P4DDPI: Securing P4-Programmable Data Plane Networks via DNS Deep Packet Inspection

Ali AlSabeh*, Elie Kfoury*, Jorge Crichigno*, Elias Bou-Harb†
*Integrated Information Technology Dept., University of South Carolina (USC), Columbia, South Carolina, USA
†The Cyber Center For Security and Analytics, Information Systems and Cyber Security Dept.
University of Texas at San Antonio (UTSA), San Antonio, Texas, USA
Email: *aalsabeh@email.sc.edu, *ekfoury@email.sc.edu, *jcrichigno@cec.sc.edu, †elias.bouharb@utsa.edu

***Abstract*—One of the main roles of the Domain Name System (DNS) is to map domain names to IP addresses. Despite the importance of this function, DNS traffic often passes without being analyzed, thus making the DNS a center of attacks that keep evolving and growing. Software-based mitigation approaches and dedicated state-of-the-art firewalls c a n become a bottleneck and are subject to saturation attacks, especially in high-speed networks. The emerging P4-programmable data plane can implement a variety of network security mitigation approaches at high-speed rates without disrupting legitimate traffic.**

This paper describes a system that relies on programmable switches and their stateful processing capabilities to parse and analyze DNS traffic solely in the data plane, and subsequently apply security policies on domains according to the network administrator. In particular, Deep Packet Inspection (DPI) is leveraged to extract the domain name consisting of any number of labels and hence, apply filtering r ules (e.g., b locking malicious domains). Evaluation results show that the proposed approach can parse more domain labels than any state-of-the-art P4-based approach. Additionally, a significant performance gain is attained when comparing it to a traditional software firewall -pfsense-, in terms of throughput, delay, and packet loss. The resources occupied by the implemented P4 program are minimal, which allows for more security functionalities to be added.

***Index Terms*—P4-programmable switches, stateful processing, high-speed networks, DNS filtering, DPI.**

I. Introduction

The Domain Name System (DNS) [1] is a hierarchical distributed database that was initially implemented to map human-readable domain names (e.g., google.com) to machine-readable Internet Protocol (IP) addresses (e.g., 8.8.8.8) [2]. Later, the DNS became an essential part of the Internet providing various services (e.g., host and mail server aliasing, load distribution, etc.). Since its conception, attacks on the DNS grew widely and wreaked havoc in numerous domains. Recent large-scale attacks, such as the Mirai botnet that affected millions of users and exceeded 600 Gigabits per second (Gbps) in volume [3], use the DNS as a main attack vector [2]. The security gap incurred by the DNS can be attributed to its ability in handling DNS records transparently, i.e., DNS should not attempt to interpret nor understand the records it is serving. While such transparency is essential for a fast and smooth deployment of new technologies without altering the infrastructure, it leaves the Internet prone to a wide variety of attacks [4].

Traditional enterprise networks use a number of components and approaches to protect against security threats. For example, they could use Intrusion Detection/Prevention Systems (IDS/IPS), Access Control Lists (ACLs) to filter out unwanted traffic (e.g., based on the IP address), and firewalls to detect application layer attacks (e.g., detect malicious payload) [5]. As the security measures are distributed among different components, the process of implementing enterprise network security policy becomes tedious and challenging. On the other hand, placing the security in one component, such as deploying a firewall at the edge to protect against all external threats, has multiple disadvantages, such as degrading the throughput, making the component a single point of failure, as well as leaving the network vulnerable to internal attacks [6].

Current architectures adopting Software-defined Networking (SDN)/OpenFlow offload attack mitigation to the control plane that operates at significantly lower speeds than the data plane. Additionally, the OpenFlow protocol is restricted and can act on a standardized set of header fields [7]. Such burdens push the cybersecurity community to adopt a solution that can swiftly deploy threat mitigation without disrupting legitimate traffic.

The advent of the P4 language [8] and programmable devices allows network operators to describe in the software (i.e., the P4 program) the behavior of how packets are processed. Such flexibility removes the entry barrier to network design, previously reserved to chip manufacturers, and spurs innovation in the data plane. The flexibility offered by the P4 language allows network designers to overcome the restrictions of OpenFlow and implement their own customized data plane based on the requirement of the network. Since the emergence of the P4 language, several network applications were offloaded to the data plane, thus, significantly enhancing the performance of the network. Despite the increasing number of network applications being offloaded to the data plane, Deep Packet Inspection (DPI) is not widely practiced by P4

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2022
28 April 2022, San Diego, CA, USA
ISBN 1-891562-78-9
https://dx.doi.org/10.14722/madweb.2022.23012
www.ndss-symposium.org

researchers as they are reluctant that it might overwhelm the switch and degrade the throughput in the network [9].

A. Contribution

This paper leverages P4 for Domain name DPI (P4DDPI), in particular, to parse and filter domains in the data plane without using the control plane. The proposed scheme is motivated by the ever-growing demand for moving and processing data (e.g., filtering malicious domains, prohibiting Command and Control (C&C) domain access), which burdens the communication and computing infrastructure. The contribution of the paper is summarized as follows:

- A P4-based approach to extract the domain name from DNS queries with any number of labels via DPI.
- Protecting networks by filtering malicious domains taken from a well-known public dataset.
- The implemented prototype shows significant improvements in the throughput, delay, and packet loss rate when compared to Central Processing Unit (CPU)-based approaches.
- The resources occupied by the P4 program are negligible, allowing other security and networking features to be implemented.

II. Related Work

This section highlights two categories in the context of security implementations with emphasis on those pertaining to DNS, in addition to DPI in P4. Furthermore, this section pinpoints the novelty of this paper compared to the literature.

A. DNS and security implementations in P4

Meta4 [10] is a framework that monitors network traffic by parsing the domain name in DNS replies. Essentially, Meta4 captures all type A DNS replies in the data plane, parses the domain name and the response IP address (IP address of the server hosting the domain), and the IP address of the client requesting the domain. The resulting flow identifier (client IP, server IP, domain name) is stored in the data plane, and all corresponding traffic belonging to this flow is monitored in the data plane using P4 registers. Meta4 is implemented on a P4 Tofino switch and analyzed on campus traffic, where it achieved high accuracy in terms of traffic volume measurement by the domain name. Meta4 is restricted to parsing only four labels of the domain name (total of 60 bytes); thus, queries with long malicious domains are easily bypassed.

WORD [11] is a data plane approach that targets DNS water torture attack, which overloads a specific domain (e.g., example.com) with an enormous number of DNS requests, each with a random subdomain (e.g., a.example.com, b.example.com, etc.). WORD analyzes per-domain DNS requests and replies and is composed of three main parts that (1) splits the subdomain and the domain of arriving DNS packets; (2) counts the number of Non-existent Domain (NXDOMAIN) responses; and (3) maintains an approximate distinct count of the number of subdomains for each domain. WORD hashes the top 3 labels only and uses a public dataset of Top Level Domains (TLDs) to split the domain and the subdomain. Consequently, WORD may not generalize well for unknown domains not found in the dataset.

P4DNS [12] is an in-network cache for storing DNS entries. Evaluations of P4DNS show that it outperforms other CPU-based counterparts, where P4DNS has a significantly higher throughput and lower latency. P4DNS stresses the limitations of DNS implementations in P4, such as the variable header length parsing and the need for loops to parse DNS names.

Other security schemes in P4 include mitigation against Distributed Denial of Service (DDoS) attacks [13], DNS amplification attacks, and policy-based firewalls [9]. However, none of the approaches perform DPI to extract DNS names.

B. Deep Packet Inspection (DPI) in P4

Jepsen et al. [14] develop a system for locating the occurrences of string keywords in the payload using P4. The PISA-based Parallel Search (PPS) takes a set of search patterns and converts it to a partitioned Deterministic Finite Automata (DFA) that can run on each stage, thus, achieving parallelism across pipelines. Inspired by the use of recirculation for DPI, P4DDPI utilizes this concept for performing domain name extraction and applying it in security fields.

DeepMatch [15] provides a line-rate DPI primitive in the data plane using Netronome SMART. DeepMatch showcases new applications in the data plane that were not realized before, such as Quality of Service (QoS) policies and network monitoring that require processing application layer information. Additionally, IDS can be integrated to perform advanced DPI security policies. DeepMatch provides stateless intra-packet and stateful inter-packet regex matching capabilities, as well as supports reordering of packets since the content specified by a regex may appear anywhere in a flow.

To the best of the authors' knowledge, the proposed approach herein is the first to exploit DPI in the programmable data plane for enhancing the security of the network at the level of the DNS. This opens new opportunities for security implementations in the data plane beyond the traditional header fields (i.e., TCP/IP).

III. Background

A. The Domain Name System (DNS)

The DNS is a hierarchical decentralized system that maps the physical location (i.e., IP address) of a service and its logical address (i.e., its domain name) so that a service can be reached through its name. Domain names are organized as a suffix tree structure called domain namespace. Each node (hierarchy) in the tree has an associated label, and the dot character is used to separate hierarchies. Each label is a length octet followed by an octet string. The farthest right label is named the Top Level Domain (TLD), such as ".com" in "www.example.com". All domain names end at the root, which has a null string label, thus, the length of the last label is zero [1, 16].

DNS records stored in DNS servers hold crucial information about a domain and a record can be of different types

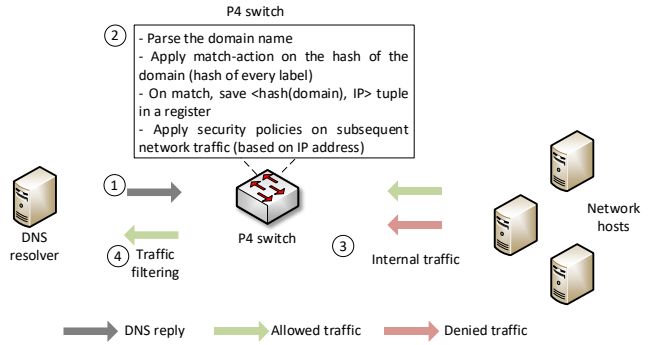


Fig. 1: Proposed system.

(Resource Record (RR)). For instance, RR type A stores a hostname and its corresponding IPv4 address, while RR type AA stores the IPv6 address. Other common types include Canonical Name (CNAME) record for aliasing hostnames, Mail eXchanger (MX) record for mailing services, Name Server (NS) for specifying a DNS zone.

B. P4 language

Programming Protocol-independent Packet Processors (P4) is a domain-specific programming language for network devices that defines how packets are processed in the data plane devices (e.g., switches, routers, Network Interface Cards (NICs), etc.). The programmable forwarding data plane is an evolution of the SDN paradigm, that was earlier restricted to the OpenFlow protocol [17]. Since its conception, P4 has been leveraged in multiple research areas, most notably in In-band Network Telemetry (INT), load balancing, network performance, congestion control, security, etc. In network security, robust solutions, such as Poseidon [18], have been proposed to mitigate DDoS attacks using programmable devices. In congestion control, Kfoury et al. [19] proposed a P4-based method to automate end-hosts' TCP pacing. P4 switches are designed to process Terabits per second (Tbps) of data. Thus, several limitations accompany them, such as the flexibility of the P4 language, the memory in the switch, the processing complexity, etc. For instance, DPI is not widely practiced in P4 and very few papers discuss it.

IV. Proposed System

A. Overview

The proposed architecture advocates for implementing security policies requiring DPI in the data plane. In particular, policies that act on packets occurring frequently in the network (e.g., DNS queries). Typically, security policies requiring deep analysis on non-traditional fields (e.g., DNS, HTTP/HTTPs, FTP, SMPT, etc.) are implemented in next-generation firewalls. However, this complexity comes at the expense of throughput degradation and resource consumption. The proposed approach focuses on inspecting DNS type A query due to its ubiquitous use in the Internet as the first step to access websites, domains, etc.

The high-level architecture of the proposed approach shown in Fig. 1 can be summarized in the following steps. (1) The DNS resolver replies with a DNS response to a previously sent DNS query. (2) The P4 switch intercepts the DNS response and parses all the labels in the domain name using packet recirculation. Furthermore, the P4 switch stores the IP address of the domain name (the IP address of the client can also be stored and used depending on the policy) in a register. (3) Subsequent packets having the destination IP address of the domain are matched in the data plane (using P4 registers). (4) The security policy (e.g., blocking a malicious domain) is enforced at line-rate without involving the controller. For the proposed architecture to be highly effective, the placement of the programmable switch should be carefully chosen to intercept all DNS queries from the hosts to the DNS resolver/cache.

While parsing a variable length header is supported in P4, its operations are limited; thus, to parse domain name labels with variable size, a state is created for each supported length. Since the resources in the parser are limited, the implemented P4 program parses up to 19 characters (i.e., 19 bytes $\times$ 8 = 152 bits) in each label, where labels with longer lengths can be forwarded to the controller for further processing. Ideally, the matching process should occur on each label as it is in the domain; however, the maximum length of a domain can go up to 253 bytes [20] and several labels can exist, which can exhaust the switch. In order to efficiently match against domain names, the hash of the label is used for matching instead of the label.

In one pipeline pass, the P4 program parses up to four labels; however, many domains, especially malicious ones, are long and have more than four labels [21]. To accommodate such domains, packet recirculation is leveraged in P4. The pseudocode of the P4 program is summarized in Algorithm 1. First, the parser parses DNS packets (i.e., UDP packets having source or destination ports 53) and the first four labels of the domain name. In the Switch Ingress control, the CRC_{32} hash of every parsed label is computed. If more labels exist in the DNS packets, then a recirculation flag is set so that the packet gets recirculated at the end of the egress pipeline. Additionally, the concatenation of the hashes is calculated, hashed, and sent in a customized header with the recirculated packet.

The purpose of recirculation is to parse the remaining labels in the domain. To do that, the currently parsed domain label headers (i.e., per-pipeline pass) are set to invalid (i.e., removed from the packet). The DNS packet keeps recirculating in the pipeline until all domain labels are parsed. The last recirculated packet contains a concatenation hash (representing the labels in previous recirculations), as well as the hashes of the labels corresponding to the last recirculated packet. Eventually, when recirculation is no longer needed, the P4 program parses the IP address of the domain (DNS response type A) and stores it in a register for various security policies to be applied.

Such an approach is conceived to support matching on all the labels of the domain without exhausting the resources of the switch.

Algorithm 1 Pseudocode of P4DD

```
Parser():
    Parse_traditional_headers(
    label1 ← pkt.extract(p.doma
    label2 ← pkt.extract(p.doma
    label3 ← pkt.extract(p.doma
    label4 ← pkt.extract(p.doma
SwitchIngress():
    table domain_name_table
        key : cnct_h; label1_h; ..; la
        actions : send; drop; modi
    label1_h = hash(label1)
    label2_h = hash(label2)
    label3_h = hash(label3)
    label4_h = hash(label4)
    if not_recirculated_packet then
        cnct_h = 0
    if len(label_5) > 0 then
        recirculate_flag = 1
        cnct_h = hash(cnct_h, label1_h, ..., label4_h)
        set_invalid(label1, label2, label3, label4)
    else
        if domain_name_table.hit() then
            register.add(domain_IP)
```

V. Implementation and Evaluation

To evaluate the proposed approach, two types of experiments were conducted. First, P4DDPI is compared with pfsense, a well-known open-source firewall that is used in thousands of enterprises as well as in research [22]. Pfsense software distribution is compatible with most hardware supported by FreeBSD. It is a full-fledged firewall that encompasses various functionalities, such as routing and forwarding, IDS, IPS, load balancing, GeoIP blocking, etc. Second, P4DDPI is compared with Meta4 [10] to evaluate the number of domains that can be parsed from a list of open-source blacklisted domains [21].

Tests configuration and used metrics. Fig. 2 shows the topology used to conduct experiments. Each of pfsense, host H1, and host H2 runs solely on a physical server equipped with 48 Xeon 6130 cores operating at 2.1 GHz, and 189 GB of Random Access Memory (RAM). To emulate DNS inspection in pfsense, DNSBlocking from the pfBlocker package is loaded, in addition to routing between the two hosts in the topology. Hosts H1 and H2 are used to generate DNS and background traffic. Edgecore Wedge100BF-32X [23] is the programmable switch used in the experiment to load the P4 program (e.g., P4DDPI and Meta4). All the links connecting the devices have a 40 Gbps rate. When evaluating P4DDPI, P4 rules are inserted so that the traffic does not pass through pfsense. On the other hand, to evaluate pfsense, the switch will act as a simple L3 program and all the traffic will pass through pfsense.

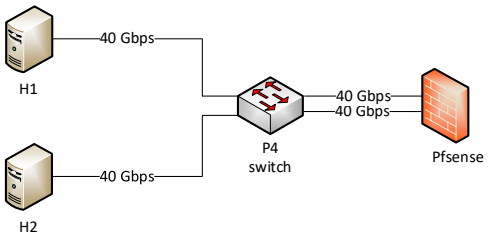


Fig. 2: Experimental evaluation topology.

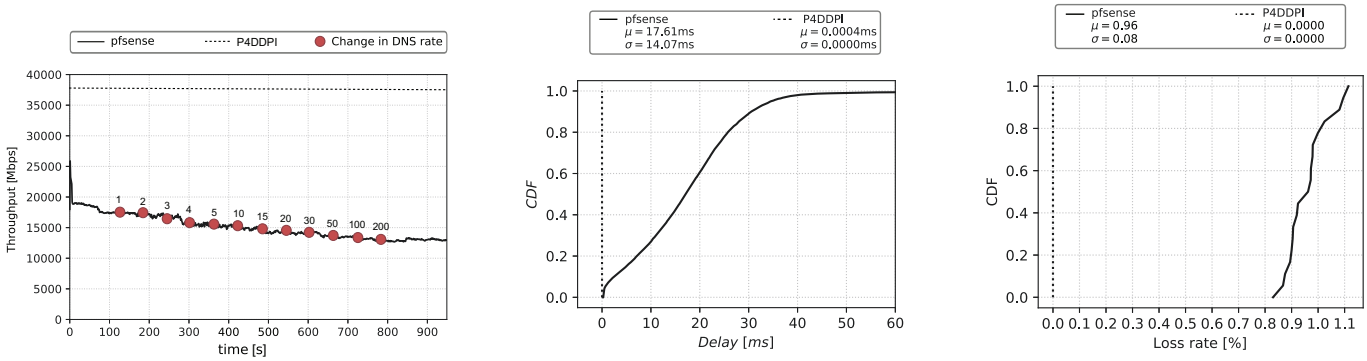
To generate background traffic, 500 iPerf [24] tests run between hosts H1 and H2 to utilize the bandwidth. The TCP send and receive buffers of the end-hosts were set to a large value (i.e., 200 MB) so that they are not the bottleneck in the performed experiments. Furthermore, small delays were introduced between the iPerf tests to prevent TCP global synchronization. dns-flood tool [25] and Token Bucket Filter (tbf) [26] were used to generate DNS traffic with various rates. The generated DNS packets contain malicious domain names derived from a public dataset [21], and rules were inserted in the switch/firewall to ensure that the whole domain name is being parsed.

The conducted experiments include performance measures in (1) throughput (the actual rate of packets successfully transferred from the sender to the receiver); (2) delay (the time interval a DNS query spends in the switch and in pfsense); (3) packet loss (the percentage of packets that failed to reach the destination); and (4) CPU usage of pfsense (measured under sending DNS traffic with various rates).

A. Throughput

The conducted throughput experiments aim to study the effect of deep packet inspecting DNS queries (under normal and heavy DNS rates) on the throughput of the network. Google allows anyone including, Internet Service Providers (ISPs) and large organizations, to use its public DNS servers free of charge under specific rate limits. In particular, each client IP address should not exceed 1500 queries per second (qps) [27], where the DNS query length is normally between 50 to 550 bytes [28]. Based on these metrics, the authors considered that DNS rates up to 6 Megabits per second (Mbps), derived from 550*8*1500, could be safely considered normal.

Fig. 3a shows the throughput of the generated background traffic using iPerf while sending DNS queries with various traffic rates. For the first 120 seconds, only the background traffic was generated without any DNS queries. At t = 120 s, host H1 started sending DNS queries with a 1 Mbps rate. The rate of the DNS queries incrementally increases every 60 s to reach 200 Mbps at the end of the test. Under pfsense, the throughput of the iPerf tests drops from approximately 18 Gbps (no DNS traffic) to 12 Gbps under 200 Mbps DNS rate. Note that the throughput starts to degrade under normal DNS rates (<= 6 Mbps), meaning that enabling DNS inspection in pfsense can easily affect the performance of the network



(a) (b) (c)

Fig. 3: Throughput, delay, and packet loss comparison between pfsense and P4DDPI.

under normal traffic. As for the throughput under P4DDPI, no performance penalty was observed under any DNS traffic rate and the throughput remained constant at approximately 37 Gbps; thus, utilizing the whole bandwidth.

B. Delay

To measure the delay in the P4 switch, DNS response messages are sent from host H1 to host H2. For every DNS message, the P4 switch adds the timestamps on ingress (port facing host H1) and egress (port facing host h2) and sends them to the controller via message digests. To measure the delay in pfsense, DNS request messages are sent from host H1 to pfsense, which in turn sends a DNS reply with the same transaction ID. The P4 switch logs the timestamp at the ingress ports facing host H1 and pfsense, along with DNS transaction ID, and sends them to the control plane to measure the delay.

To guarantee that the delay measured only corresponds to the processing time in pfsense and not to the Round-Trip Time (RTT) between pfsense and an external DNS server, the sent DNS requests contain domain names that should be blocked by pfsense. Thus, once pfsense parses the domain name of a query, it will send a DNS reply with the IP address of the DNS sinkhole address without forwarding to an external DNS resolver.

Fig. 3b shows the Cumulative Distribution Function (CDF) of the delay observed during the experiments in both scenarios. The CDF was formed with the dataset that constitutes the delay measurements for all DNS packets in P4DDPI and pfsense when background traffic was generated.

The delay incurred in packets when they are processed in pfsense ranges from 0.207 milliseconds (ms) to 455 ms with a mean (μ) of 17.61 ms and a standard deviation (σ) of 14.07ms. Approximately, 57% of the packets experienced a delay above 15ms, 40% have delays above 20ms, and 1% have delays above 50ms. Such a delay is attributed to the CPU-based hardware component on which pfsense is running. The delay experienced in P4DDPI remains almost constant at a few hundred of nanoseconds.

C. Packet loss

Relying on the reports generated by iPerf to measure the loss in pfsense or the P4 switch does not produce precise results, since the loss could occur from the server or the client sides (hosts H1 and H2). To accurately measure the loss incurred by the middlebox, the P4 switch is utilized to count the number of incoming and outgoing packets at the ingress and egress ports.

Fig. 3c shows the CDF of packet loss observed in the background traffic (i.e., iPerf) while generating DNS traffic with the same rates applied in the throughput experiments (Section V.A). The loss rate reported in pfsense approximately ranges between 0.84 and 1.1. On the other hand, no packet loss was observed in P4DDPI under varying DNS rates. As for the dropped DNS packets, the authors noticed that pfsense drops a significant number of DNS queries especially when the rate is high.

D. CPU usage

Fig. 4 shows the CPU utilization of pfsense while incrementally changing the DNS traffic rates. The CPU utilization increases exponentially to reach 100% at 50 Mbps DNS rate. This explains the drop in throughput and packets, as well as the increasing delay in pfsense. On the other hand, in P4DDPI the DNS packets are handled in the Application-Specific Integrated Circuit (ASIC), which can process Tbps; thus, no performance penalty is incurred.

E. Supported domains and resources used

Meta4 [10] is the closest state-of-the-art approach to P4DDPI. However, the P4 program only supports 4 labels in the domain name, which is not sufficient for acting as a security middlebox. Shalla blacklist dataset [21] was analyzed, and P4DDPI was compared against Meta4 based on the number of domains that are supported. The dataset is public and contains a collection of URL lists grouped into several categories. P4DDPI misses 4% of the analyzed blacklist domains, while Meta4 misses 17%. The domains missed in P4DDPI belong to those that have labels exceeding 19 characters; however,

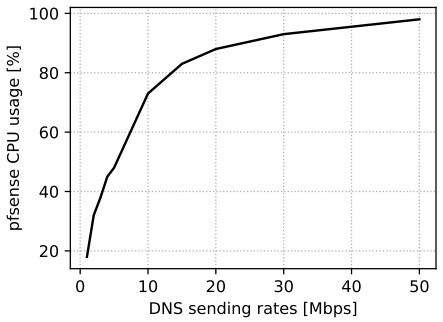


Fig. 4: CPU usage of pfsense under different DN

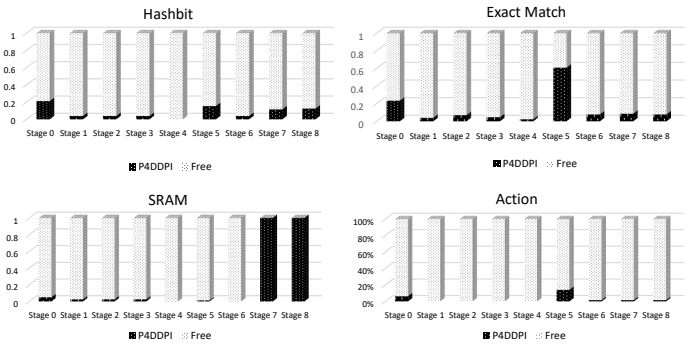


Fig. 5: Resources occupied by P4DDPI.

they are infrequent and can be resolved by sendin the control plane.

Since P4DDPI uses recirculation to parse all in a domain, the processing time of a packet in t is proportional to the size of the domain name. ... ..., the maximum length of a domain name is 253, however, in practice domain names are much shorter. For instance, the evaluated Shalla blacklist dataset [21] shows that the maximum number of recirculations for a domain is 3 (labels' length is less than 19). This attests to the low overhead created by P4DDPI in recirculating DNS packets.

DNS inspection is one security functionality network administrators might need to implement alongside several others. Fig. 5 shows the resources occupied by P4DDPI, in particular, hashbits, exact match, Static RAM (SRAM), and actions. In the majority of the stages, P4DDPI occupies less than 50% of the displayed resources. This allows for other functionalities to be implemented alongside DNS inspection.

VI. Limitations and Lessons Learned

P4DDPI promotes offloading security functionalities into the data plane, especially those that can impact the performance when implemented on a general-purpose CPU. The advantage of P4DDPI is that it can act as the first line of defense to filter domains listed by the network administrator. The effect of P4DDPI can be mostly realized in high-speed networks where security and performance are both required. Despite the advantages of programmable switches, the authors stress that they are not intended to fully replace firewalls.

When implementing P4DDPI, the hardware resource did not permit parsing the whole length of the label in a domain that can go up to 63 characters [1]. Additionally, the number of domains that can be stored and matched at line-rate is limited to the memory in the switch. In particular, P4DDPI is able to fit around 200,000 entries of domain names in the match-action tables. For a larger number of blocked domains, external memory access is needed. For instance, the authors in [29] showed that implementing Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) protocol in a switch using a P4 program, while allocating 10 GB of buffer size, will achieve a rate of around 20 million packets per second without any packet loss.

If P4DDPI is implemented on a network switch that does *bump-in-the-wire* processing, it will process DNS response packets but cannot filter them out. Instead, P4DDPI can filter the subsequent packets having a malicious destination IP address (observed from a previous DNS query response). The reason for this limitation is that P4DDPI removes the parsed DNS labels after every recirculation (in order to parse more labels). Thus, the DNS response packet that arrives at the switch must be forwarded normally for the DNS protocol to work properly in the network. A workaround for that is to use external memory to buffer DNS packets [14, 30] and forward/drop them after the recirculation ends.

Encrypted DNS, such as DNS over TLS (DoT) and DNS over HTTPS (DoH), has been prevailing to preserve privacy, bypass censorship, prevent spoofing DNS responses, etc. P4DDPI is ineffective against parsing encrypted DNS queries, however, this is not a limitation restricted to programmable switches as it applies to general-purpose DPI techniques as well [31].

VII. Conclusion and Future Work

This paper presents P4DDPI, a novel approach that performs DPI on DNS to enforce security policies and block malicious domains in the data plane. The scheme leverages packet recirculation to parse any number of labels in a domain; thus, offloading heavy DNS inspection done by the firewall. Evaluations show that P4DDPI does not have any performance penalty, whereas DNS inspection on a general-purpose server running pfsense can exponentially overload the CPU usage and degrade the throughput. Future work includes exploiting the DPI to mitigate DNS attacks, such as DNS water torture that can overwhelm a specific domain [11]. Additionally, DPI and packet metadata can be utilized to infer malicious traffic exploiting encrypted DNS traffic (e.g., DoT and DoH). Optimization techniques of the P4 program can be further explored to parse more characters in a label, store more domain names in the switch, etc.

Acknowledgement

This work was supported by the U.S. National Science Foundation, award 2118311.

REFERENCES

[1] P. Mockapetris *et al.*, "Domain names-concepts and facilities," 1987.

[2] S. Torabi, A. Boukhtouta, C. Assi, and M. Debbabi, "Detecting internet abuse by analyzing passive dns traffic: A survey of implemented systems," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3389–3415, 2018.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110, 2017.

[4] P. Jeitner and H. Shulman, "Injection attacks reloaded: Tunnelling malicious payloads over dns," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 3165–3182, 2021.

[5] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer security: principles and practice*. Pearson Education Upper Saddle River, NJ, USA, 2012.

[6] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, "Sane: A protection architecture for enterprise networks.," in *USENIX Security Symposium*, vol. 49, p. 50, 2006.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[8] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[9] A. AlSabeh, J. Khoury, E. Kfoury, J. Crichigno, and E. Bou-Harb, "A survey on security applications of p4 programmable switches and a stride-based vulnerability assessment," *Computer Networks*, p. 108800, 2022.

[10] J. Kim, H. Kim, and J. Rexford, "Analyzing traffic by domain name in the data plane," 2021.

[11] A. Kaplan and S. L. Feibish, "Dns water torture detection in the data plane," in *Proceedings of the SIGCOMM'21 Poster and Demo Sessions*, pp. 24–26, 2021.

[12] J. Woodruff, M. Ramanujam, and N. Zilberman, "P4dns: in-network dns," in *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pp. 1–6, IEEE, 2019.

[13] 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, IEEE, 2020.

[14] T. Jepsen, D. Alvarez, N. Foster, C. Kim, J. Lee, M. Moshref, and R. Soulé, "Fast string searching on pisa," in *Proceedings of the 2019 ACM Symposium on SDN Research*, pp. 21–28, 2019.

[15] J. Hypolite, J. Sonchack, S. Hershkop, N. Dautenhahn, A. DeHon, and J. M. Smith, "Deepmatch: practical deep packet inspection in the data plane using network processors," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, pp. 336–350, 2020.

[16] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through dns data analysis," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[17] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, 2021.

[18] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, "Poseidon: Mitigating volumetric ddos attacks with programmable switches," in *the 27th Network and Distributed System Security Symposium (NDSS 2020)*, 2020.

[19] E. F. Kfoury, J. Crichigno, E. Bou-Harb, D. Khoury, and G. Srivastava, "Enabling tcp pacing using programmable data plane switches," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 273–277, IEEE, 2019.

[20] Raymond, "What is the real maximum length of a DNS name?." [Online]. Available: https://tinyurl.com/y8evymtx.

[21] Shalla Secure Services, "Shalla's Blacklists." [Online]. Available: https://tinyurl.com/ymf87tdj.

[22] "pfsense." [Online]. Available: https://www.pfsense.org/.

[23] E. Networks, "Wedge 100BF-65X - 100 GbE Data Center Switch Bare-Metal Hardware." [Online]. Available: https://tinyurl.com/4z8wz53r.

[24] A. Tirumala, "Iperf: The tcp/udp bandwidth measurement tool," *http://dast. nlanr. net/Projects/Iperf/*, 1999.

[25] N. Winn, "dns-flood," 2015. [Online]. Available: https://tinyurl.com/mwmp542z.

[26] "Token bucket filter," 2021. [Online]. Available: https://tinyurl.com/5n94hcef.

[27] "Google Public DNS for ISPs." [Online]. Available: https://tinyurl.com/y85kksay.

[28] R. Kiaei, "Securing Network Infrastructure for DNS Servers." [Online]. Available: https://tinyurl.com/ywzmnmvv.

[29] R. Beltman, S. Knossen, J. Hill, and P. Grosso, "Using p4 and rdma to collect telemetry data," in *2020 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*, pp. 1–9, IEEE, 2020.

[30] D. Kim, Y. Zhu, C. Kim, J. Lee, and S. Seshan, "Generic external memory for switch data planes," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pp. 1–7, 2018.

[31] L. Jin, S. Hao, H. Wang, and C. Cotton, "Understanding the impact of encrypted dns on internet censorship," in *Proceedings of the Web Conference 2021*, pp. 484–495, 2021.