Enabling Line-Rate TLS SNI Inspection in P4 Programmable Data Planes

Ali AlSabeh*, Ali Mazloum†, Elie Kfoury †, Jorge Crichigno†, Hala Strohmier Berry*
*College of Sciences and Engineering, University of South Carolina Aiken (USCA)
†College of Engineering and Computing, University of South Carolina (USC)
Email: ali.alsabeh@usca.edu, {amazloum, ekfoury}@email.sc.edu, jcrichigno@cec.sc.edu, hala.strohmier@usca.edu

***Abstract*—With the increasing adoption of the HyperText Transfer Protocol Secure (HTTPS), organizations face new challenges in monitoring traffic to defend against attacks and enforce security policies, such as filtering malicious websites. One widely used technique to monitor HTTPS is by scrutinizing the hostname in the Server Name Identification (SNI) extension during the Transport Layer Security (TLS) handshake. Parsing the SNI typically involves Deep Packet Inspection (DPI), often performed on general-purpose processors, which can create bottlenecks and significantly impact network throughput. In response, this paper introduces a novel framework for parsing and identifying SNI hostnames in the data plane at line-rate using P4. Evaluation results on recent publicly available datasets from various regions and platforms demonstrate that our framework can successfully parse 85%-99% of hostnames in P4. Furthermore, performance analysis reveals that the proposed data plane solution can inspect the hostname in approximately 1 microsecond (μs), representing orders of magnitude improvement over solutions running on Central Processing Units (CPUs).**

***Index Terms*—TLS, SNI, HTTPS, P4 programmable switches, high-speed network monitoring, DPI**

I. Introduction

The Google Transparency Report reveals a significant surge in the adoption of the HyperText Transfer Protocol Secure (HTTPS) in the Chrome browser within the United States, soaring from 44% in March 2015 to an impressive 98% by March 2024. Furthermore, data indicates that over 94% of requests to Google from the top ten countries globally are now encrypted [1]. This dramatic increase can be attributed to various factors, including heightened awareness of user privacy, widespread adoption of HTTPS by major services such as Gmail, Meta, and YouTube, and advancements in computational power enabling encryption and decryption [2].

HTTPS, which runs on top of the Transport Layer Security (TLS) protocol [3], provides significant benefits in terms of confidentiality and authentication. HTTPS monitoring is desirable for a multitude of reasons related to network security, such as attack detection, security policy violations, and malicious activities [4]. This is partly due to the limitations of traditional monitoring techniques, which struggle to keep pace with modern network architectures and protocols. For instance, monitoring solely based on low-level identifiers such as IP addresses and port numbers proves ineffective when multiple services share the same IP address (resulting in multiple domain names associated with a single IP address) or when dynamic port allocation mechanisms such as Network Address Translation (NAT) are implemented [5].

Higher-level network monitoring, extending beyond network layers 3 and 4, is extensively discussed in the literature and implemented in practice. However, such mechanisms often rely on Deep Packet Inspection (DPI), typically performed by security middleboxes with general-purpose CPUs and GPUs, can cause bottlenecks under high traffic [6] and raise privacy concerns [7].

The introduction of P4 Programmable Data Plane (PDP) switches offers a novel approach for advanced network security monitoring, including ransomware attacks, which have wreaked havoc across the Internet [8], and botnets [9, 10]. Additionally, the Domain Name System (DNS), which maps domain names to IP addresses, is being implemented in P4 PDP switches for domain name extraction and monitoring [7, 11, 12].

Although raw DNS traffic can be used for monitoring and security services, it becomes ineffective in modern browsers that encrypt DNS traffic using the DNS over HTTPS (DoH) protocol. As a proof of concept, we attempted to monitor and block domain names in the Palo Alto virtual firewall using the DNS sinkhole feature, but access to these domains remained possible in Firefox and Chrome, where DoH is the default setting [13, 14]. To overcome encrypted DNS hurdles, Palo Alto virtual firewall offers other features to monitor traffic, such as inspecting the hostname in the TLS Server Name Identification (SNI) extension (namely, the URL filtering capability). This extension contains the DNS hostname the client needs to contact.

Monitoring the TLS protocol in the P4 PDP switches offers several advantages to the security and operation of the network, however, it has not been addressed in the literature due to the complexity of the TLS protocol headers and the software and hardware limitations of P4 PDP switches [7]. To this end, this paper proposes an efficient P4-based implementation of the TLS header to extract the hostname in the SNI extension, enabling both fine-grained and coarse monitoring applications. The contributions of this paper are summarized as follows:

- Implement a line-rate P4 program that extracts the hostname from the TLS SNI extension in ≈ 1 microsecond (μs). The implementation is publicly available online for deployment and research endeavors [15].

- Design a framework that can perform fine-grained monitoring (exact match) on 500,000 hostnames, and coarse-grained monitoring (ternary match) on 15,000 hostnames. The framework is deployed entirely in the data plane, ensuring privacy-preserving monitoring.
- Evaluate the proposed framework on recent TLS datasets collected from various sources and platforms containing normal and malicious hostnames. Our results show that the proposed approach is highly effective on current TLS traffic usage on the Internet.
- Showcase the performance gains of the proposed system against Suricata Intrusion Detection/Prevention System (IDS/IPS) [16]. The system can be deployed as an active middlebox without affecting normal traffic.

The remainder of the paper is organized as follows. Section II reviews related works and compares them to our approach. Section III provides background information on the TLS protocol and P4 PDP switches. In Section IV, we describe the proposed approach and detail the P4 implementation. Section V presents experimental results. Section VI discusses the limitations of the proposed approach and outlines future research directions. Finally, Section VII concludes the paper.

II. Related Work

Previous works on TLS inspection have been applied to various network security applications [17]. However, these approaches rely on general-purpose programming languages, which simplify DPI and TLS packet parsing.

Interest in using P4 for DPI is growing, with early efforts focusing on the DNS protocol. Meta4 [7] is a network monitoring framework that parses up to four DNS subdomains, with a limit of 15 characters per subdomain. Kaplan et al. [11] expand on this by parsing six subdomains, with a total of 60 characters. In our previous work [6, 12], we addressed DNS parsing challenges using packet recirculation and truncation. This technique processes the packet repeatedly, removing parts in each iteration until the full domain name is parsed. Thus, allowing more domains to be parsed at the expense of keeping the packet for a longer period in the data plane.

DNS parsing poses challenges in P4 due to the variable lengths of subdomains. However, DNS operates at layer 7 and is encapsulated within a UDP packet with a fixed, short header length, facilitating access to the DNS layer. In contrast, parsing the TLS protocol is more complex due to the variable lengths of header fields and the random order of TLS extensions.

Other P4-based DPI efforts focus on string matching and domain filtering techniques. DeepMatch [18] offers a line-rate DPI primitive in the data plane with smart NICs [19], enabling new applications like Quality of Service (QoS), network monitoring, and IDS for advanced security policies. Jepsen et al. [20] developed a P4-based system to locate string keywords in packet payloads using parallel search with partitioned Deterministic Finite Automata (DFA). They use packet recirculation and truncation to search the entire packet payload. DeeP4R [21] is similar but keeps the original packet intact during recirculation, taking ≈ 1 millisecond (ms) to search domain names. This is primarily due to the approach that searches byte-by-byte for any string occurring in the packet. In contrast, our system runs at line-rate and is much more efficient, taking ≈ 1 μs.

III. Background

A. TLS

HTTPS is the secure version of HTTP, the primary protocol used to transfer data between a web browser and a website. HTTPS uses the TLS protocol, formerly known as the Secure Socket Layer (SSL) protocol, for encryption and authentication. To establish a TLS connection, a handshake process must occur between the client and the server. The handshake process starts with a Client Hello message, where the client asks for the web server's TLS certificate. Following this, a series of messages is exchanged between the client and the server before the session is established.

The exhaustion of IPv4 addresses has led to shared web hosting, where multiple domains share a single IP address. For instance, the domains `example.net` and `example.com` share the same IP address. As a result, the client has to indicate the domain it is connecting to. SNI is an extension for the TLS protocol that ensures that the clients can specify the domain name of the server during the TLS handshake during the Client Hello message [22]. Almost all browsers, web servers, and OSes support the SNI extension, and it has been utilized for several applications, such as service inspection, URL filtering, etc. [5, 23, 24]. In the context of TLS SNI, the terms hostname and domain name are used interchangeably.

B. Programmable Switch Primer

The Protocol Independent Switch Architecture (PISA) is a data plane programming model that includes the following elements: programmable parser, programmable match-action pipeline consisting of multiple stages, and programmable deparser. The programmable parser is represented as a state machine that can define the headers that need to be parsed (e.g., Ethernet, IP, DNS, or even custom headers). The programmable match-action pipeline consists of multiple match-action units to match against packet header fields and apply actions with supplied action data. Each unit can include one or more Match-Action Tables (MATs) that are coupled with Static Random Access Memory (SRAM) or Ternary Content Addressable Memory (TCAM) for storing lookup keys and action data. Additional action logic can be implemented using stateful objects, such as registers that are stored in SRAM. Lastly, the programmable deparser defines how packet headers are reassembled when they exit the switch [25]. The high-level language used for programming PISA is P4. Unlike general-purpose programming languages, P4 is domain-specific and optimized to handle Tbps of network traffic. Despite the increasing number of emerging applications in P4 (e.g., network telemetry, security, etc. [26, 27]), preserving Tbps throughput requires limiting the complexity of the pipeline stages and permitting only elementary actions (e.g., simple arithmetic). Such limitations require expertise and knowledge of the language

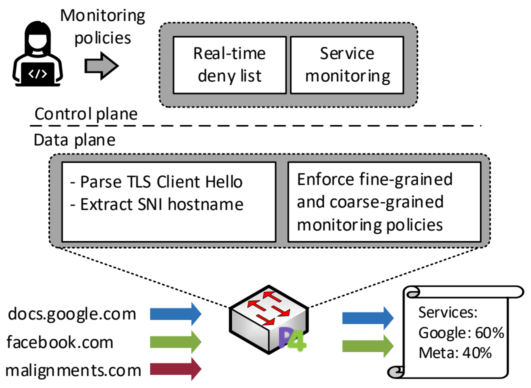


Fig. 1: Overall architecture of the proposed system.

and architecture to devise workarounds when implementing complex functionalities in the switch.

IV. TLS SNI Packet Processing in P4

A. System overview

Fig. 1 presents a high-level overview of the proposed system. The P4 PDP switch serves as the core component, responsible for processing TLS Client Hello packets and extracting hostnames from the TLS SNI extension. Once the hostname is extracted, the P4 PDP switch can enforce fine-grained and coarse-grained monitoring on it. Fine-grained analysis enables exact matching of hostnames, ensuring precise identification. Coarse-grained monitoring offers a more flexible approach by allowing ternary matching (e.g., any hostname that ends with `google.com` can be monitored in a rule matching `*.google.com`); however, it is more limited in the data plane. Fine-grained monitoring can be used for real-time deny lists sourced from various intelligence sources. Coarse-grained monitoring, on the other hand, can be used for service monitoring and fingerprinting [23]. The P4 program operates at line-rate with basic packet routing and forwarding functionalities, thus enabling the P4 PDP switch to be deployed as an active privacy-preserving monitoring device.

B. P4 implementation

Fig. 2 shows the P4-based ingress parser and match-action pipeline implementation of the proposed system, where most of the implementation resides. The P4 PDP switch intercepts the TLS Client Hello packet and starts parsing the fixed header fields. The TLS header has multiple variable-length fields that need to be parsed before reaching the hostname in the SNI extension, namely the session ID, ciphers, and compression methods supported by the client. Each of these header fields is preceded by its length. The parser reads this length and then skips the entire header using the Tofino parser "advance" capability. To skip header fields with variable lengths, a state is created for each possible length. For instance, in the session ID states, the parser can parse any session ID of any length between 1 and 32 bytes. The choice to efficiently skip these variable-length headers, rather than extracting them into the PHV header, is key to parsing the hostname at line-rate in the presence of a complex header. Once the parser skips the session ID, ciphers, and compression methods, multiple TLS extensions (each with variable length) can be encountered before reaching the SNI extension. Consequently, we form a loop in the parser that keeps skipping TLS extensions until the SNI extension is reached (SNI extension is type 0). To comply with the parser's limitations, each encountered TLS extension can be parsed as long as its length is less than 30 bytes. Randomizing the order of the TLS extensions has been specified in TLS version 1.3 [28] to reduce the risk of ossification by external implementers that make it difficult to deploy future modifications to TLS [29].

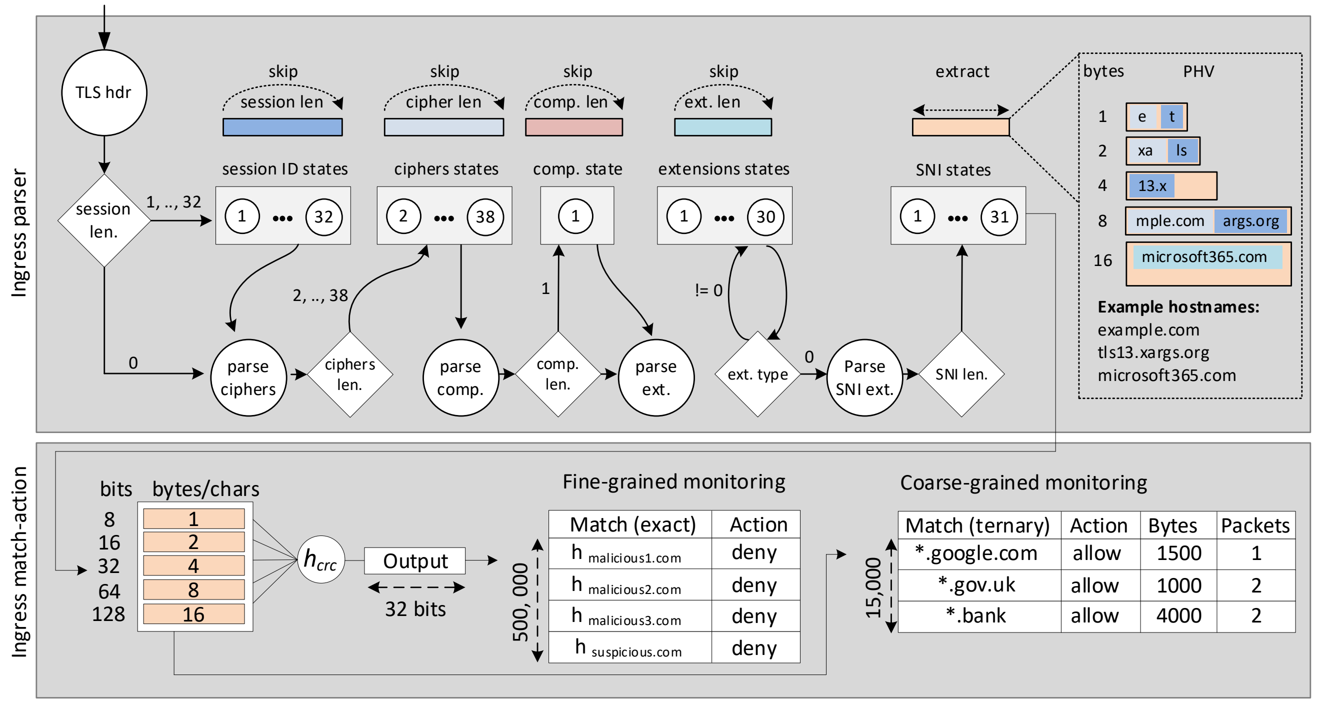
To parse the variable-length hostname, its length is first read, and then the parser starts extracting it if it is less than 31 characters to avoid partial extraction. A straightforward, but impractical, approach to extracting the hostname is to define a header for each possible length and extract the entirety of the hostname at once. This would require N headers, where N is the maximum number of characters that can be parsed. However, this approach is not scalable and would waste scarce resources. Alternatively, the proposed system follows a more efficient approach by extracting a predefined number of bytes that is an exact power of two. The proposed system defines headers with lengths of $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, and $2^4 = 16$, thus, accommodating a total length of 31 bytes/characters. For instance, the hostname `example.com`, with a length of 11, will be extracted into the headers 1, 2, and 8.

After parsing the hostname, the ingress match-action pipeline implements a two-fold monitoring technique: fine-grained and coarse-grained. In fine-grained monitoring, the extracted hostname is hashed using CRC_{32}, and the output is matched to a table that performs exact matching. Thus, allowing the inspection of $500,000$ hostnames. In coarse-grained monitoring, the entirety of the extracted hostname is matched to a table that performs ternary matching. Since the hostname is much longer than the hash and the ternary match occupies more resources, the table can inspect $15,000$ entries. The coarse-grained monitoring table is associated with counters to collect packet and byte counts for each entry for service monitoring use cases.

V. Evaluation

A. Dataset

Virus Total malware traffic. Using their academic services, we crawled and retrieved hundreds of gigabytes of malware samples between 2017 and 2021 from Virus Total [30]. Samples that do not initiate TLS connections are removed using the metadata file associated with each sample. Subsequently, each sample was instrumented in an isolated environment to capture its network traffic behavior, i.e., the Packet Capture (PCAP), file for 5 minutes. The resulting dataset contains 276 malware samples. We make this dataset, along with the others, available online for analysis of the network behavior in contemporary malware samples [15].



Match (exact)	Action
h $_{malicious1.com}$	deny
h $_{malicious2.com}$	deny
h $_{malicious3.com}$	deny
h $_{suspicious.com}$	deny

Match (ternary)	Action	Bytes	Packets
*.google.com	allow	1500	1
*.gov.uk	allow	1000	2
*.bank	allow	4000	2

Fig. 2: Parsing TLS SNI and applying it for fine-grained and coarse-grained analysis using P4.

CTU normal and botnet traffic. This dataset contains Zeek [31] log files captured at CTU University [32], which include a large amount of real botnet traffic mixed with normal and background traffic. The Zeek log files contain information on TLS connections, including the hostname.

Cross platform and browser traffic. This dataset is collected by the authors of Flowprint [33] and is publicly available. The cross-platform dataset consists of user-generated data from 215 Android and 196 iOS applications in the US, India, and China. The browser traffic contains traces from scraping the top $1,000$ Alexa websites with Chrome, Firefox, Samsung Internet, and UC browsers.

B. P4 TLS Parsing Capabilities

The complexity of the TLS header, combined with the limited hardware resources of P4 PDP switches, results in two main cases where TLS Client Hello packets cannot be parsed up to the hostname. The first case occurs when the hostname length is greater than 31. The second case occurs if any of the TLS headers preceding the SNI extension (e.g., the session ID, cipher, compressions, or other prior extensions) have a length greater than the specified threshold in P4.

To study the effectiveness of the implemented system, the length of hostnames from various normal and malicious dataset sources was analyzed in Fig. 3. In the top 1 million hostnames from Cloudflare [34] and normal CTU traffic, 0.31% and 5.9% of hostnames, respectively, cannot be parsed in P4 since they have a length greater than 31 characters. As for malicious hostnames from CTU botnet and Virus Total, 15.9% and 4.8% of hostnames are missed, respectively. In this experiment, only the hostname length is analyzed since packet capture files were not available or were truncated due to privacy reasons (except the Virus Total dataset). Nonetheless, this shows us that the chosen threshold for the hostname length (31 characters) is effective enough to cover the vast majority of hostnames.

Moreover, we study the effectiveness of the proposed approach on the entirety of TLS headers from traces in the cross platform and browser datasets. Fig. 4 shows the ratio of traffic missed due to headers that could not be parsed in P4. Noticeably, TLS traffic in iOS applications has the highest ratio of missed traffic (maximum of 12%) due to the large

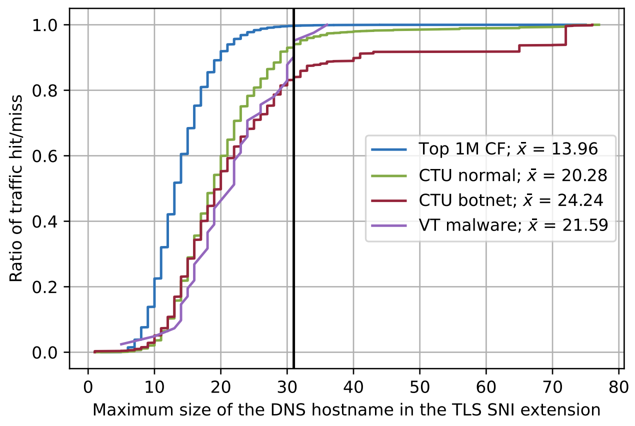


Fig. 3: Ratio of hit/missed DNS hostnames in multiple benign and malicious datasets. The vertical black line at $x = 31$ indicates the maximum number of characters in a hostname that the proposed system can parse.

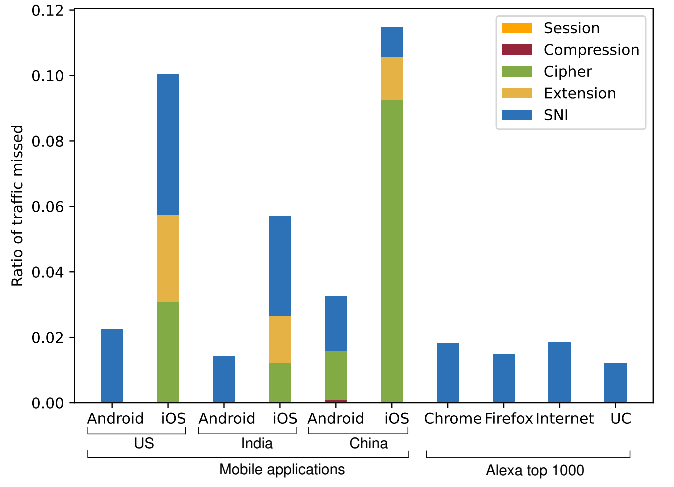


Fig. 4: Ratio of traffic missed in TLS data from mobile applications and top 1000 Alexa websites.

number of ciphers supported.

C. Performance

To showcase the performance gains of deploying TLS inspection in the data plane, the proposed approach is implemented on the Edgecore Wedge 100BF-32X, which is designed with programmable Tofino switch silicon from Intel Networks [35] and allows programming using the P4 language. These obtained results from the Tofino switch are compared with Suricata IDS/IPS version 6.0.4 running on a virtual machine with 32 GB RAM and 8 cores allocated. For fairness, only Suricata rules that inspect the hostnames were installed and the delay incurred from inspecting TLS traffic is measured. Similarly, the P4 code is modified to include built-in timestamps and measure the delay in the switch. Fig. 5 shows that the P4 switch takes around 700 nanoseconds (ns) to 900 ns when the TLS Client Hello packet does not have extensions in random order preceding the SNI extension (seen in TLS 1.2). This delay can increase to $\approx$ 1.5 μs, when the P4 parser uses loops to bypass extensions in random order (seen in TLS 1.3). The obtained data plane speeds are orders of magnitude faster than those obtained in CPU-based Suricata, which can take tens to hundreds of microseconds.

VI. Limitations and Future Work

The proposed framework heavily relies on the ingress parser and match-action pipeline for its implementation. It can parse a vast majority of hostnames in TLS traffic; however, some use cases cannot be accommodated within the current program. This limitation primarily stems from the parser being fully utilized as depicted in Fig. 6. Nonetheless, our approach can forward the small percentage of unparsed packets to the control plane for further inspection. Future work involves leveraging both the ingress and egress to further maximize the percentage of hostnames that can be parsed in the data plane. Alternatively, packet recirculation can be utilized with some sacrifice in delay (each recirculation adds $\approx$ 1 μs delay).

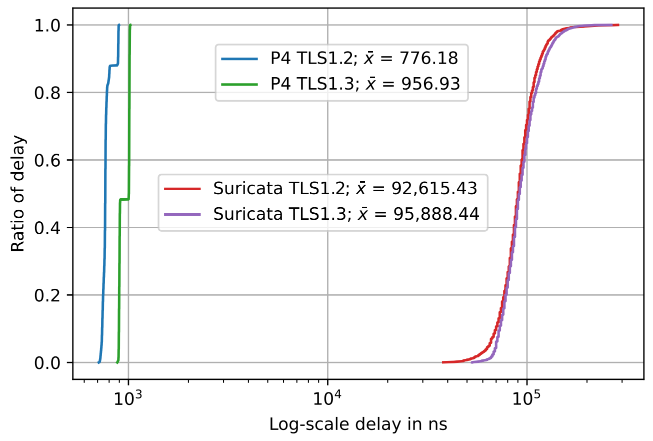


Fig. 5: Delay incurred in P4 to parse the hostname in TLS extensions compared to that in Suricata IPS.

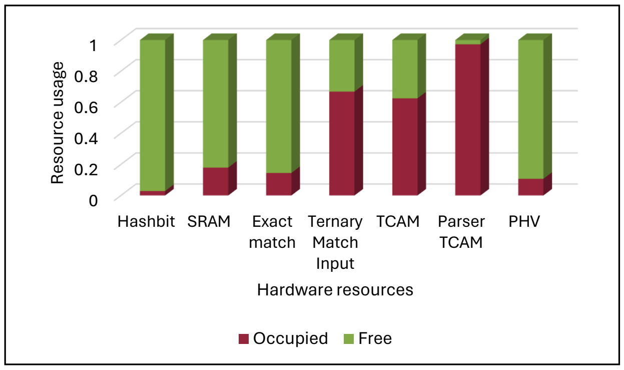


Fig. 6: Resource usage in the P4 Tofino ASIC across all pipeline stages.

It is also worth mentioning two exploits where SNI inspection becomes more challenging. The first occurs when some servers continue the handshake process even if they do not recognize the SNI (backward compatibility exploit [36]). In such cases, attackers can inject fake hostnames that do not match firewall rules. The second exploit arises when servers have alternative names that can appear in their TLS certificate. For instance, alternative names for YouTube's TLS certificate may include `*.google.com`, `*.googlevideo.com`, `*.google.fr`. This can affect the accuracy of monitoring and filtering services. Future work aimed at enhancing P4 in such exploits might involve shifting focus to more DPI and analyzing additional parameters in the TLS session handshake [4].

Finally, the new TLS Encrypted Client Hello (ECH) extension, which encrypts the hostname, renders the proposed approach, along with many monitoring services, impractical. However, this extension faces several challenges and is not widely adopted, primarily because many services use TLS 1.2, which does not support ECH [37]. Future efforts to empower P4 in encrypted hostname traffic might involve focusing on a set of unencrypted metadata exchanged between the client and the server.

VII. Conclusion

This paper presents a novel line-rate approach that performs DPI on TLS Client Hello packets to extract the DNS hostname in the SNI extension in the data plane. The scheme implements an efficient P4 program that overcomes the complexity of TLS headers and the hardware and software limitations of P4 PDP switches. Running at line-rate, the program allows the PDP switch to serve as an active device for both fine-grained and coarse-grained monitoring. The former can track $500,000$ exact hostnames, making it effective for domain name filtering, while the latter can match $15,000$ hostnames using regular expressions for service monitoring. Evaluations show the proposed framework can parse 85%-99% of hostnames across various datasets, platforms, and regions (e.g., Android, iOS, different browsers, normal and malware traffic). Compared to Suricata IDS/IPS, the framework processes data in $\approx 1\ \mu s$, orders of magnitude faster than Suricata on a general-purpose CPU. Resource usage indicates that the scheme occupies minimal space in the match-action pipeline, leaving room for further innovation in the data plane. Optimization techniques could explore utilizing both the ingress and egress pipelines to cover additional use cases.

Acknowledgement

This work was supported by the U.S. National Science Foundation, OAC-Core award number 2403360 and OAC-CC* award number 2346726.

References

[1] Google Transparency Report, "HTTPS Encryption on the Web." [Online]. Available: https://tinyurl.com/2r6dre39.

[2] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS," in *ACM International on Conference on Emerging Networking Experiments and Technologies*, 2014.

[3] K. Moriarty and S. Farrell, "RFC 8996 Deprecating TLS 1.0 and TLS 1.1," *Internet Eng. Task Force,(IETF)*, 2021.

[4] W. M. Shbair, T. Cholez, J. François, and I. Chrisment, "Improving SNI-Based HTTPS Security Monitoring," in *IEEE International Conference on Distributed Computing Systems Workshops*, 2016.

[5] R. Asaoka, A. Nakao, M. Oguchi, and S. Yamaguchi, "Accuracy Improvement by Occurrence Probability of Service Identification based on SNI," in *2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW)*, 2022.

[6] A. AlSabeh, K. Friday, J. Crichigno, and E. Bou-Harb, "Effective DGA Family Classification using a Hybrid Shallow and Deep Packet Inspection Technique on P4 Programmable Switches," in *IEEE International Conference on Communications*, 2023.

[7] J. Kim, H. Kim, and J. Rexford, "Analyzing traffic by domain name in the data plane," in *ACM SIGCOMM Symposium on SDN Research (SOSR)*, 2021.

[8] A. AlSabeh, H. Safa, E. Bou-Harb, and J. Crichigno, "Exploiting Ransomware Paranoia For Execution Prevention," in *IEEE International Conference on Communications (ICC)*, 2020.

[9] K. Friday, E. Kfoury, E. Bou-Harb, and J. Crichigno, "INC: In-Network Classification of Botnet Propagation at Line Rate," in *European Symposium on Research in Computer Security*, 2022.

[10] K. Friday, E. Bou-Harb, and J. Crichigno, "A learning methodology for line-rate ransomware mitigation with p4 switches," in *International Conference on Network and System Security*, 2022.

[11] A. Kaplan and S. L. Feibish, "Practical Handling of DNS in the Data Plane," in *Proceedings of the Symposium on SDN Research*, pp. 59–66, 2022.

[12] A. AlSabeh, E. Kfoury, J. Crichigno, and E. Bou-Harb, "P4DDPI: Securing P4-Programmable Data Plane Networks via DNS Deep Packet Inspection," in *Proceedings of the 2022 Network and Distributed System Security (NDSS) Symposium*, pp. 1–7, 2022.

[13] C. B. Enterprise, "DNS-over-HTTPS Setting." [Online]. Available: https://tinyurl.com/3czxd2c4.

[14] M. Support, "Configure DNS over HTTPS Protection Levels in Firefox." [Online]. Available: https://tinyurl.com/mr3dyjjs.

[15] "P4-TLS." [Online]. Available: https://tinyurl.com/56rn6sny.

[16] "Suricata." [Online]. Available: https://suricata.io/.

[17] M. Laštovička, S. Špaček, P. Velan, and P. Čeleda, "Using TLS Fingerprints for OS Identification in Encrypted Traffic," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–6, IEEE, 2020.

[18] 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.

[19] E. F. Kfoury, S. Choueiri, A. Mazloum, A. AlSabeh, J. Gomez, and J. Crichigno, "A comprehensive survey on smartnics: Architectures, development models, applications, and research directions," *IEEE Access*, vol. 12, 2024.

[20] 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.

[21] S. Gupta, D. Gosain, M. Kwon, and H. B. Acharya, "DeeP4R: Deep Packet Inspection in P4 using Packet Recirculation," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pp. 1–10, IEEE, 2023.

[22] Cloudflare, "What is SNI? How TLS server name indication works." [Online]. Available: https://tinyurl.com/dp5nvpsh.

[23] R. Asaoka, Y. Soma, H. Yamauchi, A. Nakao, M. Oguchi, S. Yamaguchi, and A. Kobayashi, "Service Identification of TLS Flows Based on Handshake Analysis," *Journal of Information Processing*, vol. 31, pp. 131–142, 2023.

[24] PaloAlto, "URL Filtering Lookup and Enforcement Differences between Clear Text HTTP and Encrypted HTTPS Traffic." [Online]. Available: https://tinyurl.com/42jnxemm.

[25] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, "A survey on data plane programming with p4: Fundamentals, advances, and applied research," *Journal of Network and Computer Applications*, vol. 212, p. 103561, 2023.

[26] 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*, vol. 207, p. 108800, 2022.

[27] 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*, vol. 9, 2021.

[28] RFC8446, Internet Engineering Task Force (IETF), " The Transport Layer Security (TLS) Protocol Version 1.3." [Online]. Available: https://tinyurl.com/4w3dhk9a.

[29] Chrome Platform Status, "Feature: TLS ClientHello extension permutation." [Online]. Available: https://tinyurl.com/5c6zf47p.

[30] "VirusTotal." [Online]. Available: https://tinyurl.com/2p9cscna.

[31] "Zeek. An Open Source Network Security Monitoring Tool." [Online]. Available: https://zeek.org/.

[32] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[33] T. Van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. Van Steen, and A. Peter, "FlowPrint: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic," in *Network and distributed system security symposium (NDSS)*, vol. 27, 2020.

[34] "Cloudflare Radar - Domain Rankings." [Online]. Available: https://tinyurl.com/y9x64v5e.

[35] Edgecore Networks, "Data Center Switch: Wedge100BF-32X." [Online]. Available: https://tinyurl.com/mrhf83st.

[36] D. Eastlake 3rd, "Transport Layer Security (TLS) Extensions: Extension Definitions," tech. rep., 2011.

[37] M. Trevisan, F. Soro, M. Mellia, I. Drago, and R. Morla, "Attacking DoH and ECH: Does Server Name Encryption Protect Users' Privacy?," *ACM Transactions on Internet Technology*, vol. 23, no. 1, pp. 1–22, 2023.