport-layer services. We made similar observations with the coarse-grained classifier.

TABLE IV: Fine-grained models using subsets of attributes.

Configuration	Best model	Hyper parameters
Full attribute set	99.62%	80 attributes, 300 trees
Outbound↑ only	99.51%	50 attributes, 200 trees
Inbound↓ only	83.27%	50 attributes, 150 trees
↑ excluding ultra-high costs	99.42%	50 attributes, 200 trees
↑ low & med. costs only	**98.88%**	**30 attributes, 200 trees**
↑ low costs only	35.32%	5 attributes, 50 trees

Recall that our dual-grained classification scheme (§IV-A) classifies a given enterprise host into ten fine-grained types and six coarse-grained types. When the confidence of the fine-grained model is not high enough, network operators refer to the prediction of the coarse-grained model. However, the coarse-grained classifiers may still give a low-confidence prediction, requiring further investigations. In our prototype deployment (will be described in §V), enterprise hosts that receive a low confidence level from the coarse-grained model will be isolated for deeper packet-level investigation.

3) ML Classifiers with Partial Information: We now analyze the performance of our models trained on subsets of the 256 attributes considering computing costs and retention period, making them more suitable and scalable for real-time operation in large enterprise networks.

Optimizing Attribute Selection: We first train and tune RF models for both classification tasks using various combinations of attributes considering their qualitative costs. Table IV summarizes the performance of best performing fine-grained models at various configurations – we have omitted results of the coarse-grained models as similar observations were made for both types of models.

The baseline model with 99.62% accuracy (top row in Table IV) is trained on the full set of attributes (§IV-C).

TABLE V: Size of data structure and accuracy of models as a function of retention period.

Retention period	Avg. # entries	Fine-grained	Coarse-grained
15 min	15.9M	98.91%	98.89%
5 min	5.4M	98.89%	98.82%
1 min	**959K**	**98.38%**	**98.53%**
30 sec	433K	86.84%	87.19%
15 sec	254K	72.58%	60.35%

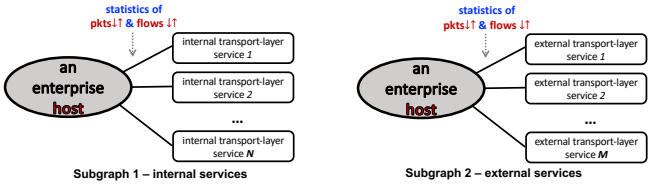


Fig. 10: Two optimized sub-graphs providing cost-effective attributes of an enterprise host.

Interestingly, only outbound attributes (half of the attributes) yield a model with a very similar accuracy of 99.51% (second row). However, inbound half of the attributes cannot achieve better than 83.27% accuracy (third row). This is probably because outbound traffic is less polluted than inbound. Focusing on the outbound traffic, we can still achieve fairly high accuracy of 99.42% if we exclude ultra-high attributes (fourth row). The overall accuracy is slightly compromised to 98.88% by considering only low and medium-cost attributes of outbound traffic (fifth row). However, as highlighted by the sixth row, we cannot further optimize the cost when only low-cost attributes are used to train the model since the obtained accuracy is far unacceptable. Therefore, we choose to continue with the best models trained on low-cost and medium-cost attributes of outbound traffic (*i.e.,* 36 attributes in total), given a combination of prediction and cost metrics.

Practical Graph Structure: Given the selected 36 attributes for our cost-effective classification, we now refine our rooted graph structure to make it practical for operations in a large enterprise. As discussed in §IV-B3, low-cost and medium-cost attributes require a subset of nodes in the original four-layer graph. Therefore, instead of maintaining a dense and expensive graph, shown in Fig. 3, we deduce two sub-graphs, shown in Fig. 10, each with two-layer nodes (*i.e.,* enterprise host→internal services and enterprise host→external services). It can be seen that the complexity of our structure is significantly reduced from $N \times M \times Q$ to $N + M$, ensuring our inference scheme scales cost-effectively while yielding an acceptable classification accuracy.

Tuning Retention Period: As mentioned earlier in §IV-B2, our attributes are computed by setting the retention period to 1 hour, which is relatively expensive to maintain states, particularly at scale. Therefore, we investigate the impact of shorter retention period on the accuracy of our models. Table V shows various settings and their corresponding impact on both fine-grained and coarse-grained models.

In addition to model accuracy, we compute the average number of entries (*i.e.,* key-value pairs) in the data structure for maintaining graphs in our dataset. It can be seen that both the model accuracy and the average number of entries (size of data structure) fall as the retention period gets shorter. The retention period of one minute seems to be the sweet spot in terms of accuracy (more than 98%) and size of data structure (less than a million entries). For the rest of this paper, we set the retention period to a minute.

4) Testing our Models on a Fresh Open Set: To evaluate the efficacy of our classification scheme, we test its performance against an open set derived from our 1-hour PCAP trace (discussed in §III) not shown to the model during the training phase. Attributes of hosts (with our ground-truth label) for both granularities are selected and calculated in the same way discussed above (*i.e.,* 1- min retention period with low-cost and medium-cost attributes of outbound traffic only).

Performance of the Two Models: Both models give high accuracy of 99.76% (fine-grained) and 98.57% (coarse-grained), interestingly slightly higher than those obtained during the cross-validation phase (in §IV-C3). Note that our open set (testing data) was obtained from the traffic of a busy working hour (*i.e.,* 9-10 am) when enterprise hosts are likely to be highly active and display clear network behavioral patterns. In contrast, the closed set (training and cross-validation data) corresponds to host behaviors during the entire day (covering both working and off-work hours).

Mis-Classified Instances: For those instances that were misclassified, the models display low confidence levels below 0.70. Manually investigating into their packet traces, we found two reasons for misclassification: (i) mixed functionality: a website proxy configured by a school also provides DNS resolution, file storage, remote accessing, and Redis proxy services, which are not recommended best practices for asset management and network security; and (ii) compromised and targeted by attacks: an end-host was correctly classified with high confidence levels (*i.e.,* ≥ 0.9) consistently for 13 minutes, and thereafter its predicted class with relatively low confidence levels (between 0.4 and 0.6) fluctuates between UDP server and UDP proxy for 47 minutes. We inspected its traffic during the low-confidence period and found that this host, in addition to its regular activities, was sending repeated DNS queries (asking for `10.129.14.2xy.in-addr.arpa`). at a constant rate of 2 packets-per-second to a public recursive resolver managed by ARIN. The host behaved like a bot-infected device in a query flooding attack [30] (possibly distributed across many bot devices).

Therefore, to help network administrators better identify cyber risks associated with their "suspicious" hosts (displaying unexpected behaviors and receiving low confidence scores from trained models), later in §V-A, we introduce a reactive mechanism using programmable networks that dynamically and selectively collects packets specific to the "focused" hosts for deeper investigation.

V. System Prototype and Campus Field Trail

In this section, we prototype a practical system to: (a) classify enterprise hosts in real-time via our dual-grained classification scheme, and (b) dynamically isolate and inspect

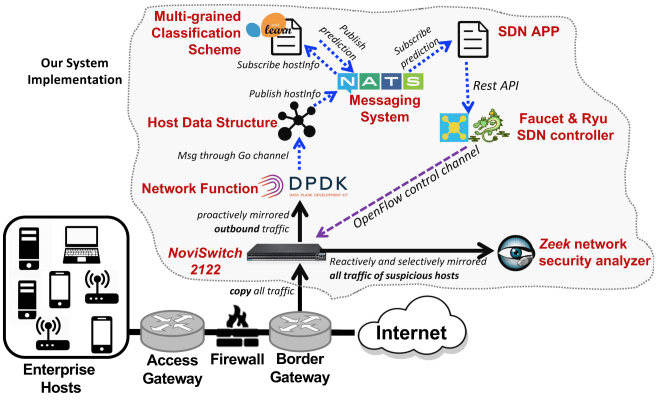


Fig. 11: Prototype implementation.

full traffic of "suspicious" hosts that are with low-confident predictions for a deeper packet-level diagnosis. Dynamic and reactive diagnosis is enabled by software-defined networking (SDN) techniques. We begin with the design of our prototype. We next draw insights into results of host classification and forensic analysis on the isolated traffic of suspicious hosts. Finally, we explain the system performance from a one-month trial.

A. Design and Implementation of our Prototype

We implemented our system using commodity hardware and open-source tools/libraries, and deployed it at the edge of our university campus network. Implementation details are shown in Fig. 11. Our system receives inbound and outbound live traffic of the entire campus network (provisioned by the IT department of our university) via two 10 Gbps links. For our deployment, each functional block runs as an independent micro-service [20], and hence our system is more resilient to the failure of individual modules.

An SDN switch (NoviFlow 2122 [39]) is instructed by an SDN application we developed for the Faucet SDN controller (inserting proactive static rules) and the Ryu SDN controller (managing run-time reactive rules). Proactive rules mirror outbound traffic to a generic server configured with Ubuntu version 16.04.4, which hosts our virtual network function (VNF) written in Golang using DPDK (data plane development kit) framework and the NFF-Go [36] library. The VNF parses packets, extracts required metadata, and updates our host data structure. Attributes of each host are calculated and periodically forwarded to the trained models (dual-grained classification scheme developed in Python3) via a messaging system (NATS) that acts as a data broker for information exchange. Predictions (classified labels and confidence levels) are published to the messaging system.

The hosts that receive a low confidence score (*i.e.,* less than 0.8) from the coarse-grained model (suspicious hosts) will need further and deeper investigation to determine the cause of deviation from their expected behavior. Such deep inspections for a host last till it receives a high confidence score. To achieve this, the SDN component of our system dynamically inserts reactive rules to mirror both inbound and outbound traffic of those suspicious hosts – rules are initiated by the SDN app to the Ryu controller through RESTful APIs.

To demonstrate the selective packet inspection of suspicious hosts, we set up Zeek [53] (a popular open-source security analyzer previously known as Bro). This software-based deep packet analyzer does not scale to a large volume of traffic. Our system, instead, feeds Zeek with a minor fraction of the entire traffic, only that belongs to the hosts that are flagged suspicious by the inference models.

B. Insights into Campus Host Types

We now draw insights into the types of campus hosts predicted by our trained models and the consistency of their prediction by analyzing results of our system during the trial.

1) Fine-Grained and Coarse-Grained Classes: By analyzing the prediction results of our system during the field trial, we obtained the label (of network roles and behaviors) for a large set of hosts not managed by our organizational IT department. Table VI and VII summarize the classification results of fine-grained and coarse-grained types, respectively.

Types of hosts we determined their ground-truth type by their DNS name (discussed in §III-B2) have been validated. Note that some of those hosts (5 website servers, 4 remote computing platforms, and 1198 end-hosts.) were not present on the network during the field trial – data used in §III-B2 was collected nine months prior to the field trial. In addition, a total of 18,619 previously unknown hosts are classified to their fine-grained types for at least half of their active life during working hours (*i.e.,* 9am – 5pm), as shown by the rightmost column of Table VI, including 306 website servers, two authoritative name servers, 147 remote computing platform, four file storage servers, one mail server, two DNS proxies, three web proxies, 13 NAT gateways, and 18,128 end-hosts.

As expected, the fine-grained type can be determined confidently for a portion (*i.e.,* not all) of active hosts. The hosts, which do not receive a confident prediction from the fine-grained models, are classified by the coarse-grained model with relatively high confidence (above 0.80). We have found 1,774 TCP-dominant servers, 63 UDP-dominant servers, a non-typical NAT gateway, and 1,460 non-typical end-hosts. The coarse-grained predictions are accompanied by their dominant transport-layer services. For example, two TCP-dominant servers consistently used TCP/3306 (used by MySQL servers) as their top services, and one UDP-dominant server that is heavily active on UDP/427, suggesting a CIM server for managing hardware health information.

2) Behavioral Consistency of Enterprise Hosts: We note that some hosts may display different behavioral patterns during certain periods. Main servers (*e.g.,* organizational servers configured by the IT department) are consistently found to behave as their expected class with high confidence. However, subsidiary servers (managed by sub-departments) displayed variable behaviors during different hours. As an illustrative example, we show in Fig. 12(a) the time-trace of the model confidence for a server (the student web portal of our university), which is mostly classified as a website server with a few instances misclassified as end-host throughout our field trial.

TABLE VI: Summary of hosts with fine-grained types from the one-month campus field trial.

Fine-grained host type	# host	avg consistency	avg consistency (wrk)	avg utilization	**# previously unknown hosts**
Website server	362	0.93	0.98	0.33	**306**
Authoritative name server	17	0.96	0.99	0.65	**2**
VPN Server	13	0.94	0.98	0.41	**0**
Remote computing platform	159	0.83	0.89	0.34	**147**
File storage server	18	0.93	0.95	0.26	**4**
Mail server	19	0.98	0.92	0.34	**1**
DNS proxy	9	0.81	0.88	0.37	**2**
Web proxy	7	0.76	0.79	0.44	**3**
NAT gateway	272	0.61	0.89	0.61	**13**
End-host	18,891	0.99	0.97	0.17	**18,128**

TABLE VII: Summary of hosts with coarse-grained types obtained from the one-month campus field trial.

Coarse-grained host type	# host	avg consistency	avg consistency (wrk)	avg utilization	**Low-conf. predictions**	**Low-conf. hosts**
TCP-dominant Server	2,005	0.55	0.77	0.19	65,978	**231**
UDP-dominant Server	144	0.79	0.81	0.25	23,061	**81**
TCP-dominant Proxy	0	0	0	0	0	**0**
UDP-dominant Proxy	8	0.17	0.81	0.38	2,919	**8**
Non-typical NAT Gw.	19	0.43	0.85	0.56	971	**18**
Non-typical End-host	1,946	0.88	0.92	0.24	230,029	**486**

It can be seen in Fig. 12(a) that the model confidence is fairly high (close to 1) when this host is classified as a website server (dashed blue lines). We observe that the model confidence for predicting it as an end-host (solid red lines) is fairly low (mostly $<$0.4), with a few instances crossing 0.6 – only an instance exceeds 0.8, resulting in misclassification. We manually verified that this server undertakes routine maintenance (*i.e.,* fetching updates from the Internet) around midnight.

Proxies and NAT gateways tend to display varying profiles as their network activities depend highly on internal user behaviors. Therefore, these networked assets will likely get predicted as end-hosts during off-peak hours. For example, Fig. 12(b) illustrates the model prediction for a NAT gateway in our field trial. During working hours (9am – 7pm) on weekdays and weekends between 18^{th} Nov and 7^{th} Dec, the host was classified as a NAT gateway with high confidence (the blue dashed line). In contrast, it gets labeled as end-host with high confidence (the solid red line) during idle hours, including night time and days of study period (before final examinations of the academic term).

Note that each host may receive different predictions (classes) from the trained models during its lifetime. Therefore, the "consistency" of our inference models per each enterprise host is an important metric to measure. We compute a measure of consistency (per host), which is the fraction of time the host is classified as its most frequent type (class). For illustration purpose, within a particular class (fine-grained and coarse-grained) we compute the overall average consistency (third column) and the average consistency during working hours 9am – 5pm that are respectively reported in the third and fourth columns of Tables VI and VII. The average utilization (active fraction of lifetime) per class is shown in the fifth column of the two tables.

End-hosts and servers display a fairly consistent behavior compared to other types. Proxies and NAT gateways may behave as end-hosts during the inactive time, resulting in relatively low behavioral consistency. Hosts across all types (fine-grained and coarse-grained) display a more distinct behavior of their type (hence receive consistent prediction) during working hours.

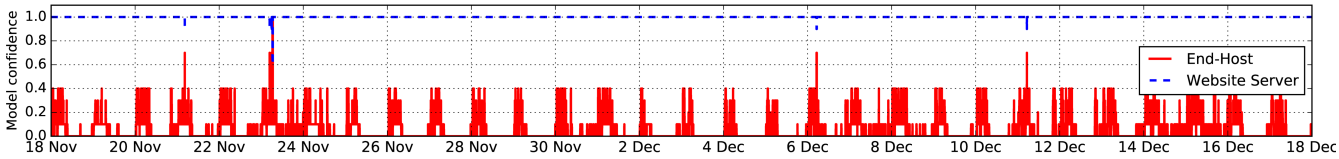
Another observation is that many servers (especially those configured by sub-departments) are fairly under-utilized (the probability of being active is less than 30%). For example, four of 9 DNS proxies are adequately utilized (more than 16 hours a day), while others are relatively idle (*i.e.,* less than 2 hours of activity per day); thus can be candidates for getting merged (for economic and security management reasons) with other proxies on the network.

Lastly, for those hosts whose reliable prediction is only available by the coarse-grained model, 824 of them receive a low confidence score (less than 0.8) for a total of 322K prediction instances (details are shown in Table VII). Once a host receives a low-confidence prediction from the coarse-grained model, its entire traffic (inbound and outbound) is mirrored for a deep packet inspection – we use Zeek in our prototype. The host remains under the deep-inspection mode until it receives a high-confidence prediction.

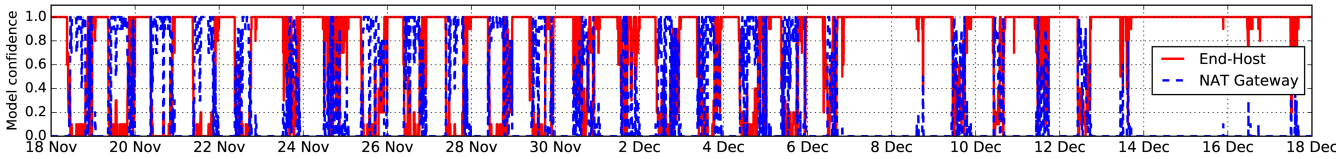
C. Insights into Suspicious Hosts from Deep Packet Inspection

During the field trial, the Zeek packet inspector raised a total of 381,499 packet-level alarms for 714 suspicious hosts, including 465 end-hosts, 18 NAT gateways, 6 UDP proxies, 45 UDP servers, and 159 TCP servers. Resulted alerts are from 32 types. The top alert types are "`truncated_tcp_payload`" indicating TCP-based attacks using crafted packets, "`possible_split_routing`" indicating single directional flows that may belong to scans and DDoS floods, "`data_before_established`" for potential volumetric anomalies, and "`inappropriate_FIN`" for TCP-FIN based anomalies.

The distribution of alerts per host would help IT departments infer the root cause of their abnormal behaviors. For example, we found that a third of all resulted alerts correspond



(a) Main website server: it is either classified as website server or end-host.



(b) NAT gateway: it is either classified as NAT gateway or end-host.

Fig. 12: Time-trace of model confidence per class for two host examples: (a) website server and (b) NAT gateway.

to only 11 suspicious hosts, suggesting extra attention in the forensics analysis. In what follows, we discuss our manual investigations specific to suspicious hosts whose behaviors triggered 8% (UDP proxy) and 5% (TCP server) of alerts, respectively. We emphasize that a more systematic forensic analysis is beyond the scope of this paper and is left for future studies.

Let us start with the suspicious DNS (UDP) proxy, configured by an affiliated organization of our university, that consistently displays typical behaviors of UDP proxies most of the time. However, every 5 to 10 minutes (periodically), this host is misclassified (with low confidence) as a NAT server during peak hours and as an end-host or even TCP proxy during off-peak hours. Analyzing the Zeek logs, we found this host generates many TCP SYN-ACK packets followed by empty ACK packets targeting a range of TCP ports on external victims (on Microsoft Azure cloud) for a minute and then goes idle for 5-10 minutes. Such behaviors result in a large number of alerts of "`truncated_tcp_payload`", "`SYN_with_data`", and "`TCP_seq_underflow_or_mismatch`" for malformed TCP packets and incomplete connections. This host seems to be infected by malware to participate in TCP ACK-based scans [37] or flooding [43] activities.

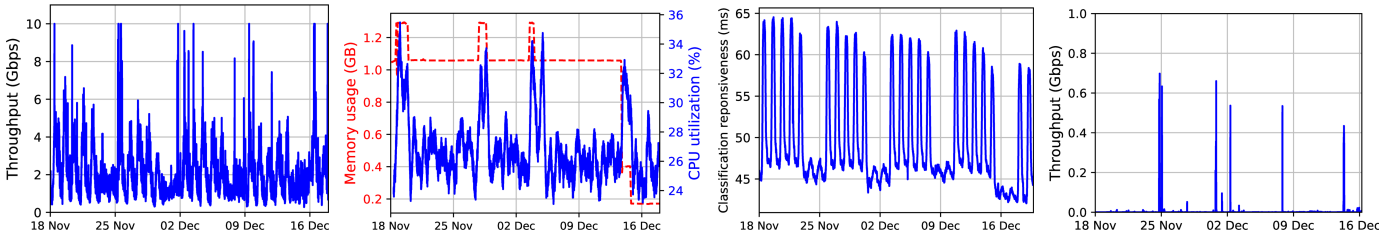
Moving to the suspicious TCP server that purely offers TCP/3274 was classified as TCP server for most (86%) of its active time. For other times, we found the prediction of this host fluctuates between end-host and TCP server with fairly low confidence scores (about 0.3). Analyzing the alerts generated by the Zeek tool, we see that in addition to its typical inbound traffic, the server sent many outbound TCP SYN packets (without any response) targeting TCP/443 on a wide range of external victims (from a block of /16 IP address dedicated to Amazon cloud services) – about 5 packets per victim. These suspsicious packets led to frequent alerts such as "`window_recision`" and "`TCP_ACK_underflow_or_mismatch`". This pattern suggests that the host is possibly involved in SYN reconnaissance attacks, probing the availability on HTTPS service on external hosts as a preliminary step before reflection attacks [44].

D. System Performance

As discussed in §V-A, our system operates multiple software modules (*e.g.,* Network Function and Host Data Structure), which run on a commodity Ubuntu server equipped with four-core 2.10GHz CPU, 62GB RAM, two 10Gbps network interfaces handling data-plane traffic, and two 1Gbps network interfaces for management communications. We now report the real-time performance our system during our field trial. Fig. 13(a) shows the throughput of the entire outbound network traffic (proactively mirrored) processed by the Network Function engine (which only processes packet headers). The rate of analyzed traffic varies between 0.3 Gbps to 10 Gbps, where daily peaks occur around mid-day on weekdays. In Fig. 13(b), CPU utilization of our server (which hosts traffic parsing VNF, host graph data structure, classification scheme, and SDN controller) follows a periodic pattern and is bounded between 24% and 36%, as shown by solid blue lines. Also, the memory usage varies from 0.2 GB to 1.3 GB, as shown by dashed red lines – note that the host-based data structure contributes to majority of memory usage. Fig. 13(c) illustrates the responsiveness of the classification scheme called every minute, which is less than 65 ms even during peak hours, proving that our system can give real-time inference of enterprise hosts behavior. Lastly, reactively mirrored traffic load (to fine-grained packet inspection engine via our SDN-based mechanism) for "focused" enterprise IP addresses is shown in Fig. 13(d). It was light enough (*i.e.,* typically below 300 Mbps) to be processed by a computational extensive deep packet inspector.

VI. Discussions

In this section, we discuss specific limitations of our work that could be addressed by future research: **(i)** we profile asset behaviors from their Internet communications measured at the border of an enterprise network, and thus, those enterprise assets that only communicate locally are missed. Future research may aim to obtain insights into how analyzing internal communications can extend and improve the inference from pure Internet (external) communications at additional computing costs; **(ii)** operators may find it challenging to manually train and tune the two models (fine-grained and coarse-grained) with their own network data. Therefore, automating the process of model tuning could further enhance the utility of our approach; and, **(iii)** hosts with suspicious behaviors may be involved in unintended (or perhaps malicious) activities. We employ an off-the-shelf packet inspector



(a) Proactive mirrored load. (b) CPU & memory usage. (c) Responsiveness. (d) Reactive mirrored load.

Fig. 13: Real-time performance of our system: (a) proactive mirrored load, (b) CPU and memory usage of the server hosting VNF, host data structure, inference engine and SDN controller, (c) responsiveness of the classifier (per epoch), and (d) reactive mirrored load.

for further investigations. As a future research direction, one may choose to develop specialized models (one per specific class) that can better perform security and/or operational health investigations.

VII. Conclusion

Real-time classification of hosts and tracking their behavior are critical for enterprise network operators to manage their network assets effectively and securely. In this paper, we developed a method that continuously classifies and monitors the network behavior of enterprise hosts. We conducted a large-scale analysis on traffic traces of an enterprise network and characterized network behavioral patterns of various host types at two levels of granularity. We then identified 36 (out of 256) cost-effective but significant attributes of host behavior that are computed from two lightweight graph structures and developed a multi-grained inference scheme consisting of a ten-class classifier and a six-class classifier that yields a high accuracy of 99%. Finally, we built a practical system empowered by software-defined networking and virtual network functions and deploy it in a large university network. We presented insights obtained over a month field trial, such as the ability to identify hundreds of typical servers and their utilization and thousands of non-typical assets, and highlight anomalous behaviors pertinent to possible cyber-threats.

References

[1] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescap, "Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges," *IEEE Transactions on Network and Service Management*, Jun 2019.

[2] Akamai Technologies, "How securing recursive dns proactively protects your network," https://bit.ly/3kjFgTI, 2017, accessed: 2021-05-14.

[3] S. Alcock, J.-P. Möller, and R. Nelson, "Sneaking past the firewall: Quantifying the unexpected traffic on major tcp and udp ports," in *Proc. ACM IMC*, Santa Monica, California, USA, Nov 2016.

[4] D. Apiletti *et al.*, "Selina: A self-learning insightful network analyzer," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 696–710, Sep 2016.

[5] Balbix, "The 10 most common cybersecurity posture blindspots," https://bit.ly/3s9CNNO, 2020, accessed: 2020-11-26.

[6] T. Benson, A. Akella, and D. Maltz, "Unraveling the Complexity of Network Management," in *Proc. NSDI*, Boston, Massachusetts, USA, Apr 2009.

[7] D. Berrar and W. Dubitzky, *Information Gain*. New York, NY: Springer New York, 2013, pp. 1022–1023.

[8] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, Jun 2018.

[9] C. Brignone. et al., "Real time asset tracking in the data center," *Distributed Parallel Databases*, vol. 21, pp. 145–165, Jun 2007.

[10] A. Dainotti *et al.*, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, Jan 2012.

[11] S. Darji, "What is Asset Discovery? Benefits, Best Practices Explained," https://cybersecurity.att.com/blogs/security-essentials/why-is-asset-discovery-crucial-for-any-business, 2020, accessed: 2022-01-31.

[12] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "Spotlight: Detecting anomalies in streaming graphs," in *Proc. ACM KDD*, London, United Kingdom, Aug 2018.

[13] S. K. Fayaz *et al.*, "Bohatei: Flexible and Elastic DDoS Defense," in *Proc. USENIX Security*, Washington, D.C., USA, Aug 2015.

[14] D. Gruber, "As IT Complexity Increases, Visibility Plummets," Axonius, Tech. Rep., Mar 2020.

[15] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-driven streaming network telemetry," in *Proc. ACM SIGCOMM*, Budapest, Hungary, Aug 2018.

[16] H. Habibi Gharakheili *et al.*, "itelescope: Softwarized network middlebox for real-time video telemetry and classification," *IEEE Transactions on Network and Service Management*, Sep 2019.

[17] X. Hu *et al.*, "Baywatch: Robust beaconing detection to identify infected hosts in large-scale enterprise networks," in *Proc. IEEE/IFIP DSN*, Toulouse, France, Jun 2016.

[18] K. Jackson, "What causes network blind spotsand how to prevent them," https://bit.ly/3bVIzMC, 2019, accessed: 2020-11-26.

[19] A. Jakalan *et al.*, "Social relationship discovery of ip addresses in the managed ip networks by observing traffic at network boundary," *Comput. Netw.*, vol. 100, no. C, p. 1227, May 2016.

[20] P. Jamshidi, C. Pahl, N. C. Mendonca, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Software*, vol. 35, no. 03, pp. 24–35, May 2018.

[21] P. Kalmbach, D. Hock, F. Lipp, W. Kellerer, and A. Blenk, "Noracle: Who is communicating with whom in my network?" in *Proc. ACM SIGCOMM Posters and Demos*, Beijing, China, 2019.

[22] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, and M. Faloutsos, "Is p2p dying or just hiding?" in *Proc. IEEE GLOBECOM*, Dallas, TX, USA, Nov 2004.

[23] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," in *Proc. ACM SIGCOMM*, Aug 2005.

[24] J. Kohout and T. Pevn, "Network traffic fingerprinting based on approximated kernel two-sample test," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 788–801, Mar 2018.

[25] V. Labayen, E. Magaa, D. Morat, and M. Izal, "Online classification of user activities using machine learning on network traffic," *Computer Networks*, Nov 2020.

[26] B. Li, M. H. Gunes, G. Bebis, and J. Springer, "A supervised machine learning approach to classify host roles on line using sflow," in *Proc. ACM HPPN*, New York, USA, Jun 2013.

[27] C. Liu, A. Raghuramu, C.-N. Chuah, and B. Krishnamurthy, "Piggybacking network functions on sdn reactive routing: A feasibility study," in *Proc. ACM SOSR*, Santa Clara, CA, USA, Apr 2017.

[28] Z. Liu *et al.*, "Nitrosketch: Robust and general sketch-based monitoring in software switches," in *Proc. ACM SIGCOMM*, Beijing, China, Aug 2019.

[29] I. Livadariu *et al.*, "Inferring Carrier-Grade NAT Deployment in the Wild," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, Oct 2018.

[30] M. Lyu *et al.*, "Hierarchical anomaly-based detection of distributed dns attacks on enterprise networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 1031–1048, Mar 2021.

[31] M. Lyu, H. Habibi Gharakheili, C. Russell, and V. Sivaraman, "Mapping an Enterprise Network by Analyzing DNS Traffic," in *Proc. Springer PAM*, Puerto Varas, Chile, Mar 2019.

[32] M. Mesarina. et al., "Automating server tracking for data centers," in *Proc. ACM SenSys*, Baltimore, Maryland, Nov 2004.

[33] S. Marshall, "CANDID: Classifying Assets in Networks by Determining Importance and Dependencies," University of California at Berkeley, Electrical Engineering and Computer Sciences, Tech. Rep., May 2013.

[34] M. H. Mazhar and Z. Shafiq, "Real-time Video Quality of Experience Monitoring for HTTPS and QUIC," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, Apr 2018.

[35] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescap, "Xai meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, Jul 2021.

[36] L. Q. Nguyen, "Build Performant Network Functions with NFF-Go," https://intel.ly/2PcR3XB, 2019, accessed: 2019-06-28.

[37] Nmap, "TCP ACK Scan," https://bit.ly/3BZIp10, 2019, accessed: 2019-08-29.

[38] L. Nobach, O. Hohlfeld, and D. Hausheer, "New kid on the block: Network functions visualization: From big boxes to carrier clouds," *SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 3, Jul. 2018.

[39] NoviFlow, "NoviSwitch 2122 High Performance OpenFlow Switch," https://bit.ly/3wt4jIS, 2018, accessed: 2018-28-1.

[40] E. Papadogiannaki, C. Halevidis, P. Akritidis, and L. Koromilas, "Otter: A scalable high-resolution encrypted traffic identification engine," in *Proc. RAID*, Heraklion, Crete, Greece, Sep 2018.

[41] PCI Security Standards Council LLC, "The prioritized approach to pursue pci dss compliance," Tech. Rep., 2018.

[42] K. Potdar, T. Pardawala, and C. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *International Journal of Computer Applications*, vol. 175, pp. 7–9, 10 2017.

[43] Red Button DDoS Experts, "Ack flood," https://www.red-button.net/ddos-glossary/ack-flood/, 2019, accessed: 2019-05-10.

[44] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," in *Proc. NDSS*, San Diego, CA, USA, Feb 2014.

[45] A. Sivanathan *et al.*, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE Transactions on Mobile Computing*, 2019.

[46] G. Tan, M. Poletto, J. Guttag, and F. Kaashoek, "Role classification of hosts within enterprise networks based on connection patterns," in *Proc. USENIX ATEC*, San Antonio, Texas, Jun 2003.

[47] J. Touch *et al.*, "Service name and transport protocol port number registry," https://bit.ly/3bTnPF7, 2019, accessed: 2019-03-24.

[48] T. Yang *et al.*, "Elastic sketch: Adaptive and fast network-wide measurements," in *Proc. ACM SIGCOMM*, Budapest, Hungary, Aug 2018.

[49] T.-F. Yen *et al.*, "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proc. ACSAC*, New Orleans, Louisiana, USA, Dec 2013.

[50] T. Yu *et al.*, "PSI: Precise Security Instrumentation for Enterprise Networks," in *Proc. NDSS*, San Diego, CA, USA, Feb 2017.

[51] S. E. Yusuf *et al.*, "Security Modelling and Analysis of Dynamic Enterprise Networks," in *Proc. IEEE CIT*, Nadi, Fiji, Dec 2016.

[52] A. Zand, A. Houmansadr, G. Vigna, R. Kemmerer, and C. Kruegel, "Know your achilles' heel: Automatic detection of network critical services," in *Proc. ACSAC*, Los Angeles, CA, USA, Dec 2015.

[53] Zeek, "The zeek network security monitor," https://www.zeek.org, 2019, accessed: 2019-11-21.

[54] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "Coda: Toward automatically identifying and scheduling coflows in the dark," in *Proc. ACM SIGCOMM*, Florianopolis, Brazil, Aug 2016.

Minzhao Lyu received his B.Eng. (First Class Hons.) and Ph.D. degree from the University of New South Wales, Sydney, Australia in 2017 and 2022 respectively. He has worked at CSIRO's Data61, Sydney, Australia as a student fellow and at National Telemedicine Center of China as a research intern. He is currently a Postdoctoral Research Associate at the University of New South Wales, Sydney, Australia. His research interests include computer networks, network data analytics, network security, programmable networks, and 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.

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.