Chapter 1

On Offloading Network Forensic Analytics to Programmable Data Plane Switches

Kurt Friday
The Cyber Center For Security and Analytics
The University of Texas at San Antonio, USA

Elias Bou-Harb
The Cyber Center For Security and Analytics
The University of Texas at San Antonio, USA

Jorge Crichigno
Integrated Information Technology
The University of South Carolina, USA

Mark Scanlon
Computer Science and Informatics
University College Dublin, Ireland

Nicole Beebe
The Cyber Center For Security and Analytics
The University of Texas at San Antonio, USA

The extent to which cyber crimes are now being executed has reached a frequency that has never been observed before. To detect these events and extract relevant network artifacts for investigations, network forensics has long been the de-facto approach. However, the time and data storage necessary to perform traditional forensic procedures has put investigators at odds, often resulting in substantial artifact extraction latency and poor incident response. To mitigate what have now become inherent pitfalls for the forensics community, we propose a novel means of transforming network forensics to a procedure that functions at line rate, while the event of interest is taking place, by harnessing the newfound programmable switch technology.

Amid the prevailing cyber-crime themes dominating today's headlines are Distributed Denial of Service (DDoS) activities and the misuse of Internet of Things (IoT) devices. To this end, we implement two switch-based use cases for conducting the relevant network forensics associated with each of these classes of misdemeanors, respectively. In particular, the first use case employs dynamic thresholds generated from real-time artifact statistics extracted by the switch to infer contemporary DDoS attacks. The empirical results confirm that the proposed approach mitigates UDP amplification at line rate and SYN flooding attacks within a fraction of a second. Moreover, the complete remediation time of slow DDoS is reduced from near 10 seconds down to 2 seconds. The second use case instruments the switch with a rule-based Projective Adaptive Resonance Theory (PART) algorithm to accurately fingerprinting the origin IoT device of network traffic from a single TCP packet at line rate. We also provide a methodology for automating the translation of such rule-based Machine Learning (ML) output to P4 programs, thereby enabling its deployment without the need for additional background expertise. The proposed fingerprinting engine was evaluated against a dataset consisting of devices of both IoT and non-IoT in nature. The results indicate that such devices can be fingerprinted with 99% accuracy. It is our hope that the research undertaken herein not only aids in the conducting of efficient and effective network forensic procedures associated with DDoS attacks and IoT devices but also in promoting the utilization of programmable switches in future forensic research endeavors. Furthermore, we expect that the proposed approach's automated translation of rule-based classifiers into P4 code will provoke the subsequent harnessing of ML's pattern recognition abilities for enhancing a number of other network forensic tasks on the switch.

1. Introduction

A network forensic practitioner's essential tasks of monitoring, inspecting, and attributing network traffic to cyber crimes has become increasingly challenging due to the extent of which such misdemeanors are currently taking place. Further, these challenges are often compounded by more sophisticated attacks (e.g., anti-forensic strategies) launched by adversaries. Several factors have contributed to this increase in cyber crime and attack sophistication, such as society's growing dependence upon the Internet,[1] the enhanced inter-connectivity amid modern technology,[2–4] an assortment of open source exploitation code and tools, widely-available attack services (e.g., DDoS-for-hire), and even the COVID-19 pandemic.[5] Moreover, the immense rates at which information is transferred between contemporary machines has led to an exponential increase of data that must be analyzed.

As a result, the amount of time and resources necessary to conduct effective investigations has also risen substantially.[6]

Unfortunately, it is also now commonplace for attackers to leverage the insecurity of the vast IoT domain for a means of conducting cyber crime. The excessive heterogeneity of these devices combined with their expedited deployment by vendors (i.e., leading to subpar security mechanisms and patching) has has left adversaries with a plethora of vulnerabilities to exploit.[7] Consequently, a very large uptick in DDoS attacks has been observed, including some of the largest recorded to date, e.g., the record-breaking attack on GitHub in 2018[8] and the majority of the more pronounced attacks that occurred in 2019.[9] Moreover, such vulnerabilities have been exploited to conduct a broad range of other malicious endeavors, ranging from gaining entry to critical infrastructure (e.g., power grids[10]) to adversaries overtaking upwards of a million devices at one juncture to launch various campaigns (e.g., spam, cryptomining, etc.)[11,12]

Ultimately, this extensive attack surface has left the forensic community with the tall task of investigating and attributing such crimes with largely offline analysis procedures. To put matters into perspective, a 10 Gbps flow of traffic using only a two-hour sliding window necessitates 10 TB of storage, and 20 Gbps utilizing a 12-hour sliding window requiring 1 PB. Furthermore, these numbers pale in comparison to the large traffic rates today's networks often encounter. For example, it has been projected that backbone networks may experience up to 170 Tbps in 2021.[13] Moreover, while these time-consuming, offline investigation procedures can eventually lead to attribution of a cyber crime, the resultant delays in identifying attacks naturally create challenges for mitigating them while they are in progress. In addition, these delays give adversaries more time to launch ensuing attacks, evade prosecution after an attack transpires (e.g., via anti-forensic attempts, fleeing, etc.), and so forth. In turn, investigators have the monumental task of monitoring and safeguarding the capture of the offending traffic amid the overwhelming rates of traffic modern networks observe. Once this objective has been completed, investigators must subsequently analyze the resultant stockpile of capture data for viable artifacts in a very small time window to ensure a successful investigation, circumvent anti-forensic attempts, and mitigate damages. Indeed, streamlining aspects this arduous process would dramatically enhance its effectiveness.

Until very recently, performing these responsibilities at line rate as the malicious traffic is traversing the wire was largely an impossibility with the excessive limitations of traditional network implementations. This pit-

fall is rooted in the fact that the devices handling such traffic are either software-based or static in nature. In particular, the software-based solutions generally consist of middleboxes, which cannot conduct complex traffic analysis without substantial degradation to the network's throughput. Alternatively, the devices that can offer better processing capabilities (e.g., switches and routers) customarily have had their behavior encoded in firmware by vendors and offer extremely limited support for network forensic endeavors.

Elaborating upon this notion, network topologies generally can be abstracted as being within the control or data plane.[14] The data plane is responsible for delivering traffic from one device to another, whereas the control plane is essentially the brains of the network and is concerned with establishing links between routers and exchanging protocol information. In the case of the aforementioned traditional networks, both of these plains are integrated into the firmware of routers and switches, and therefore these implementations have relatively fixed behavior. To offer more flexibility, Software Defined Networking (SDN) was proposed which explicitly decouples the two planes and implements the control plane in software; thus, SDN effectively transformed what has characteristically been rigid network functionality into a more efficient and flexible software development procedure. That being said, SDN is still bounded to a small set of forwarding protocols (e.g., IP, Ethernet) entertained by the data plane, which severely restricts the number of applications that can be employed which utilize the enhanced processing capabilities of the data plane's forwarding devices. Moreover, an attempt to amend this short list of protocols to implement additional applications generally requires years of waiting, given the data plane has characteristically been made up of proprietary devices with closed source code.[15] As a result, SDN has struggled to keep pace with the excessively dynamic nature of cyber crime.

Fortunately, the P4 language has since surfaced as the de-facto standard for defining the forwarding behavior of data plane. By way of programmable switches, the software that dictates the behavior of how packets are processed can now be developed, tested, deployed, and amended in a much shorter time span. Moreover, such behavior can finally be strictly governed by the given network's operators, resulting in fully customizable implementations for network forensic practitioners. In harnessing this newfound technology, the research conducted herein utilizes programmable data planes to transform the manner in which network forensics has traditionally been conducted. In particular, programmable switches can identify and extract

forensic artifacts at line rate in order to bypass storing a wealth of capture data to subsequently analyze offline. Custom switch-based programs can also use these extracted artifacts in order to fingerprint malicious events in real time amid Tbps traffic rates. This is in stark contrast to the software-based intermediary nodes (i.e., middleboxes) employing Intrusion Detection Systems (IDS), firewalls, etc., which crumble under the tremendous load of modern networks.

To this end, the notion of leveraging programmable data planes for network forensics applications is introduced by proposing two such approaches corresponding to the ever-increasing presence of staggering DDoS attacks and the harnessing of vulnerable IoT devices for malicious endeavors, respectively. In terms of DDoS, note that it can present itself in many forms and one detection strategy might only be able to detect a specific type it was designed for.[16] For example, while an entropy-related approach might be effective when a network is experiencing a flooding attack, it likely will struggle to identify the presence of a stealthier attack. To address this issue, this work combines multiple novel DDoS fingerprinting techniques into one unified detection strategy within a P4-programmed switch. Another aspect of several DDoS detection strategies that can prove problematic is utilizing static thresholds to identify when an attack is occurring. Typically, such thresholds need to be calibrated for a particular network's topology, its expected traffic, when in the day or week it is used, etc., and can also be more easily exploited by savvy adversaries. The proposed mechanism tackles this dilemma by employing dynamic thresholds that adapt to varying network conditions. To evaluate this strategy, three attack scenarios were launched, namely, SYN flooding, UDP amplification, and a stealthy variant, slow DDoS, against the proposed approach deployed on a Behavioral Model version 2 (BMv2)[17] software switch. The results confirm that UDP amplification attacks could be constrained to a dynamically allocated bandwidth coinciding with the given UDP protocol being leveraged by the attacker (e.g., NTP, DNS, etc.), at line rate. Additionally, SYN flooding was shown to be rendered ineffective with benign requests experiencing no latency, and all other TCP-based DDoS types employing various mixtures of set flags (e.g., SYN-ACK flood, FIN flood, etc.) were negated entirely. Finally, the approach was able to restore service to clients endeavoring to connect to the server within 1 second of the slow DDoS attack consuming the server's available connections and to fully remediate the attack by the following second; this is a substantial difference from past approaches that wait for the malicious connections with the target server to time out, which

ultimately takes around 10 seconds or more.

The second approach aims to offer practitioners a means of promptly fingerprinting IoT device traffic on the network. With the increasing instances of cyber crime committed by way of these devices,[18–23] such artifacts can be invaluable to conducting effective investigations. To perform this objective on this switch, a rule-based PART learning algorithm was first trained on the noteworthy dataset proposed by Sivanathan et al.[24] encompassing a thorough mixture of both IoT and non-IoT devices. From the generated rules, a unique methodology for translating the ML algorithm's output to a compact and practical P4 program was proposed. As a result, the entire classification algorithm is deployed entirely on the switch to enable line-rate fingerprinting of IoT devices. The corresponding evaluation of this classification program on BMv2 demonstrates that the exact device type from which the given traffic originates can be identified with 99% accuracy from a single TCP packet. Moreover, a direct correlation between the number of samples pertaining to a specific device in the dataset and the model's classification accuracy was observed, with the devices corresponding to less training samples suffering from more misclassifications. On the contrary, all devices with the maximum amount of samples (50,000 in this experiment) were classified with 99% accuracy. It turn, this suggests that highly accurate device-type classification can be achieved with a sufficient number and balance of samples in the dataset for each device.

In summation, the proposed approach endeavors to advance the state-of-the-art by making the following core contributions:

- Advancing the efficiency of network forensic investigations by leveraging programmable data planes in order to achieve the line-rate fingerprinting of both DDoS attacks and IoT devices. The end result is a dramatic transformation of the traditional time-consuming offline procedures of filtering a large amount of traffic captures into that which can be conducted at line rate. Further, the filtering out of attack traffic effectively circumvents the need to excessively store a wealth of such irrelevant data.
- Improving upon current DDoS protection mechanisms by providing a unified network forensic approach for identifying the broad spectrum of contemporary DDoS attacks within the switch. An adaptive threshold-based approach is used to trigger both artifact extraction and subsequent detection in order to mitigate attacks

immediately following their inception. The evaluation of three attack scenarios prevalent in the wild concurs that the proposed strategy remediates UDP amplification and SYN flooding attacks in fractions of a second, and reduces the complete mitigation time of slow DDoS from to upwards of 10 seconds down to 2. Further, the approach negates all other TCP flooding attacks that fictitiously set flags.

- Presenting an IoT fingerprinting scheme that accurately identifies the IoT devices from the traffic they transmit. When evaluated, the approach was able to identify IoT traffic with 99% accuracy. Moreover, the fingerprinting scheme's evaluation demonstrates that it is not only effective for fingerprinting IoT devices from a single TCP packet, but devices of non-IoT nature as well. In addition, the results suggest that this procedure can be applied for the fingerprinting of the exact device type on the switch by merely incorporating more training samples per device.
- Providing a novel automated methodology for converting ML rule-based output to practical P4 applications on the switch. Further, the proposed methodology has been specifically designed for compact, parallel processing and thereby is extremely practical given its small resource footprint; therefore, P4 programs utilizing it can be employed by network operators next to a multitude of other network-specific, P4 algorithms, and without the need for additional training.

The rest of the paper is organized as follows. In the next section, we cover the related literature. Following these notable works, we discuss some background information and our motivations in Section 3. In Section 4, we present the proposed approach and elaborate on the two use cases, namely, DDoS and IoT fingerprinting. Subsequently, we evaluate the approach and comment on the performance of each use case in Section 5. Finally, in Section 6, we revisit the contributions of this paper and offer some improvements for future work.

2. Related Literature

2.1. *P4-Enabled Analytics*

In recent years, the benefits of programmable data planes have garnered the attention of the research community. Though the ability to program these

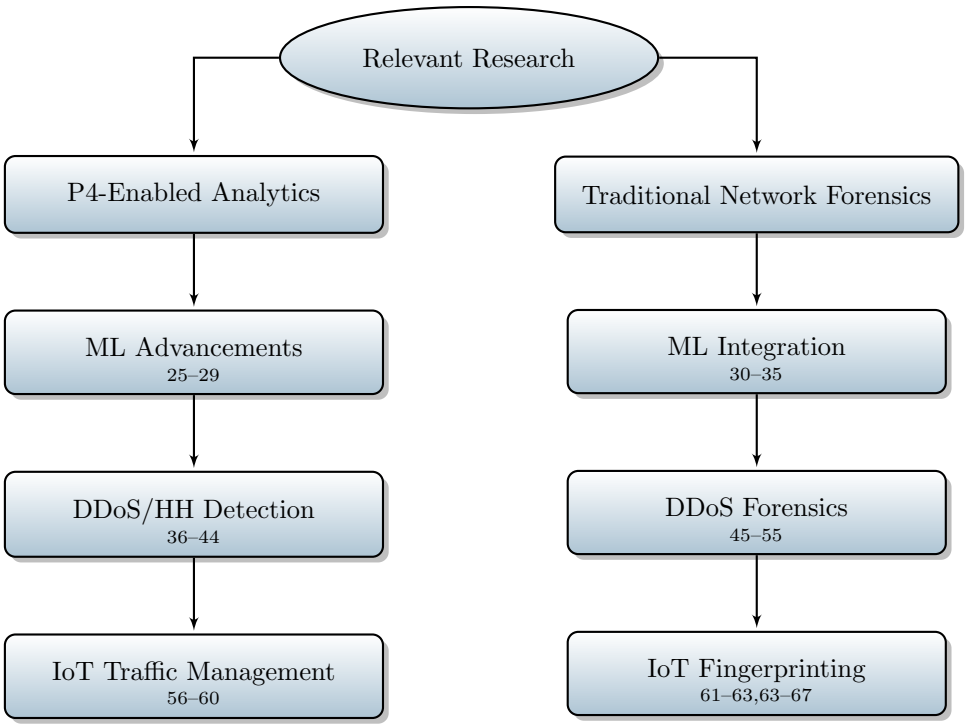


Fig. 1. Taxonomy of related literature.

forwarding devices is a relatively new technology and yet to be leveraged for tasks specific to network forensics, a number of recent research efforts have been presented to enhance network analysis procedures in the context of IoT-based measurements, addressing disproportional network flows, and enhancing machine learning implementations, as subsequently detailed and depicted in the taxonomy in Fig. 1.

Machine learning advancements. With the advantages of ML techniques becoming apparent over past decades, current research efforts have been studying how to synergize them with programmable data planes. Given that training ML models is a time consuming process that can last for weeks, traditional research avenues often endeavor to accelerate the computation process. With programmable switches, such accelerations can now be conducted throughout the network for distributed learning. To

this end, Sapio et al.[25] offered a rudimentary MapReduce application for performing data aggregation via P4 in an effort to reduce the communication overhead of exchanging model updates. In a similar context, the in-network aggregation system proposed by Yang et al.[26] was able to reduce the job completion time of a MapReduce-like framework by as much as 50%. Applying a different technique to in-network aggregation, Sapio et al.[27] used workers to perform gradiant vector computations, after which point the workers send their individual update vector to the P4 switch and receive back the aggregated model update. As a result, the authors were able to speed up the model's training by as much as 300% compared to existing distributed learning approaches. Providing an alternative for reducing processing overhead, Sanvito et al.[28] worked on analyzing options for partitioning subsets of layers of Neural Networks (NN) to offload to programmable switches and Network Interface Cards (NIC) for processing.

Another area of P4 research is the harnessing the programmable switches to perform classification tasks. This scope of study is currently still largely theoretical, though noteworthy advancements have been made. For example, Siracusano et al.[68] took a noteworthy first-step towards implementing more complex NNs in P4 via presenting a simplified NN utilizing only the bitwise logic functions that programmable switches can entertain. Additionally, Xiong and Zilberman[29] proposed some possible avenues for programming various classification algorithms in P4, namely, decision trees, k-means clustering, Support Vector Machines (SVM), and naïve Bayes. The authors' attempted to strike a balance between the limited resources the switch can use for such tasks and classification accuracy. Conversely to the bitwise logic means of model simplification leveraged by,[68] the algorithms presented by Xiong and Zilberman[29] were more complex, and the authors stated that it is uncertain as to whether these algorithms will compile on an actual hardware switch target. The proposed approach herein falls in line with the goal of the two aforementioned works of switch-based classification; however, the proposed strategy for automating the integration of rule-based classifiers entirely on the switch can be updated on-the-fly as new intelligence arrives without any downtime, and neither sacrifices accuracy nor the switch's resources.

Disproportional network flows. The generalized approach to identifying disproportional flows within a network is broadly referred to as Heavy Hitter (HH) detection. Specifically, HHs are associated with a low number of flows within a given network that consume a large amount of its bandwidth. Their swift detection has long been shown to promote effective net-

work management practices,[69–71] and has been utilized in accounting[70,72] and traffic engineering,[73,74] as well as worm and probing detection.[75,76] Following this aim, the works of Liu et al.,[36] Sivaraman et al.,[37] Xing et al.,[38] and Kučera et al.[39] extended HH detection efforts to programmable switches, which allows the traditional approach of employing software collectors residing outside the data plane to be bypassed to enhance both detection speed and accuracy. To this end, the aforementioned data plane advances have all enriched the state-of-the-art in HH identification. Ultimately, soft computing-like approaches such as HH detection, which tolerate a level of uncertainty and partial truth[77] due their generic nature, might not provide suitable evidence to implicate wrong doing in court.[78] In turn, the approach presented in this work reduces the scope of HHs to strictly DDoS detection, with the primary motivation of prompt fingerprinting for evidence extraction in order to facilitate network forensic investigations.

A number of other P4 research endeavors have also focused on DDoS detection. In particular, Zhang et al.[40] proposed a range of security policies for volumetric attack mitigation. In a different approach for addressing volumetric varieties, Lapolli et al.[41] utilized entropy for fingerprinting such traffic anomalies. In addition, Mi et al.[79] presented a deep learning technique premised upon the Pushback method[42] for tackling volumetric DDoS. To detect a particular type of volumetric DDoS, Febro et al.[43] proposed a means of fingerprinting that exploiting SIP. Alternatively, Scholz et al.[44] proposed a SYN flooding defense strategy premised upon SYN authentication and SYN cookie techniques.

While research efforts specifically tailored to DDoS fingerprinting are viable candidates for forensic procedures, if a defense mechanism is to be integrated into the network's switches, it should address all relevant attacks. This notion poses a problem for the aforementioned DDoS detection schemes as one of the caveats of the programmable switch technology is the limited resources of each switch; thus, implementing a number of different DDoS protection programs into the switch's pipeline in conjunction with fundamental programs pertaining to packet forwarding, load balancing, etc., is likely not feasible.[80] There is also a need to address the prevalence of more advanced DDoS techniques such as slow DDoS, which can circumvent the detection methods proposed in the aforementioned research efforts.[81] To this end, the work herein proposes a DDoS detection, artifact extraction, and mitigation scheme that unifies a number of techniques to function in harmony with one another in order to address an assortment of relevant DDoS attacks. Additionally, the proposed approach introduces a

novel means of providing useful forensic intelligence amid attacks employing spoofing, by way of clustering configuration artifacts on the switch. This is in contrast to previous approaches attempting to achieve this aim through source authentication techniques such a SYN cookies, which can litter the internet with the corresponding validation traffic and results in detection latency.

IoT traffic management. The IoT paradigm has unmistakably been pervasive and entrenched in contemporary society in recent years. With such an overwhelming utilization of these devices, the P4 research has focused on promoting there integration into state-of-the-art networks. One particular area of emphasis has been the significant percentage of network bandwidth that is lost while transmitting IoT packet headers. Given that these devices generally have limited processing capabilities, they typically transmit packets encapsulating small payloads (e.g., a sensor readings), which leads to large quantities of packets largely comprising redundant headers that occupy throughput and need to be processed by the network. To this extent, Wang et al.[56,57] and Lin et al.[58] proposed a promising solution of aggregating such packets on programmable switches. This is in contrast to conducting aggregation on server CPUs which can increase end-to-end latency and result in the loss of real-time functionality.

Another area of IoT research undertaken by the P4 community is service automation. Essentially, Low-power low-range IoT communication technologies characteristically utilize a Peer-to-Peer (P2P) model. While P2P offers distinct advantages such as low end-to-end latency and reduced power consumption, it's also tightly coupled with the drawbacks of subpar scalability, short reachability, and policy enforcement that is inherently inflexible. To overcome these pitfalls, Uddin et al.[59] proposed a programmable switch that automates IoT services by encoding their transactions in the data plane and utilizing the controller for address assignment, device and service discovery, subscription management, and policy enforcement. Additionally, the authors subsequently presented an extension[60] that supports multiple non-IP protocols. There is still a need to accurately fingerprint IoT devices for purposes of the aforementioned approaches and for network forensic procedures, and is thereby the motivation of IoT device fingerprinting mechanism proposed herein.

2.2. *Traditional Network Forensics*

Network forensics has safeguarded our networks for many years. Moreover, the research community has kept practictioners equiped with state-of-the-art measures for conducting effective investigations in order to hold adversaries accoutnable for their crimes. Amid some of the primary areas of study in this context are ML integration, DDoS forensics, IoT analysis, which are elaborated upon next and shown in the taxonomy in Fig. 1

ML integration. Capturing network activity lies at the root of network forensics; however, a large amount of the information captured or recorded will not be useful for investigations. Moreover, with the increasing rates of traffic modern-day networks exhibit, this equates to a large amount of wasted time, storage, and computational resources. In an effort to address this, Mukkamala and Sung[30] employed NNs and SVMs for offline intrusion analysis in order to fingerprint key features that reveal information deemed worthy for further intelligent analysis. With a similar goal, Sindhu and Meshram[31] apply the Apriori algorithm to perform association rule learning to the data their system collects in order to uncover patterns of malicious activities.

Another area of concern for practitioners has been the increased proliferation of botnets which has been causing serious security risks and financial damage. To aid the investigations of such misdemeanors, Koroniotis et al.[32] employed association rule mining, an NN, naïve Bayes, and a decision tree to detect botnets and track their activities, with the decision tree giving the best accuracy of 93.23%. In a subsequent work, Koroniotis et al.[33] facilitated the training and validation network forensic systems by way of offering a noteworthy botnet dataset. This dataset later enabled the work of Oreški and Andročec[34] which reduced the time needed for optimal feature selection by employing a genetic algorithm to optimize such parameters to be fed into an NN. In another botnet forensic undertaking, Bijalwan[35] explored the use of eight different ensembles of classifiers, showing the resultant improvement in accuracy over a single classifier. Overall, the aforesaid ML approaches brought forth advancements reducing the amount of time necessary to analyze large traffic captures for relevant artifacts. Conversely, the proposed approach conducts classifications as packets traverse the switch, which allows events to be flagged and customized actions such as the storing of evidence in the midst of an attack, in real time. Further, the presented ML-based method is automated, and thereby circumvents the need for additional expertise.

DDoS forensics. With DDoS attacks not only being a concern for decades but ever-increasing in intensity and frequency of occurrence, a number of network forensic research endeavors has been devoted to targeting this mounting issue. Following suit with the previously articulated benefits of ML, it also been leveraged for DDoS forensic tasks. One such effort is that conducted by Hoon et al.[45] which aimed to identify the best machine learning model for offline DDoS forensics, finding that naïve Bayes, gradient boosting, and distributed random forests were the most optimal. The approach taken by Kachavimath et al.[46] affirmed the effectiveness naïve Bayes and additionallly showed that k-nearest neighbors too outperforms conventional learning models. Similarly, Fadil et al.[47] utilized naïve Bayes to perform DDoS forensics on network traffic extracted from a core router via packet captures. Conversely, the proposed approach herein performs such detection as the traffic is traversing the switch. Yudhana et al.[48] also implemented a naïve Bayes classifier, however they additionally integrated a NN for conducting DDoS forensics.

Taking a more traditional approach to DDoS forensics, Zulkifli et al.[49] exercised live forensic log file analysis to identify a Denial of Service (DoS) attacks via Wireshark.[82] This live approach is in contrast to typical forensic procedures, which are executed while the system is down.[83] Another challenge for DDoS investigations has been the rise in both attack and benign traffic that networks typically observe.[45] Moreover, such a steep rise has proportionally led to a sharp growth in attack log files sizes. In an attempt to reduce the time to perform the corresponding analysis to attribute sources and victims of DDoS attacks, Khattak et al.[50] proposed using Hadoop's MapReduce. Similarly, Khattak and Anwar[51] leveraged MapReduce to parallelize the entropy-based clustering and forensic analysis of attack traffic to safeguard nodes in a cloud environment to decrease log file analysis. In building upon this aim, the proposed approach presents a technique for performing this traditionally offline procedure in a live fashion via programmable switches, which allows evidence to be obtained at line rate while the attack is simultaneously mitigated.

Additionally, Aydeger et al.[52] also worked on mitigating DDoS attacks such as Crossfire by utilizing SDN in conjunction with Network Function Virtualization (NFV) to provide a Moving Target Defense (MTD) framework for ISP networks to conceal network topologies. The authors also permitted the storing of information pertaining to potential attackers for investigations. Alternatively, the methodology introduced herein pushes relevant attack evidence to a collector for subsequent analysis immediately

upon detection to eliminate benign traffic data excessively consuming storage. P4-programmed switches process packets in nanoseconds, and allow practitioners to easily add customized code for evidence extraction once such maliciousness has been fingerprinted. Other network-specific DDoS forensic works include machine-to-machine networks presented by Wang et al.,[53] mobile ad hoc networks by Timcenko and Stojanovic,[54] and networks encompassing cellular devices by Cusack et al.[55] Further, with slow DDoS via mobile devices being a growing concern,[84] Cusack et al.[55] endeavored to idenitfy the presence of such attacks based upon the Euclidean distance similarity between the protocol (e.g., HTTP, HTTPS, etc.) counts of a past and present log file. Since this technique can lead to both false positives and negatives given the randomness of traffic patterns, the proposed approach employs a novel interarrival time analysis scheme that facilitates investigations by fingerprinting, attributing, and mitigating slow DDoS attacks in real time.

IoT fingerprinting. With the IoT paradigm being tied to a number of inherent vulnerabilities and responsible for a large number of botnet-facilitated DDoS attacks, investigating crimes conducted by way of these devices is now fundamental to network forensics. In turn, fingerprinting traffic originating from them has recently attracted significant attention from both the research community and the industry in order to identify events of interest and extract relevant artifacts. With ML's ability to recognize patterns in network traffic, it is generally leveraged for IoT fingerprinting tasks. Among these, Meidan et al.[61] used supervised learning trained upon deep packet features in order to distinguish between IoT and non-IoT devices, and to associate each IoT device to a specific class. Yang et al.[62] utilized both deep packet and header features to train a NN in order to generate IoT fingerprints. Alternatively, Sivanathan et al.[63] leveraged SDN-based, flow-level telemetry combined with machine learning for IoT classification. In another approach based upon SDN, Thangavelu et al.[64] assigned classifier maintenance to the controller and the actual tasks of classifying IoT devices to the gateways. The gateway devices utilize software for classification (i.e., unscalable to high traffic rates[63]), which differs from the hardware based classification approach proposed herein which ensures line-rate processing amid heavy traffic loads. Further, the proposed approach classifies devices from the headers of a single TCP packet which necessitates nanoseconds in hardware versus the session or flow-level analysis utilized in[64] and,[63] respectively, which gives a classification time upper-bounded by the time to analyze the encompassed successive packets. Taking a different

approach, Feng et al.[65] used IoT application-layer response data coupled with product descriptions from relevant websites to generate an Acquisitional Rule-based Engine (ARE) to classify devices. Conversely, Perdisci et al.[66] only uses DNS fingerprints IoT device classification. Lastly, Pinheiro et al.[67] five different classifiers trained with packet length statistics to identify IoT devices, from which the Random Forest algorithm achieved the highest accuracy of 96%. While all the aforesaid works advanced the state-of-the-art in IoT device fingerprinting for network forensic intelligence, all but[63] and[64] performed offline procedures which leads to delays in detection and attribution of criminal behavior, which is in stark contrast to the P4 switch-based approach which can fingerprint devices and execute corresponding customized actions as a single packet originating from the device traverses the switch's pipeline.

3. Background

3.1. *A Primer on Programmable Switches*

As previously mentioned, SDN provided an effective means of separating the control plane from the forwarding devices of the data plane. The data plane table entries are then populated by way of protocols such as OpenFlow,[85] and the control plane exposes interfaces for third-party applications where programmers can apply customized logic for the population of table entries. Despite the data plane customization that this allows for, the latest OpenFlow specification (OpenFlow 1.5.1[86]) is constrained to 45 headers, which dramatically limits the range of applications that can be used. Further, attempts to modify or add headers generally translates to about 4 years of waiting.[15]

Alternatively, recent efforts have been devoted to developing switches that allow for full data plane ASIC programmability via domain-specific languages such as P4.[87] Along with allowing for customized network implementations, programmable switches do not incur performance penalties and run on ASICs at line rate with terabit speeds. For example, the Tofino2 ASIC processes packets at 12.8 Tbps.[88]

At the root of this advance's inception is the Protocol Independent Switch Architecture (PISA), which is depicted in Fig. 2. As shown, an incoming packet enters the programmable parser where it is parsed into individual headers (parsed representation), and where states and transitions are defined. Subsequently, the packet flows sequentially into each stage of

the Programmable Match-Action Pipeline where match-action unit tables are applied. It is these tables where various header and metadata fields are typically matched on in order to provide customized behavior. Additionally, programmable data planes possess the distinct ability to perform stateful packet processing by way of storing data across packet traversals of the switch via counters and registers. As a result, network owners can leverage these storage mechanisms to implement their own complex processing logic that operates at line rate. Once the P4 program is written, it is transformed into binaries for the target architecture by the compiler provided by the particular target switch's vendor. In addition, the compilation produces interactive APIs that the control plane uses to interact with the data plane.

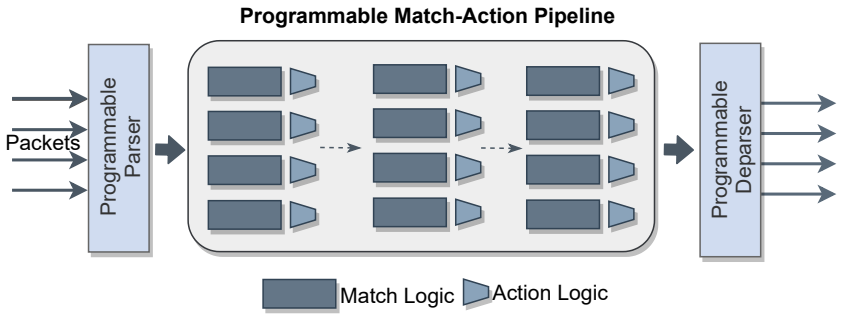


Fig. 2. Programmable switch architecture

3.2. *Motivating Line-Rate Network Forensics*

Typically, network forensics entails the storing of all observed traffic on the network via packet captures, saving sampled traffic information, or logging network events of interest. In all cases, investigations generally necessitate the later inspection of this information. Naturally, the capturing of each and every packet traversing the network has an increased potential of encapsulating forensic artifacts when present; however, this is achieved at the clear cost of storage and the time complexities associated with processing the captures. An alternative approach is conducting such analysis in an online fashion. To employ an online analytics strategy, the assistance of software-based middlebox techniques (e.g., Intrusion Detection Systems (IDS), firewalls, etc.) are generally warranted. These approaches are de-

ployed in-line, meaning that network traffic will be processed by them prior to it reaching its destination.

While middleboxes are supported by well-crafted methods and algorithms for inspecting and filtering malicious traffic, software-based solutions suffer from serious concerns in terms of performance, cost, and agility.[63] For example, DDoS attacks are now synonymous with leveraging terabits of attack traffic, which is a rate that is impossible to handle with current software solutions.[16] The end result is a significant degradation in the network's throughput, which in turn affects resource utilization. Moreover, packets inspected by software lead to a considerable increase in latency and jitter, which impacts the Quality of Service (QoS) of latency-sensitive services and user experience. Furthermore, this phenomena not only applies to DDoS attacks, as the increasing utilization of the Internet has resulted in a variety of networks experiencing exorbitant traffic loads.[89]

In addition to network performance, software-based approaches necessitate additional costs to keep up with such traffic rates. While incrementing the number of hardware resources employed will solve the problem, ultimately a steep rise in operational costs and management complexity will arise. Note that adding resources is a temporary patch given the aforementioned trend of growing traffic rates.

Lastly, proprietary middleboxes are closed source; thus, practitioners cannot readily modify algorithms or develop custom solutions that implement the latest forensic intelligence. The nature of cybercrime is dynamic, and adversaries are constantly utilizing new attack vectors and surfaces. Attempting to mitigate such maliciousness with middlebox-based techniques is a daunting task given the challenge of keeping them current without vendor support. Conversely, programmable data planes address each of the aforementioned shortcomings. They are not only cheap to deploy but allow practitioners to customize the processing logic, that once compiled, functions at line rate amid substantial traffic loads. Therefore, these forwarding devices offer the cost-effectiveness, agility, and necessary performance to meet the demands of contemporary network forensic tasks.

4. In-Network Forensic Use Cases

To effectively demonstrate the abilities of programmable switches to assist in the network forensic process, two use cases are detailed in this section in order to provide an in-network means of fingerprinting an assortment of DDoS attacks and IoT devices. The particulars of each use case are high-

lighted, along with how the intricacies of each approach are implemented on the switch.

4.1. *Assessing DDoS*

The first of the two approaches entails aggregating a number of unique DDoS detection strategies into one uniform network forensic methodology. To perform such aggregation, note that while a variety of DDoS attacks are currently exercised by adversaries, they can essentially be encapsulated by the binary classification of volumetric or stealthy (i.e., slow DDoS); thus, this proposed technique utilizes two schemes for detecting the assortment of relevant DDoS attacks, which are elaborated upon next.

4.1.1. *Slow DDoS*

What has been termed slow DDoS takes a stealthy approach to denying service to a targeted network via endeavoring to tie up the server's available connections in order to deny authentic clients access. These attacks utilize legitimate TCP behavior and send malicious packets at a frequency similar in intensity to that of benign traffic, which makes such malicious traffic incredibly hard to detect by way of traditional anomaly and signature-based techniques.[16,90]

To effectively fingerprint these stealthy attacks, the stateful processing of programmable switches is leveraged in order to track the active sessions on the server being targeted by the attacker. In particular, a record of each authentic session an outside entity retains with the target server is stored within the switch's registers. The utilization of the switch's registers (versus pushing data to the controller for storage) enables line-rate functionality as this is performed entirely on the switch hardware. Note that by maintaining such records, all assortments of TCP flooding attacks are effectively eradicated because they are not associated with any current valid connection with the server, as implied in Fig. 3. This is due to the fact that TCP flooding variations set a variety of erroneous flags without first establishing a connection (aside from SYN flooding which is addressed by the approach with a different technique), which is designed to exhaust the target server's resources, or some semblance of both.

In order to generate statistics with respect to each active session held with the server for real-time detection purposes, the switch first associates all such sessions with their corresponding source IP addresses. Note that source IPs are relevant artifacts for slow DDoS as it does not leverage

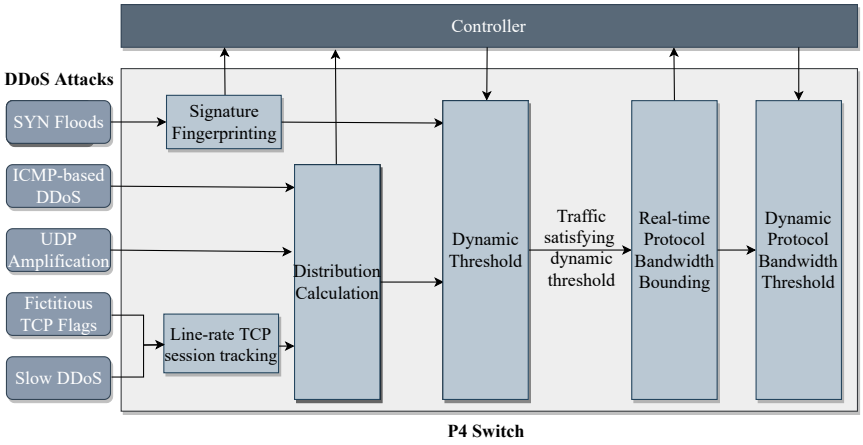


Fig. 3. DDoS detection approach overview.

spoofing because a legitimate interchange of packets between the source and destination IP addresses is fundamental to executing the attack. That being said, storing all the IP addresses that could potentially be observed naturally induces resource consumption issues. To address this issue, a 65,536 cell Bloom filter is held on the switch, which is instantiated as a register array. Bloom filters offer the distinct advantages of storage efficiency and $O(1)$ access times. In this implementation, the index of the register to be accessed in the Bloom filter is determined from the result of a Cyclic Redundancy Check 16-bit (CRC16) hash of the source IP address. The overview of the Bloom filter's behavior is shown in Fig. 4. In this instance, the registers of the Bloom filter are responsible for holding the timestamp of the last packet received from the given index. This proposed fingerprinting strategy leverages timestamps as the foundation of its detection mechanism given interarrival times are a distinguishing factor of a slow DDoS occurrence. Specifically, slow DDoS keeps interarrival times long enough as to conserve the attacker's resources and not stand out amid the flow of legitimate traffic, but not too long as to be timed-out by the target server. This behavior can be observed in the source code of implementations of this attack, such as in that of R U Dead Yet? (R.U.D.Y.)[91] and Slowloris.[92]

When a packet arrives at the switch and is found to be holding an active session with the server, the interarrival time ($\texttt{timestamp}_{\texttt{current}} - \texttt{timestamp}_{\texttt{previous}}$) of this session is extracted. Once this occurs, this value

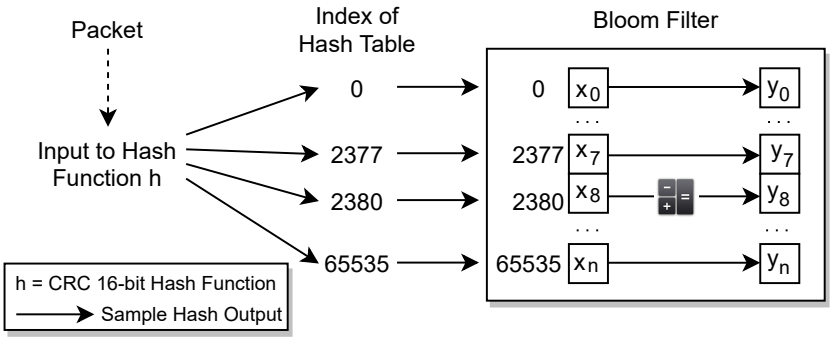


Fig. 4. In-network stateful artifact tracking by way of a Bloom filter.

is subsequently matched against ranges of interarrival times in one of the switch's match-action tables. Each of these table entries of interarrival time ranges are associated with direct P4 counters, meaning a corresponding counter is incremented for each match. At the end of each designated time window W, these counters are received by the controller as a counter array and subsequently analyzed. By way of a Python script, the controller uses these counts to formulate a distribution of the interarrival rates observed on the network during W. Note that proceeding in this fashion accounts for traffic patterns that are varying during busy or slow times. It is from this distribution that the detection strategy fingerprints anomalies associated with slow DDoS, i.e., abnormally-lengthy interarrival times. Further, given that this distribution of current network traffic is updated in real time, the switch can use its line-rate processing abilities to immediately identify such stealthy attacks.

Note that while merely employing a dynamic anomaly threshold addresses the pinpointing of slow DDoS amid varying legitimate traffic rates the network may observe, there is a chance that benign users with very slow connections could falsely be identified as malicious. In order to minimize any such impacts these users, it is paramount to impose the anomaly threshold on an as-needed basis, namely, only when the total number of connections the server has to offer are nearly all consumed. In addition, because an interarrival time calculation inherently necessitates the analysis of two subsequent packets from the same source IP, this poses a challenge if the detection mechanism waits until the all of the server's available connections are consumed before it acts, i.e., resulting in 10 seconds of DDoS for the network from an attack employing 10 second interarrival times.

As a result, it is necessary to act preemptively. To this effect, the proposed approach aims to identify the number of session establishments that can be expected to occur during W_i. The switch then strives to preemptively keep such expected number of such session establishments E open at all times during W_{i+1}. We argue that this minimizes the aforementioned false positives considering that slow DDoS interarrival times are generally much higher than that of benign users with sluggish connections. This is because if an attacker instrumented enough source IPs to consume all of the target server's available connections, using relatively-normal interrival times, the attack as a whole would lose its stealthiness as it would appear rather volumetric in nature. To incorporate this preemptive measure, the switch maintains a register holding the maximum number of session establishments which occur during time window W without any of the currently-established sessions closing. Upon the expiring of W, the controller computes the 10 second moving average of this register, est_{av}. In turn, the proposed approach can effectively identify a threshold for dropping slow DDoS connections when less than est_{av} threads exists during W, as given by:

$$\int_{thresh}^{\infty} f(x)dx = \frac{est_{av}}{\texttt{max(sessions)}}. \tag{1}$$

4.1.2. *Volumetric Analysis*

Middlebox or server-based software volumetric DDoS defenses often result in degradation to a network's throughput. This is because they simply cannot keep pace with processing the large amounts of traffic that these attacks are now generating. Conversely, the proposed volumetric detection scheme utilizes the switch's stateful storage in order to circumvent the need of such CPU-based implementations. In particular, the bandwidth artifacts utilized by TCP, UDP, and ICMP are stored within the switch's registers in Bps. This is relevant considering the direct proportionality between the bandwidth being consumed and the resource depletion of targeted server. In turn, by determining the bandwidth consumed at regular time windows W while the network is not experiencing an attack, volumetric DDoS will produce an anomaly (i.e., a deviation from the network's normal link saturation) if it transpires. These overarching bounds ($T_{allocated_i}$) are determined by assesses the expected throughput for each of the aforementioned protocols via the following equation:

$$T_{allocated_i} = B_{total} * \frac{T_{measured_i}}{\sum_{s \in (S)} T_{measured_s}}, \tag{2}$$

where B_{total} is the total amount of bandwidth allowed, $T_{allocated}$ is the throughput allocated per transport protocol, $T_{measured}$ is the current benign throughput measured by the switch during time window W, and $i \in S$ where $S = \{\text{TCP, UDP, ICMP}\}$. By bounding the protocols by $T_{allocated_i}$, the remaining protocols $S - i$ will function unimpeded if protocol i is being used to deliver a volumetric attack. Aside from the majority of TCP flooding which was previously addressed, namely, that sending fictitiously-set flags, not that this mechanism can impact service to any benign entities also using i. In turn, further action must be taken to minimize such collateral damage.

For ICMP traffic, reducing impact to its legitimate use is first promoted by not electing to adopt the approach taken by many modern-day networks of simply blocking all ICMP traffic at the edge for security purposes. The motivation for doing so is that it is often essential to circumvent issues with diagnostics and performance.[93] Secondly, such traffic is dropped when the header field Type equals 0, 3, 4, 5, 8, and several others that have been deprecated.[94] As a result, various sub-classes of ICMP-related attacks and vulnerabilities are eradicated.[95] The Bps of the remaining ICMP traffic is then recorded by the switch and bounded by $T_{allocated_{ICMP}}$.

Minimizing the impact benign entities using UDP is especially relevant. UDP encompasses a large assortment of underlying services and attempting to only bound the throughput of all UDP traffic by $T_{allocated_{UDP}}$ allows an attack using a single UDP-based service to DDoS all other UDP-encapsulated services. It is also important to note that UDP notoriously coincides with amplification attacks because some UDP-based services' response is much larger than the initial request. In turn, adversaries can easily send requests resulting in response traffic that reaches magnitudes far exceeding that which they needed to transmit. In turn, this type of attack can either consume the network's server or bombard another target with responses from the network's server via the adversaries spoofing the source IP of the initial requests (reflection attacks[96]). In fact, amplification resulted in the largest Tbps DDoS attacks to date.[97,98] To address both amplification and impact to benign entities, the proposed strategy conducts real-time distribution calculations via recording the Bps per application layer amplification protocol (identified on the switch via the destination port) over a moving average of time window W. For the sake of simplicity, the remaining application layer protocols that are associated with amplification are grouped into one distribution calculation. In the event more fine-grained consideration for such remaining protocols is needed, this tech-

nique can easily be amended with no added latency and very little additional consumption of the switches resources, namely, one extra register, counter, and port match entry per added service. Similarly to the strategy elaborated upon in Section 4.1.1, the Bps counts are extracted by the controller upon the completion of W. The controller then calculates the new threshold for each of these services and pushes them back to the switch. The switch subsequently stores these values in its registers and enforces them at line rate. Note by analyzing both the source and destination ports of incoming UDP traffic, the proposed strategy vanquishes attempts by attackers to both target or leverage (for a reflection attack) the network's server.

In terms of volumetric TCP attacks, as shown in Fig. 3, SYN flooding remains to be addressed. Given that SYN flooding generally entails spoofing of a large amount of SYN requests in order to saturate the target with empty transactions, it becomes difficult to segregate malicious request traffic from that of a benign nature. A common technique employed in the past is to merely block all SYN traffic amid such an attack, i.e., effectively denying service to all new end users as a means of mitigation. Alternative approaches have since been proposed, such as SYN cookie techniques, however they incur latency and often litter neighboring networks with response traffic.[99] To address these gaps in the literature, the proposed approach implements a signature matching scheme via hashing the headers of ingress SYN packets that have the tendency to imply different TCP/IP implementations, such as TTL, `Window_Size`, etc. This strategy is motivated by the fact that an adversary will generally target specific vulnerabilities (e.g., from a certain Operating System (OS) version[100,101]); therefore, there exists a strong likelihood that the machines exploited by this adversary to transmit the attack will possess the same signature.

The signature artifacts are maintained on the switch by of a counting Bloom filter. This Bloom filter functions similarly to that previously discussed in Section 4.1.1, however it is the configuration headers that are hashed to obtain the index of the register array, and the value stored in the given register is merely count of how many times that register has been hashed to. The highest counts within this array are stored in additional registers on the switch to be be compared against. The reason for storing multiple counts is in this event the passive signature matching procedure does not fully identify all the malicious sources; thus, blacklisting only the sources with the highest count might not be sufficient to mitigate the attack. In turn, the sources with the highest counts are incrementally

blacklisting until the SYN request rate falls below a desirable threshold in a given W. As a result, there is less likelihood that legitimate end users will inadvertently be blacklisted by the SYN flood's mitigation strategy. To calculate the aforesaid threshold dynamically, the switch first counts the SYN packets it observes during W. Upon the expiring of W, the switch's data is transmitted to the controller. At this point, the controller calculates the 10 second moving average of SYN requests and returns the resultant value (plus two standard deviations) to the switch to be used as a dynamic threshold.

4.2. *Fingerprinting IoT Devices*

With the plethora of vulnerabilities surrounding IoT couple with the increasing utilization of these devices, the value in extracting IoT-specific artifacts for investigations is evident. To date, the most effective means of fingerprinting IoT devices is by way of ML. To this end, the state-of-the-art research in P4 has been endeavoring to uncover a practical means of integrating ML functionality into the switch's pipeline.[25–27,29,68] The primary reasons for doing so are either (1) to leverage the boost in speed that the switch's hardware can offer (e.g., for distributed learning applications) or (2) to harness the classification abilities of ML within a network context. Though a few noteworthy works have been proposed addressing some nuances pertaining to (1), a viable and practical solution is still ultimately lacking in terms of (2).

One concern with (2) is whether switches can execute quantized versions of complex classification algorithms with acceptable loss to accuracy. Another debate that has arisen with (2) is such algorithms can consume a large amount of the switches limited resources and therefore, if it is realistic from an economic standpoint to have a switch strictly dedicated to classification tasks. We address these issues with (2) in this use case via identifying an ML algorithm that accurately fingerprints IoT devices without the need for any quantization and map it to the switch's pipeline in a highly efficient manner, as subsequently detailed.

4.2.1. *Switch-Based Constraints*

One of the trade-offs with leveraging the efficiency of a programmable switch is operating within its strict resource constraints. One means of meeting these tight resource bounds is by offloading tasks to the controller. That being said, such approaches can be susceptible to additional latency

due to communication and calculation delays. Whether or not this latency is acceptable is generally based on the application. Additionally, if a strict data plane application is preferred, only specific computations (e.g., simple comparisons, bitwise operations, addition, and subtraction) and a small predefined number of algorithmic operations (limited by the number of stages utilized) can be performed.[39]

While the set of computations that can be practiced within the switch is clear, a notion that cannot be understated is that of stages. Though internal switch configurations are vendor-specific and generally not disclosed to the public, it is common to employ a little over ten stages in programmable switches.[102] A stage is allocated its own dedicated resources, such as match-action tables and register arrays. Operations within a stage function independent of each other (i.e. in parallel). Though stages can pass information to subsequent stages via modifications made to a given packet's header fields and metadata, the operations encapsulated from one stage to another execute sequentially at runtime. As a result, the amount of sequential operations that a programmable switch can entertain are bounded by the number of stages the hardware switch possesses. While the choice of operation placement typically made by the compiler, it is based on whether the aforementioned operations possess dependencies (i.e., they need to be executed sequentially). For example, if $meta\_variable_1 = value_1$ and $meta\_variable_2 = meta\_variable_1 + 1$, these operations will necessitate 2 separate stages. Moreover, if intermittent stages are being filled by other P4 programs, a dependency-ridden implementation might not compile on an actual hardware switch.

In order to offer line-rate IoT artifact extraction to network forensic practitioners, the proposed IoT fingerprinting ML mechanism is converted to a resource-friendly implementation that operates entirely within the data plane. As a result, its processing is performed at a relatively constant rate as traffic traverses the switch (i.e., within nanoseconds). In particular, the Projective Adaptive Resonance Theory (PART) learning algorithm[103] is harnessed for the fingerprinting procedure. PART is a partial decision tree algorithm for rule-based classification; each rule corresponds to one traversal down the tree to a given class. Contrary to the comparable C4.5[104] and RIPPER[105] algorithms, it can generate the appropriate rules without the need to perform global optimization and hence is a more efficient alternative. Further, the proposed approach's mapping of classifier output to P4 applications can be expeditiously applied to any such rule-based approach while being extremely conservative with the aforementioned limited

resources of the switch, as subsequently elaborated upon.

4.2.2. *Meeting Hardware Restrictions*

With the proposed fingerprinting approach residing strictly within the data plane, it is paramount that it meets the aforementioned tight bounds of such implementations. To address this aim, we convert the rules generated by PART to match-action tables in the switch's pipeline. Essentially, each of these rules encompasses a group of comparison statements, with the number of statements within each rule i falling within the set $S_i = \{x \mid x \in \mathbb{N} \wedge x <= |\mathbf{f}|\}$, where $|\mathbf{f}|$ denotes the number of features used for the classification. It can be observed in Fig. 5 how each of these rules arrives at a particular class.

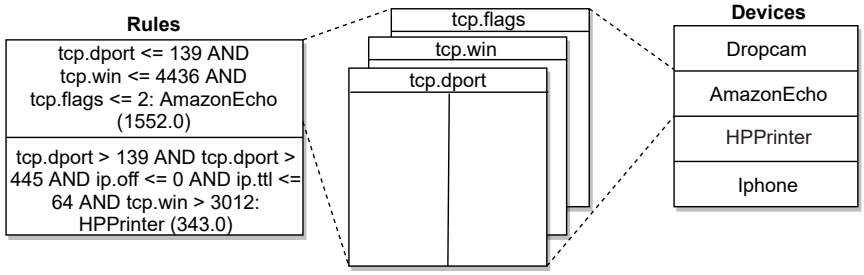


Fig. 5. Match-action tables corresponding to each feature

4.2.3. *P4-Specific Features*

With resource conservation in mind, recall that the programmable parser extracts header values prior to the match-action pipeline. This translates to packet header data being stored in variables (i.e., so they can be processed by the switch in its pipeline) without using any of the valuable switch stages for assignment operations. To leverage this notion, the PART algorithm is trained specifically only upon headers in order to determine which are best utilized for IoT device fingerprinting. The headers instrumented as features are denoted in Table 1. As shown, there are ten features in total (i.e., $|\mathbf{f}| = 10$).

Feature List		
ip.len	ip.id	ip.off
ip.ttl	ip.sum	tcp.sport
tcp.dport	tcp._off	
tcp.win	tcp.flags	

Table 1. IP and TCP header-based features used for fingerprinting

4.2.4. *Parallel Processing*

Building upon the strict employment of header-based features for classification, it should be noted that the variables that the programmable parser stores the header values in are entirely independent of one another. In turn, all features can be evaluated in parallel because there are no dependencies between them, as previously explained in Section 4.2.1. Further, evaluating whether each of these features match an explicit range or value requires no other operations other than hard coding the values to be matched against as keys in the switch's match-action pipeline. As a result, proper implementation of the PART feature evaluation component of the P4 program facilitates parallel processing, and in turn, uses a minimal number of consumed stages.

4.2.5. *Match Table Mapping*

To facilitate the generation of a program that can be updated by network operators upon the arrival of any new fingerprinting intelligence, whether during initialization or runtime, the program must be constructed in such a manner where this needed flexibility exists strictly within the entries of the match-action tables. This is because while the entries in the match-action tables can be added or removed effortlessly at any point during the program's execution, the allocation of the tables took place during the program's compilation and are therefore static. To this end, the proposed fingerprinting approach employs a shell made entirely up of tables, i.e., the actual P4 code that is visible to the forensic practitioner and will not be modified. This shell is only dependent upon the features utilized (i.e., the header values trained on). The shell encompasses one table per feature, followed by a single fingerprinting hash table to perform the classification. Each of these tables are instantiated via a simple apply statement, as shown by Alg. 1.

In turn, the number of tables implemented is always equal to $|\mathbf{f}| + 1$.

Algorithm 1 P4 implemenation algorithm.

```
Control Ingress {
    apply(ip_len_tbl);
    apply(ip_id_tbl);
    apply(ip_off_tbl);
    apply(ip_ttl_tbl);
    apply(ip_sum_tbl);
    apply(tcp_sport_tbl);
    apply(tcp_dport_tbl);
    apply(tcp_off_tbl);
    apply(tcp_flags_tbl);
    apply(tcp_win_tbl);
    apply(hash_fingerprinting_tbl);
}
```

Further, because the feature analysis can be conducted in parallel, the IoT fingerprinting approach only necessitates two stages in the programmable switch pipeline, namely, the feature tables followed by the device classification. Each of the feature tables has a declared `action()` (i.e., performs processing based on the key that was matched) that merely assigns a result to a single metadata variable which holds that table's match result (i.e., the result of that feature's evaluation). Each result falls within the set $T_{result} = \{j \mid j \in \mathbb{N} \wedge j <= |K|\}$, where K is the ranges a particular feature must be evaluated against. Effectively, each feature matching result can be thought of as one feature check in the rules (such as that depicted in Fig. 5) that, when grouped together for all features, correspond to a classification. Thus, once the features to be evaluated for a given classification task have been decided upon, it is only the match ranges (i.e., the tables keys) that can vary after retraining the PART algorithm for this implementation. In turn, the proposed fingerprinting approach for translating rule-based classification schemes to P4 programs can be generalized to any rule-based classification task, given any additional functionality incorporated by practitioners falls within the switch's resource constraints.

4.2.6. *Device Fingerprinting*

Once the feature tables are applied, the remaining stage consists of fingerprinting the device based upon the aforementioned results of the feature tables. With these results being held in 10 respective feature metadata vari-

ables, device fingerprinting is achieved by performing a single hash of the aggregation of the returned feature table results. In turn, the output of the hash acts a unique device identification (ID), which acts as its fingerprint for forensic investigations. Further, the textual representation of the specific device can be obtained by matching the device ID against the unique hashes held by the controller, which can be immediately computed after the PART algorithm has generated the rules from its training. Moreover, specific behaviors can be defined on the switch to take the appropriate measures when the hash of a given type of device's packets traverse the switch. This entire process of fingerprinting the IoT device necessitates exactly one traversal of a SYN packet, which the switch can compute in nanoseconds.

4.2.7. *Automating Program Configuration*

Mapping the rule-based ML classifier output into the switch's match-action tables can be achieved by way of the controller upon switch initialization, or at any time during execution. The method for automating this strategy is depicted in Fig. 6. The underlying motivation for proceeding in an automated fashion is twofold: (1) the ML mechanism can be leveraged by practitioners regardless of background expertise and (2) the implementation allows for the freedom of updating the model while the approach is running or offline. As shown in Fig. 6, the rule-based classifier feeds its trained rules to the control program. These rules are then processed by a Python script residing on the controller. The script's pseudocode is articulated in Alg. 2. By way of the compiler-generated API for interacting with the switch, the controller first removes the existing entries in the switches tables. Subsequently, the controller repopulates the tables with the updated entries by adding the output from Alg. 2. The match-action tables in the switch pipeline shown in Fig. 6 perform ternary matching in order to determine if a given number falls within a specific range. As shown in Alg. 2, the lower k and upper $k+1$ bounds corresponding to each feature j for a given rule are stored in $\mathbf{T}_{matrix}[j,k]$. These bounds are sorted to act as a division of the ranges held by K.

5. Evaluation

To assess the effectiveness of both the DDoS detection and IoT fingerprinting network forensic use cases, an evaluation of both was conducted and detailed in this section. With the proposed DDoS network forensic de-

Algorithm 2 Converting classifier output to P4 table entries.

Result: P4Runtime table population commands

$l \leftarrow rule\_list$
$\mathbf{T}_{matrix} \leftarrow [|l| * 2, |\vec{f}|]$
$j \leftarrow 0$
while f *in* $\vec{f}$ **do**
  $k \leftarrow 0$
  while *rule in* l **do**
    $n \leftarrow count(devices)$
    if f **then**
      $\mathbf{T}_{matrix}[j,k] \leftarrow bound_{lower}$
      $\mathbf{T}_{matrix}[j,k+1] \leftarrow bound_{upper}$
      $k \leftarrow k+2$
    end
  end
end
while *row* r *in* $\boldsymbol{T}_{matrix}$ **do**
  $sort(\mathbf{T}_{matrix}[r,])$
end
while *column* c*, row* r *in* $\boldsymbol{T}_{matrix}$ **do**
  write ternary range $\mathbf{T}_{matrix}[r,c]$ *to feature table* $f[c]$
end

tection and mitigation use case tackling the broad spectrum of currently relevant attacks, three of such scenarios were employed that generalize well to contemporary DDoS attack vectors, namely, SYN flooding, UDP amplification, and Slow DDoS. Alternatively, the IoT fingerprinting use case necessitated a single scenario to measure its ability to fingerprint such devices entirely within the switch amid a stream of traffic passing through it.

5.1. *Environmental Setup*

The experimental topology shown in Fig. 7 was employed to evaluate the DDoS detection and mitigation use case and was implemented on Mininet[106] in conjunction with the BMv2 software switch. The underlying OS utilized was Ubuntu 16.04.6 LTS, with 16GB of memory and eight Intel Xeon Gold 6130 CPUs running at 2.10GHz. Fig. 7 also denotes the six clients which were connected to the P4-programmed software switch by

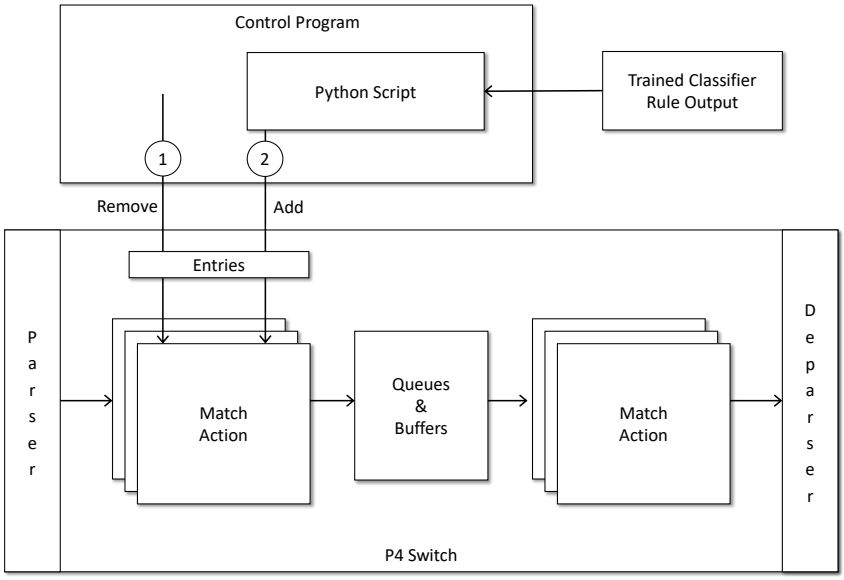


Fig. 6. Automation of IoT device fingerprinting.

way of Linux network namespaces. These client machines were responsible for generating traffic towards the target server. The artifacts serving as the experimental results were extracted by the switch via P4 and polled by the controller for analysis.

5.2. *DDoS Detection Results*

UDP amplification. For this scenario, traffic utilizing two common protocols leveraged by adversaries, DNS and NTP, were transmitted to Client 4 which was configured to emulate a resolver to perform the reflection against the target server. The requests it served were spoofed to the server's IP, and it responded with an amplification factor consistent with.[107] The maximum allowable UDP bandwidth was set to 300 Mbps, and the `hping3` Linux tool was used to produce the traffic from the respective client machines. As depicted in Fig. 7, Client 1 and Client 2 transmitted benign DNS and NTP traffic, respectively, at approximately 714 datagrams/sec to the server to establish the baseline rates measured by the switch. To alternatively represent a network environment entertaining a variety of UDP protocols that do

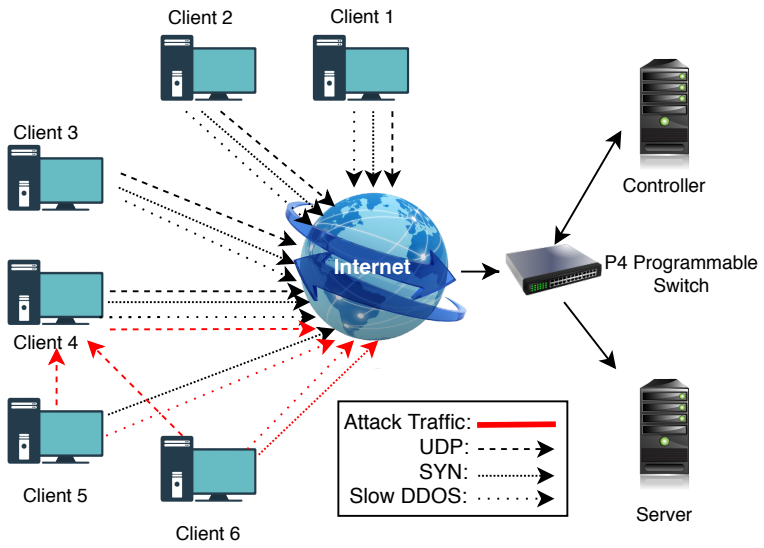


Fig. 7. The topology of the DDoS network forensics evaluation environment

not promote amplification, Client 3 produced such legitimate traffic toward the server at a rate of 2,856 datagrams/sec. After an arbitrary 10 seconds that passed for baseline establishment, Client 5 began flooding Client 4 with spoofed DNS traffic as shown at second 1 in Table 2. Note that by second 2 of Table 2, while the available DNS throughput had been fully consumed, neither NTP nor any of the other UDP protocols were affected by the attack. Moreover, these unaffected protocols could increase throughput unimpeded amid the DNS amplification attack. In a similar fashion, Client 6 launched a concurrent amplification attack at second 3 of the use case utilizing NTP. As shown in Table 2, the remaining UDP protocols traversed the network unimpeded. In fact, they encountered no latency or packet drops while the network was under the aforementioned two amplification attacks simultaneously, and were able to effectively double their transmission rates collectively to 5712 datagrams/sec towards the server unaffected, which was the maximum amount of bandwidth allotted during the baseline establishment.

SYN Flooding. For the SYN flooding scenario, a SYN queue size of 1024 was assumed and `hping3` was again leveraged for traffic generation.

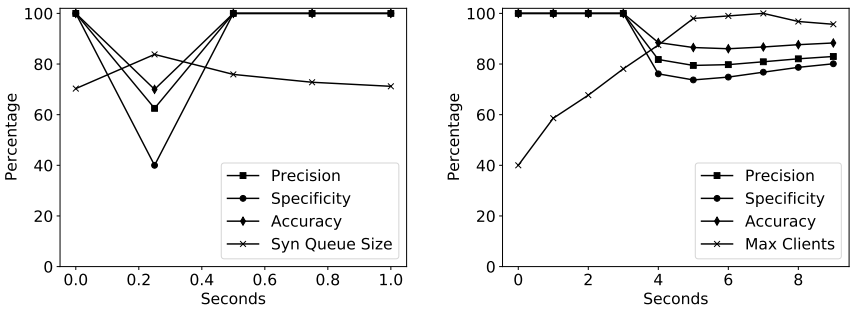
Protocols	# Packets per Second					Avg Bounds
NTP Avail.	553	562	557	0	0	1252
NTP Utilized	698	688	694	1251	1254	
DNS Avail.	557	0	0	0	0	1254
DNS Utilized	693	1257	1254	1254	1254	
Misc. Avail.	2068	2053	2067	2094	2029	4626
Misc. Utilized	2582	2547	2582	2555	2554	
	1s	2s	3s	4s	5s	

Table 2. UDP Amplification attack mitigation results

Amid the arbitrary baseline establishment period of 10 seconds, Clients 1 through 5 each made 600 SYN requests per second to the server with subsequent ACKs. At second 0 of Fig. 8a, Client 6 began transmitting malicious SYN requests peaking at approximately 2000 packets/sec. With the six client machines in Fig. 7 utilizing the same OS environment, their configurations were modified to mimic the diversity in real-world settings in order to effectively evaluate the proposed approach's signature attribution mechanism. A binary classification of the transmitted traffic was conducted by the switch to perform the evaluation, with the negative class encompassing malicious traffic and the positive class pertaining to that with legitimate intent. In turn, the metrics of true positives (TP), false negatives (FN), true negatives (TN), and false positives (FP) were leveraged to obtain the $specificity = {}^{TN}/_{(TN+FP)}$, $precision = {}^{TP}/_{(TP+FP)}$, and $accuracy = {}^{(TP+TN)}/_{(TP+TN+FP+FN)}$ of the results, as shown in Fig. 8a. The dip in performance peaked at approximately the 0.25 second mark. Note that this 0.25 second window needed to observe the signature deviation is proportional to the traffic rate deviation from the dynamic threshold. That being said, a flawless performance was given by all metrics by 0.5 seconds. Additionally, no latency was observed in the services of legitimate SYN requests throughout the attack, and the SYN queue never exceeded its conservative size of 1024 set forth for evaluation purposes. In fact, the occupancy of the SYN queue only increased by roughly 13% due to the attack traffic. Additionally, it should be noted that this experiment's artifacts extracted by the switch at line rate were polled by the controller at arbitrary 0.25 second intervals for convenience, and thus the aforementioned 0.5 second interval in this case is an upper bound of a complete mitigation.

Slow DDoS. To better put the scheme's full capabilities to the test, the maximum number of clients that the server can entertain concurrently was set to a conservative 256, which thereby necessitates a smaller margin for error. To emulate realistic TCP traffic interarrival times, the transmission rates employed for this evaluation scenario were typical fast transmissions (t_{b_fast}), slower traffic originating from sparsely-connected regions (t_{b_slow}), and the lengthier interarrival times of slow DDoS (t_m).

Specifically, the aforementioned rates are defined as rational numbers $\mathbb{Q}$ and fall within the ranges $t_{b\_fast} \in \mathbb{Q}(0.00 < t_{b\_fast} \leq 1.00)$, $t_{b\_slow} \in \mathbb{Q}(1.00 < t_{b\_slow} \leq 2.50)$, and $t_m \in \mathbb{Q}(1.75 < t_m \leq 5.00)$. Note the existence of $S_{intersect} = \{t_{intersect} \mid t_{intersect} = t_{b\_slow} \wedge t_{intersect} = t_m\}$ which further challenges the performance of the proposed approach given $|S_{intersect}|$ is much greater than would be exhibited in a real-world setting. Once more harnessing `hping3` for traffic generation and following



(a) Syn Flood (b) Slow DDoS

Fig. 8. SYN flooding and Slow DDoS mitigation results.

the blueprint laid out in Fig. 7, 50 connections were each established by Clients 1 through 3 with the server, which resulted in 58.59% of the it's connection limit being occupied, as depicted at second 1 of Fig. 8b. Following this first wave of transmissions, 50 additional threads were then consumed by Client 4 with interarrival times encompassed by t_{b_slow}, as given by 78.13% of that available shown at second 3 of Fig. 8b. At approximately second 4, Clients 5 and 6 then began generating slow DDoS traffic leveraging the interarrival times encapsulated by t_m with the aim of overtaking 128 connections each.

The malicious requests resulted in 22.27% of the server's maximum number of connections initially being exhausted at second 5 by the attack, as

displayed in Fig. 8b, and ultimately increased to 24.61% two seconds later. At this point, 193, or 75.39% of the malicious sources were successfully fingerprinted and subsequently denied by the switch. Furthermore, the switch dropped 11 more slow DDoS connections by second 10, which was in fact the dynamic value calculated for number of legitimate requests the switch can expect to receive consecutively (i.e., est_{av}) per time window W in this use case. Ultimately, the amount of the legitimate sessions that did not lose service never extended below 96.5%, and it only took two seconds from session exhaustion to open up est_{av} connections for new users; thus, the automatic retransmissions of the 3.5% (7) of benign clients that temporarily lost their connections would have granted them service again with negligible latency.

5.3. *IoT Fingerprinting Assessment*

5.3.1. *Dataset Selection*

With forensic investigations necessitating fine-grained artifact extraction, this use case should not only be evaluated on its binary classification of IoT versus non-IoT, but it ability to fingerprint the specific device itself. To perform this evaluation strategy, the network traffic captures taken by Sivanathan et al.[24] were instrumented for training and testing, given its variety of encapsulated devices. Specifically, the devices and the amount of packets each utilized for training are listed in Table 3. In addition, the proposed approach was trained to fingerprint the source devices using artifacts from a single TCP packet. Consequently, note that for the interarrival times i corresponding to a specific source device d and the time taken for fingerprinting a device $t_{fingerprint}$, the source to output a given packet t_{source}, and the switch to process a given packet t_{switch}, approaches necessitating the analysis of 5 consecutive packets for fingerprinting devices would require time $t_{fingerprint_d} = \sum_{x=1}^{4}(i_{d_x} + t_{source_x} + t_{switch_x})$, versus $t_{fingerprint_d} = t_{switch_x}$ for the proposed use case.

5.3.2. *Execution*

To arrive at the PART model, the first 50,000 packets associated with a given device in Table 3 were extracted until 1 million packets were reached. The associated IP and TCP headers of the aforesaid packets were subsequently placed in a CSV file for training of the PART model via the Weka tool.[108] The resultant 851 rules were then written to a text file that was

parsed by the Python script elaborated upon in Alg. 2, which resided on the controller for table entry population. Subsequently, another 1 million packets were transmitted through the P4-programmed switch by way of `tcpreplay`.

Index	**Device Name**	**#Packets**	**Type**
a	SmartThings	50,000	IoT
b	AmazonEcho	50,000	IoT
c	NetatmoWelcome	50,000	IoT
d	TP-LinkDayNightCloudCamera	50,000	IoT
e	SamsungSmartCam	50,000	IoT
f	Dropcam	50,000	IoT
g	InsteonCamera	50,000	IoT
h	WithingsSmartBabyMonitor	50,000	IoT
i	BelkinWemoSwitch	50,000	IoT
j	TP-LinkSmartPlug	15,301	IoT
k	iHome	22,820	IoT
l	BelkinWemoMotionSensor	50,000	IoT
m	NESTProtectSmokeAlarm	1,430	IoT
n	NetatmoWeatherStation	16,760	IoT
o	WithingsSmartScale	1,923	IoT
p	BlipcareBloodPressureMeter	90	IoT
q	WithingsAuraSmartSleepSensor	50,000	IoT
r	LightBulbsLiFXSmartBulb	26,523	IoT
s	TribySpeaker	49,401	IoT
t	PIX-StarPhotoFrame	16,228	IoT
u	HPPrinter	38,596	IoT
v	SamsungGalaxyTablet	50,000	NIoT
w	NestDropcam	31,818	IoT
x	Windows Laptop	50,000	NIoT
y	MacBook	50,000	NIoT
z	AndroidPhone	20,156	NIoT
aa	Iphone	8,954	NIoT
ab	TPLinkRouterBridgeLAN	50,000	NIoT

Table 3. IoT and NIoT devices in the dataset

5.3.3. *Results*

Every fingerprint the switch made was immediately pushed to the controller in order to aggregate the results, which are visualized in the confusion matrix heatmap depicted in Fig. 9.

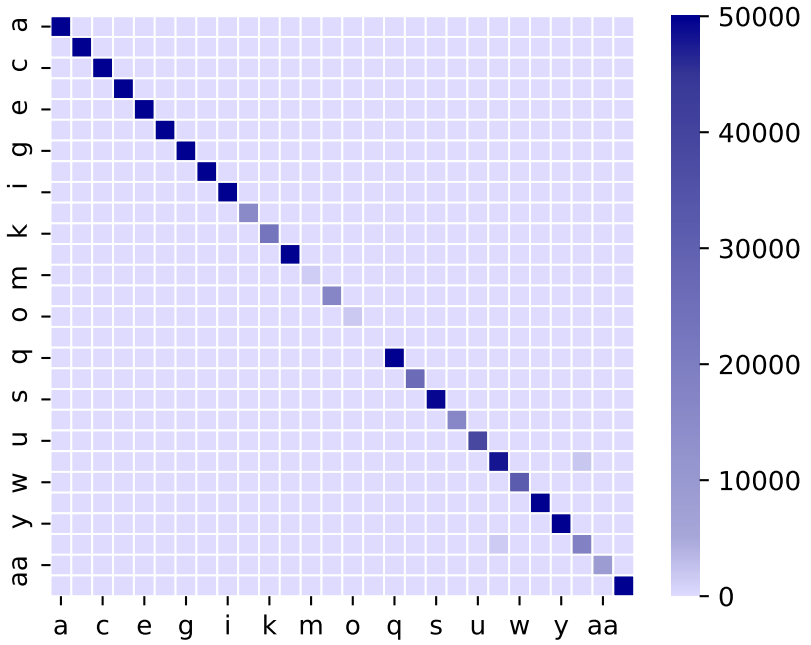


Fig. 9. Confusion matrix of the IoT fingerprinting results

By taking into account the indices of the devices in Table 3 and their corresponding classification rate, the highest misclassification rate was given by the Android phone of approximately 8.886%, with roughly 8.499% of its records being wrongly fingerprinted as the Samsung Galaxy tablet. Similarly, the Samsung Galaxy tablet had the second highest misclassification rate of precisely 4.16% with exactly 4.072% of its packets being fingerprinted as the Android phone. Additionally, it can be observed in Fig. 9 that the devices associated with higher misclassification rates map to those with less training samples in 3. As a result, this suggests that per-device misclassification rates can be reduced merely by incorporating more samples for these devices. The overall accuracy of the proposed approach is

portrayed in Table 4. As shown, approximately 99.581% and 0.419% of the 1 million packets were correctly and incorrectly fingerprinted as being IoT devices, respectively.

	IoT	**Non-IoT**	**Total**
Correct	99.9668 %	98.2825 %	99.5809 %
Incorrect	0.0332 %	1.7175 %	0.4191 %
Total	770,890	229,110	1,000,000

Table 4. The accuracy of the proposed IoT fingerprinting approach.

6. Conclusion and Future Directions

Network forensics has long been used for fingerprinting network events and extracting important artifacts for investigation purposes. Nevertheless, traditional forensic procedures will continue to suffer from latency and poor incident response as long as they fail to keep pace with the current technology trends. To this end, we proposed the transformation of network forensic procedures into that functioning at line rate by leveraging the newfound programmable switch technology. In turn, we presented two use cases applicable to major areas of concern within the network forensic community. The first use case remediates DDoS attacks by employing dynamic thresholds from line-rate artifact extraction offered by the switch to infer contemporary DDoS in real time. The empirical results confirm that the approach efficiently mitigates both UDP amplification and SYN flood attacks, and significantly reduces the remediation time of slow DDoS. The second use case facilitates forensic investigations connected to the vulnerable IoT paradigm via employing a rule-based PART learning algorithm on the switch in order to accurately fingerprint the origin device from a single TCP packet, at line rate. Further, the IoT fingerprinting mechanism was automated to translate the output of rule-based learning algorithms to P4 programs. The results show that the approach can fingerprint devices with 99% accuracy. We are optimistic that the proposed approaches will promote the utilization of programmable switches in a range of network forensics procedure, in addition to that presented. Moreover, we anticipate our procedure for automating the integration of rule-based classifiers into the data plane will inspire a number of other switch-based ML advance-

ments within the forensics community.

For future work, both use cases will be deployed on actual hardware switches; while BMv2 has been widely utilized for P4 development, it can not offer the precise resource utilization assessments and fine-grained nanosecond-level measurements that a hardware switch can. Secondly, exploring further avenues to compact ML classifier output to P4 code would facilitate additional advancements within both the network forensic and P4 communities. To this effect, subsequent endeavors will explore the integration of other ML algorithms within programmable data planes.

7. Acknowledgment

This material is based on research sponsored by the Department of Homeland Security (DHS), United States Secret Service, National Computer Forensics Institute (NCFI) via contract number 70US0920D70090004.

References

1. C. Fachkha, E. Bou-Harb, and M. Debbabi. Towards a forecasting model for distributed denial of service activities. In *2013 IEEE 12th International Symposium on Network Computing and Applications*, pp. 110–117 (2013).
2. E. Bou-Harb, M. Debbabi, and C. Assi, Cyber scanning: a comprehensive survey, *Ieee communications surveys & tutorials*. **16**(3), 1496–1519 (2013).
3. E. Bou-Harb, W. Lucia, N. Forti, S. Weerakkody, N. Ghani, and B. Sinopoli, Cyber meets control: A novel federated approach for resilient cps leveraging real cyber threat intelligence, *IEEE Communications Magazine*. **55**(5), 198–204 (2017).
4. M. Husák, J. Komárková, E. Bou-Harb, and P. Čeleda, Survey of attack projection, prediction, and forecasting in cyber security, *IEEE Communications Surveys & Tutorials*. **21**(1), 640–660 (2018).
5. Interpol. Cybercrime: Covid-19 impact. URL `https://www.interpol.int/en/content/download/15526/file/COVID-19CybercrimeAnalysisReport-August2020.pdf` (Aug, 2020).
6. S. Khan, A. Gani, A. W. A. Wahab, M. Shiraz, and I. Ahmad, Network forensics: Review, taxonomy, and open challenges, *Journal of Network and Computer Applications*. **66**, 214–235 (2016).
7. K. Kaur, S. Garg, G. Kaddoum, E. Bou-Harb, and K.-K. R. Choo, A big data-enabled consolidated framework for energy efficient software defined data centers in iot setups, *IEEE Transactions on Industrial Informatics*. **16** (4), 2687–2697 (2019).
8. S. Ranger. Github hit with the largest ddos attack ever seen (2018).
9. S. Ikeda. Iot-based ddos attacks are growing and making use of common vulnerabilities. URL `https://www.cpomagazine.com/cyber-security/iot-`

based-ddos-attacks-are-growing-and-making-use-of-common-vulnerabilities/ (Apr, 2020).
10. S. Soltan, P. Mittal, and H. V. Poor. Blackiot: Iot botnet of high wattage devices can disrupt the power grid. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 15–32 (2018).
11. C. Fachkha, E. Bou-Harb, and M. Debbabi. Fingerprinting internet dns amplification ddos activities. In *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5 (2014).
12. C. Fachkha, E. Bou-Harb, A. Keliris, N. D. Memon, and M. Ahamad. Internet-scale probing of cps: Inference, characterization and orchestration analysis. In *NDSS* (2017).
13. T. Enderle and U. Bauknecht. Modeling dynamic traffic demand behavior in telecommunication networks. In *Photonic Networks; 19th ITG-Symposium*, pp. 1–8 (2018).
14. J. Crichigno, E. Bou-Harb, and N. Ghani, A comprehensive tutorial on science dmz, *IEEE Communications Surveys & Tutorials*. **21**(2), 2041–2078 (2018).
15. N. McKeown and J. Rexford. Clarifying the differences between p4 and openflow. URL `https://p4.org/p4/clarifying-the-differences-between-p4-and-openflow.html` (May, 2016).
16. K. Friday, E. Kfoury, E. Bou-Harb, and J. Crichigno. Towards a unified in-network ddos detection and mitigation strategy. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pp. 218–226 (2020).
17. p4lang/behavioral-model. URL `https://github.com/p4lang/behavioral-model` .
18. E. Bou-Harb, M. Debbabi, and C. Assi. Behavioral analytics for inferring large-scale orchestrated probing events. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 506–511 (2014).
19. E. Bou-Harb, M. Debbabi, and C. Assi, A novel cyber security capability: Inferring internet-scale infections by correlating malware and probing activities, *Computer Networks*. **94**, 327–343 (2016).
20. F. Shaikh, E. Bou-Harb, N. Neshenko, A. P. Wright, and N. Ghani, Internet of malicious things: Correlating active and passive measurements for inferring and characterizing internet-scale unsolicited iot devices, *IEEE Communications Magazine*. **56**(9), 170–177 (2018).
21. S. Torabi, E. Bou-Harb, C. Assi, M. Galluscio, A. Boukhtouta, and M. Debbabi. Inferring, characterizing, and investigating internet-scale malicious iot device activities: A network telescope perspective. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 562–573 (2018).
22. N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, Demystifying iot security: an exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations, *IEEE Communications Surveys & Tutorials*. **21**(3), 2702–2733 (2019).
23. M. S. Pour, E. Bou-Harb, K. Varma, N. Neshenko, D. A. Pados, and K.-

K. R. Choo, Comprehending the iot cyber threat landscape: A data dimensionality reduction technique to infer and characterize internet-scale iot probing campaigns, *Digital Investigation.* **28**, S40–S49 (2019).
24. A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, Classifying iot devices in smart environments using network traffic characteristics, *IEEE Transactions on Mobile Computing.* **18**(8), 1745–1759 (2018).
25. A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis. In-network computation is a dumb idea whose time has come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pp. 150–156 (2017).
26. F. Yang, Z. Wang, X. Ma, G. Yuan, and X. An, Switchagg: A further step towards in-network computation, *arXiv preprint arXiv:1904.04024* (2019).
27. Sapio and et al., Scaling distributed machine learning with in-network aggregation, *arXiv preprint arXiv:1903.06701* (2019).
28. D. Sanvito, G. Siracusano, and R. Bifulco. Can the network be the ai accelerator? In *Proceedings of the 2018 Morning Workshop on In-Network Computing*, pp. 20–25 (2018).
29. Z. Xiong and N. Zilberman. Do switches dream of machine learning? toward in-network classification. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pp. 25–33 (2019).
30. S. Mukkamala and A. H. Sung, Identifying significant features for network forensic analysis using artificial intelligent techniques, *International Journal of digital evidence.* **1**(4), 1–17 (2003).
31. K. Sindhu and B. Meshram, Digital forensics and cyber crime datamining (2012).
32. N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay. Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques. In *International Conference on Mobile Networks and Management*, pp. 30–44 (2017).
33. N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset, *Future Generation Computer Systems.* **100**, 779–796 (2019).
34. D. Oreški and D. Andročec. Genetic algorithm and artificial neural network for network forensic analytics. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 1200–1205 .
35. A. Bijalwan, Botnet forensic analysis using machine learning, *Security and Communication Networks.* **2020** (2020).
36. Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 101–114 (2016).
37. V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford. Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*, pp. 164–176 (2017).
38. J. Xing, W. Wu, and A. Chen. Architecting programmable data plane de-

fenses into the network with fastflex. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pp. 161–169 (2019).
39. Kučera and et al. Enabling event-triggered data plane monitoring. In *Proceedings of the Symposium on SDN Research*, pp. 14–26 (2020).
40. Zhang and et al. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In *Proceedings of NDSS* (2020).
41. Lapolli and et al. Offloading real-time ddos attack detection to programmable data planes. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 19–27 (2019).
42. J. Ioannidis and S. M. Bellovin, Implementing pushback: Router-based defense against ddos attacks (2002).
43. A. Febro, H. Xiao, and J. Spring. Distributed sip ddos defense with p4. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–8 (2019).
44. Scholz and et al., Me love (syn-) cookies: Syn flood mitigation in programmable data planes, *arXiv preprint arXiv:2003.03221* (2020).
45. K. S. Hoon, K. C. Yeo, S. Azam, B. Shunmugam, and F. De Boer. Critical review of machine learning approaches to apply big data analytics in ddos forensics. In *2018 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5 (2018).
46. A. V. Kachavimath, S. V. Nazare, and S. S. Akki. Distributed denial of service attack detection using naïve bayes and k-nearest neighbor for network forensics. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pp. 711–717 (2020).
47. A. Fadlil, I. Riadi, and S. Aji, Review of detection ddos attack detection using naive bayes classifier for network forensics, *Bulletin of Electrical Engineering and Informatics*. **6**(2), 140–148 (2017).
48. A. Yudhana, I. Riadi, and F. Ridho, Ddos classification using neural network and naïve bayes methods for network forensics, *International Journal of Advanced Computer Science and Applications*. **9**(11), 177–183 (2018).
49. M. A. Zulkifli and U. Dahlan, Live forensics method for analysis denial of service (dos) attack on routerboard, *Int. J. Comput. Appl.* **180**(35), 23–30 (2018).
50. R. Khattak, S. Bano, S. Hussain, and Z. Anwar. Dofur: Ddos forensics using mapreduce. In *2011 Frontiers of Information Technology*, pp. 117–120 (2011).
51. R. Khattak and Z. Anwar. D3tac: Utilizing distributed computing for ddos attack traffic analysis on the cloud. In *2016 19th International Multi-Topic Conference (INMIC)*, pp. 1–6 (2016).
52. A. Aydeger, N. Saputro, and K. Akkaya, A moving target defense and network forensics framework for isp networks using sdn and nfv, *Future Generation Computer Systems*. **94**, 496–509 (2019).
53. K. Wang, M. Du, Y. Sun, A. Vinel, and Y. Zhang, Attack detection and distributed forensics in machine-to-machine networks, *IEEE Network*. **30** (6), 49–55 (2016).
54. V. Timcenko and M. Stojanovic. Application of forensic analysis for intru-

sion detection against ddos attacks in mobile ad hoc networks. In *Proceedings of the 1st WSEAS Int. Conf. on Information Technology and Computer Networks (ITCN'12), Vienna* (2012).
55. B. Cusack, R. Lutui, and R. Khaleghparast. Detecting slow ddos attacks on mobile devices. In *The 27th Australasian Conference on Information Systems* (2016).
56. S.-Y. Wang, C.-M. Wu, Y.-B. Lin, and C.-C. Huang, High-speed data-plane packet aggregation and disaggregation by p4 switches, *Journal of Network and Computer Applications*. **142**, 98–110 (2019).
57. S.-Y. Wang, J.-Y. Li, and Y.-B. Lin, Aggregating and disaggregating packets with various sizes of payload in p4 switches at 100 gbps line rate, *Journal of Network and Computer Applications*. p. 102676 (2020).
58. Y.-B. Lin, S.-Y. Wang, C.-C. Huang, and C.-M. Wu, The sdn approach for the aggregation/disaggregation of sensor data, *Sensors*. **18**(7), 2025 (2018).
59. M. Uddin, S. Mukherjee, H. Chang, and T. Lakshman. Sdn-based service automation for iot. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pp. 1–10 (2017).
60. M. Uddin, S. Mukherjee, H. Chang, and T. Lakshman, Sdn-based multi-protocol edge switching for iot service automation, *IEEE Journal on Selected Areas in Communications*. **36**(12), 2775–2786 (2018).
61. Meidan and et al. Profiliot: a machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*, pp. 506–509 (2017).
62. K. Yang, Q. Li, and L. Sun, Towards automatic fingerprinting of iot devices in the cyberspace, *Computer Networks*. **148**, 318–327 (2019).
63. A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, Managing iot cyber-security using programmable telemetry and machine learning, *IEEE Transactions on Network and Service Management*. **17**(1), 60–74 (2020).
64. Thangavelu and et al., Deft: A distributed iot fingerprinting technique, *IEEE Internet of Things Journal*. **6**(1), 940–952 (2018).
65. X. Feng, Q. Li, H. Wang, and L. Sun. Acquisitional rule-based engine for discovering internet-of-things devices. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 327–341 (2018).
66. Perdisci and et al., Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis .
67. A. J. Pinheiro, J. d. M. Bezerra, C. A. Burgardt, and D. R. Campelo, Identifying iot devices and events based on packet length from encrypted traffic, *Computer Communications*. **144**, 8–17 (2019).
68. G. Siracusano and R. Bifulco, In-network neural networks, *arXiv preprint arXiv:1801.05731* (2018).
69. A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: Scaling flow management for high-performance networks. In *Proceedings of the ACM SIGCOMM 2011 conference*, pp. 254–265 (2011).
70. C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of the 2002 conference on Applications, technolo-*

gies, architectures, and protocols for computer communications, pp. 323–336 (2002).
71. L. Jose, M. Yu, and J. Rexford. Online measurement of large traffic aggregates on commodity switches. In *Hot-ICE* (2011).
72. N. Duffield, C. Lund, and M. Thorup. Charging from sampled network usage. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 245–256 (2001).
73. T. Benson, A. Anand, A. Akella, and M. Zhang. Microte: Fine grained traffic engineering for data centers. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, pp. 1–12 (2011).
74. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, Deriving traffic demands for operational ip networks: Methodology and experience, *IEEE/ACM Transactions On Networking*. **9**(3), 265–279 (2001).
75. S. Venkataraman, D. Song, P. B. Gibbons, and A. Blum. New streaming algorithms for fast detection of superspreaders. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE (2004).
76. Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang. Worm origin identification using random moonwalks. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pp. 242–256 (2005).
77. D. Ibrahim, An overview of soft computing, *Procedia Computer Science*. **102**, 34–38 (2016).
78. S. Khan, A. Gani, A. W. A. Wahab, A. Abdelaziz, K. Ko, M. K. Khan, and M. Guizani, Software-defined network forensics: Motivation, potential locations, requirements, and challenges, *IEEE Network*. **30**(6), 6–13 (2016).
79. Y. Mi and A. Wang. Ml-pushback: Machine learning based pushback defense against ddos. In *Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies*, pp. 80–81 (2019).
80. E. F. Kfoury, J. Crichigno, and E. Bou-Harb. Offloading media traffic to programmable data plane switches. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7 (2020).
81. D. Anstee, Preparing for tomorrow's threat landscape, *Network Security*. **2015**(8), 18–20 (2015).
82. Wireshark. URL `https://www.wireshark.org/` .
83. M. I. Mazdadi, I. Riadi, and A. Luthfi, Live forensics on routeros using api services to investigate network attacks, *International Journal of Computer Science and Information Security (IJCSIS)*. **15**(2) (2017).
84. E. Cambiaso, G. Papaleo, and M. Aiello. Taxonomy of slow dos attacks to web applications. In *International Conference on Security in Computer Networks and Distributed Systems*, pp. 195–204 (2012).
85. McKeown and et al., Openflow: enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review*. **38**(2), 69–74 (2008).
86. Nygren and et al., Openflow switch specification version 1.5. 1, *Open Networking Foundation, Tech. Rep* (2015).
87. P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, et al., P4: Program-

ming protocol-independent packet processors, *ACM SIGCOMM Computer Communication Review*. **44**(3), 87–95 (2014).
88. B. Networks. Tofino 2: Barefoot. URL https://www.barefootnetworks.com/products/brief-tofino-2/ .
89. Cisco annual internet report - cisco annual internet report (2018–2023) white paper. URL https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (Mar, 2020).
90. K. Hong, Y. Kim, H. Choi, and J. Park, Sdn-assisted slow http ddos attack defense method, *IEEE Communications Letters*. **22**(4), 688–691 (2017).
91. DanielRTeixeira. R.u.d.y. URL https://github.com/DanielRTeixeira/R.U.D.Y. (Dec, 2016).
92. Gkbrk. Slowloris. URL https://github.com/gkbrk/slowloris/blob/master/slowloris.py .
93. K. Scarfone and P. Hoffman, Guidelines on firewalls and firewall policy, *NIST Special Publication*. **800**, 41 (2009).
94. Iana. URL https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml .
95. Z. Trabelsi, S. Zeidan, and K. Hayawi, Denial of firewalling attacks (dof): The case study of the emerging blacknurse attack, *IEEE Access*. **7**, 61596–61609 (2019).
96. C. Fachkha, E. Bou-Harb, and M. Debbabi, Inferring distributed reflection denial of service attacks from darknet, *Computer Communications*. **62**, 59–71 (2015).
97. C. Morales. Netscout arbor confirms 1.7 tbps ddos attack; the terabit attack era is upon us. URL https://www.netscout.com/blog/asert/netscout-arbor-confirms-17-tbps-ddos-attack-terabit-attack-era (Mar, 2018).
98. C. Cimpanu. Aws said it mitigated a 2.3 tbps ddos attack, the largest ever. URL https://www.zdnet.com/article/aws-said-it-mitigated-a-2-3-tbps-ddos-attack-the-largest-ever/ (Jun, 2020).
99. S. Shin, V. Yegneswaran, P. Porras, and G. Gu. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 413–424 (2013).
100. M. S. Pour, A. Mangino, K. Friday, M. Rathbun, E. Bou-Harb, F. Iqbal, K. Shaban, and A. Erradi. Data-driven curation, learning and analysis for inferring evolving iot botnets in the wild. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pp. 1–10 (2019).
101. M. S. Pour, A. Mangino, K. Friday, M. Rathbun, E. Bou-Harb, F. Iqbal, S. Samtani, J. Crichigno, and N. Ghani, On data-driven curation, learning, and analysis for inferring evolving internet-of-things (iot) botnets in the wild, *Computers & Security*. **91**, 101707 (2020).
102. B. Networks. Copyright © 2017 -barefoot networksprogrammable data plane at terabit speeds. URL https://p4.org/assets/p4_d2_2017_

programmable_data_plane_at_terabit_speeds.pdf .
103. E. Frank and I. H. Witten, Generating accurate rule sets without global optimization (1998).
104. J. Quinlan, *C4. 5: programs for machine learning*. Elsevier (2014).
105. W. W. Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pp. 115–123. Elsevier (1995).
106. N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 253–264 (2012).
107. M. Kührer, T. Hupperich, C. Rossow, and T. Holz. Exit from hell? reducing the impact of amplification ddos attacks. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pp. 111–125 (2014).
108. Weka. URL https://www.cs.waikato.ac.nz/ml/weka/ .