NÖHÜ Müh. Bilim. Derg. / NOHU J. Eng. Sci., 2023; 12(1), 001-029

Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences

Derleme makalesi / Review article

www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

A comprehensive survey on deep packet inspection for advanced network traffic analysis: Issues and challenges

Modern ağ trafiği analizi için derin paket incelemesi hakkında kapsamlı bir çalışma: Sorunlar ve zorluklar

Merve Çelebi[1,*], Alper Özbilen[2], Uraz Yavanoğlu[3]

[1] *Gazi University, Department of Computer Forensics, Informatics Institute, 06500, Ankara, Türkiye*
[2] *Interprobe, 06800, Ankara, Türkiye*
[3] *Gazi University, Department of Computer Engineering, 06570, Ankara, Türkiye*

Abstract

Deep Packet Inspection (DPI) provides full visibility into network traffic by performing detailed analysis on both packet header and packet payload. Accordingly, DPI has critical importance as it can be used in applications i.e network security or government surveillance. In this paper, we provide an extensive survey on DPI. Different from the previous studies, we try to efficiently integrate DPI techniques into network analysis mechanisms by identifying performance-limiting parameters in the analysis of modern network traffic. Analysis of the network traffic model with complex behaviors is carried out with powerful hybrid systems by combining more than one technique. Therefore, DPI methods are studied together with other techniques used in the analysis of network traffic. Security applications of DPI on Internet of Things (IoT) and Software-Defined Networking (SDN) architectures are discussed and Intrusion Detection Systems (IDS) mechanisms, in which the DPI is applied as a component of the hybrid system, are examined. In addition, methods that perform inspection of encrypted network traffic are emphasized and these methods are evaluated from the point of security, performance and functionality. Future research issues are also discussed taking into account the implementation challenges for all DPI processes.

Keywords: Deep packet inspection, Network traffic analysis, Security, Survey

Öz

Derin Paket İnceleme (Deep Packet Inspection-DPI), hem paket başlığı hem de paket yükü üzerinde ayrıntılı analizler gerçekleştirerek ağ trafiğinin tam görünürlüğünü sağlar. Ağ güvenliği veya devlet gözetimi gibi uygulamalarda kullanılabilmesi yönüyle DPI, kritik bir öneme sahiptir. Bu çalışmada, DPI hakkında kapsamlı bir araştırma sunulmuştur. Diğer inceleme çalışmalarından farklı olarak bu çalışmanın amacı, modern ağ trafiğinin analiz edilmesi sürecinde performansı sınırlandıran parametreleri belirleyerek DPI tekniğinin ağ analizi mekanizmalarına verimli ve etkili bir şekilde entegrasyonunu sağlamaktır. Karmaşık davranışlar gösteren ağ trafiği modelinin incelenmesinin birden fazla tekniğin bir araya getirilerek güçlü hibrit sistemlerle gerçekleştirildiği göz önünde bulundurularak, DPI metodu, ağ trafiğinin analizinde kullanılan diğer tekniklerle birlikte incelenmiştir. Ağ güvenliği hususunda kritik öneme sahip DPI metodunun IoT ve SDN mimarileri üzerindeki güvenlik uygulamaları tartışılmış ve DPI'ın IDS'lere hibrit sistemin bir bileşeni olarak uygulandığı mekanizmalar incelenmiştir. Ayrıca, Şifreli ağ trafiğinde inceleme gerçekleştiren yöntemler üzerinde durulmuş ve bu yöntemler güvenlik, performans ve fonksiyonellik açılarından değerlendirilmiştir. Son olarak, tüm DPI süreçleri için uygulama zorlukları ve bu zorluklarla ilişkili gelecek araştırma konuları ele alınmıştır.

Anahtar kelimeler: Derin paket inceleme, Ağ trafiği analizi, Güvenlik, Araştırma

1 Introduction

Modern networks with large number of nodes, such as IoT, need to be regularly monitored in order to maintain their performance. Maintaining performance in these networks with different purposes may include prioritizing one or more of the issues such as ensuring quality of service (QoS) requirements, identifying problems that threaten network security or improving resource consumption. These objectives can be achieved through the application of network traffic monitoring and analysis (NTMA) techniques such as network security, network traffic classification, fault management and traffic forecasting. In the process of implementing NTMA techniques, different requirements can be defined for the acquisition of traffic data. In this regard, network packets are generally considered as targets to be examined at the traffic data collection tasks [1]. Packet inspection can be expressed as an ability to inspect network traffic for a specific aim in real-time or offline. Three basic methods, classified according to depth, are used to monitor network traffic and evaluate its performance [2]. In Shallow Packet Inspection (SPI) technique, only header information of each packet is examined, and the payload is not taken into account. This technique focuses on the second and the third layers in the OSI model. With SPI method, IP addresses of the sender and the receiver, the number of packets that a

* Sorumlu yazar / Corresponding author, e-posta / e-mail: merveorakci@gazi.edu.tr (M. Çelebi)
Geliş / Recieved: 04.10.2022 Kabul / Accepted: 14.11.2022 Yayımlanma / Published: 15.01.2023
doi: 10.28948/ngmuh.1184020

message is broken into, the number of hops per packet and the synchronization data that allows for reassembling packets can be examined. Medium Packet Inspection (MPI) technique inspects header and payload of network packets up to the presentation layer. MPI technologies can prioritize some packets by examining application commands in the application layer and file formats in the presentation layer. This examination method is commonly used by application proxies and provides a more comprehensive analysis than SPI technique [3]. Unlike SPI and MPI techniques, DPI performs a detailed inspection covering all headers of the whole layers and the packet payload.

The ability to have information on both packet header and payload in real-time provides control over the communication between two endpoints. Network controllers such as Internet Service Providers (ISP) use network technologies that enable real-time monitoring of network packets. This technology is known as DPI [4]. DPI generally provides an in-depth analysis of packets passing through a certain network point and makes some decisions based on the analysis. This method is called DPI because analysis covers both packet header and packet payload. There are two main processes of DPI; identification and action. Identification is the process of examining packets and discovering their hidden features. After identification, operations such as keeping logs for analysis of the network or dropping from the network for network security can be performed in the action process [5].

DPI technique is used to detect well-known malware signatures. Additionally, the attack pattern which includes the attack order, the path followed by the attacker and the techniques used by the attacker can be detected in connection with the network flow. Besides detecting known attack patterns with high accuracy, new exploitation techniques can also be discovered via DPI. This allows explorers to build new protection mechanisms and signatures [2]. Thus, DPI can be used in network security and government surveillance applications. Moreover, DPI can be used in content filtering to detect and block harmful or illegal content, bandwidth management, copyright management and applications that allow ISPs to inject advertisements into websites according to users' interests [6-8]. Many open-source tools such as the Linux firewall and commercial tools such as Norton Core use the DPI approach in their products for analysis of the network traffic [9]. Additionally, IDS commonly use payload-based classifiers to identify malicious network activity [10].

There are numerous reviews on current applications of DPI technology [5,11-13]. In the next sub-section, these reviews are examined. Also, the contribution of this paper to the literature are discussed. In the second sub-section of the introduction, the organizational structure of this paper is presented.

1.1 Existing surveys on DPI

In the study numbered [11], the literature review was conducted on the techniques required to develop DPI systems. The impact of challenges associated with complex signature sets and hardware or operating system on DPI implementations are discussed. The analysis of open source DPI modules used in traffic classification is presented in the study numbered [12]. This analysis evaluates classification accuracy and computational requirements over a real data set. The study numbered [5] provides a detailed review of the Regular Expression (RE) Matching Technique. In this study, the state explosion problem in the Finite State Machine (FSM) created for *RE Matching* is emphasized. In this direction, suggested methods for avoiding or mitigating state explosion have been examined, and these suggestions are presented for the creation of compact and efficient automata. Parallel platforms such as Graphics Processing Unit (GPU) or Field-Programmable Gate Array (FPGA) that accelerate the pattern matching process are also discussed. In the study numbered [13], the existing literature on infrastructure and communication for the energy sector and smart grids, also a review of DPI techniques and application areas are presented. The study pays attention to the use of the DPI technique as a security tool for smart grids, and proposes a SDN-based security monitoring framework that uses a hybrid model combining DPI and Deep Learning (DL) technique. Also, an additional framework that performs network forensic analysis is proposed to expand the capabilities of this framework.

These studies numbered [5, 11, 12] and [13], which focus on DPI technology, mainly examine DPI and the application areas of these techniques. Only in the study numbered [11], the parameters that limit the performance of DPI systems are partially mentioned. Despite the widespread use of encrypted network traffic, it is an important shortcoming that a detailed analysis of DPI techniques in this field is not existing in the current literature. In addition, there is no study examining the difficulties of implementing DPI techniques in SDN or IoT architectures in today's modern networks. Accordingly, this paper aims to present a roadmap for the application of DPI technology in today's modern networks. Considering that the analysis of the state-of-the-art network traffic model is carried out with powerful hybrid systems by combining more than one technique, the DPI method is examined together with other techniques used in the analysis of network traffic. In order to protect the confidentiality of network traffic, the proposed techniques for analyzing encrypted traffic are discussed with their advantages and disadvantages. In addition, a comprehensive review of DPI implementation challenges for scenarios that generate state-of-the-art network traffic such as IoT or SDN architectures is conducted, and the security applications of DPI technique on these architectures, whose main focus is a security concern, are discussed. The main purpose of this paper is to perform a detailed analysis on detection and improvement of performance-limiting parameters in all processes from collecting state-of-the-art network traffic and analyzing it via DPI, and to evaluate the contribution of DPI technique to the mechanisms created to analyze the network traffic by examining DPI applications in the existing literature. The contributions of this paper can be listed as follows:

- We comprehensively review the challenges of advanced network traffic analysis, performance-

limiting parameters and implementation of difficulties of DPI techniques.

- We identify implementation challenges and open research issues for DPI systems, and provide insight into topics that will shed light on future work.
- We create a roadmap that enables the use of appropriate DPI techniques to address advanced network analysis challenges.
- We point to the other techniques that complement the DPI technique instead of focusing only on this technique in determining the procedures to be applied in future mechanisms.
- We present a classification of the proposed methods to perform an inspection on encrypted traffic, and discuss the advantages and disadvantages of the techniques that directly use or do not use the DPI approach or limit the usability of this approach.
- We evaluate the role of IDS in ensuring the security of IoT and SDN architectures whose main focus is on security concerns, and discuss the implementation challenges of DPI on these architectures.

1.2 Paper organization

The structure of the paper is shown in Figure 1. In this direction, the schedule in this paper is as follows: In Section 2, the problems encountered in the process of receiving and processing network packets from Network İnterface Card (NIC) are determined and the solutions to these problems are presented. In addition, the platforms developing based on these solution methods are examined. In the first sub-section of Section 3, considering that the analysis of the network traffic model with complex behaviors is carried out with powerful hybrid systems by combining more than one technique, the DPI method is examined together with the other techniques used in the analysis of network traffic. In the following sub-section, the software and hardware-based methods proposed to improve DPI performance are examined. In the last sub-section of the section, suggested methods for the analysis of encrypted traffic are presented. The advantages and disadvantages of these methods, which directly use or do not use the DPI approach or limit the usability of this approach, are discussed. In Section 4, the role of IDS in ensuring the security of IoT and SDN architectures, whose main focus is on security concerns, is evaluated, and implementation challenges of DPI on these architectures are discussed. Finally, in Section 6, DPI implementation difficulties are evaluated and deficiencies in this field are identified. In addition, new discussion issues are suggested.

2 Packet capturing and processing with commodity hardware

In this section, the problems encountered in the process of receiving and processing network packets from the NIC are determined and the solutions to these problems are presented. In addition, the platforms developed based on these solution methods are examined. Accordingly, a common software application for network packet processing using commodity hardware is examined. In this direction, Unix-based operation systems are used as an example.

The first step in packet processing is transferring the network packets to the main memory. In the Linux kernel, the network packets are stored in a *sk-buff* structure in the main memory. Also, NIC has a ring queue that stores the descriptors used for these *sk-buff* structures. This ring queue is called a *ring buffer*. When a packet is accepted by NIC, the *sk-buff* structure is mapped to kernel memory space using the Direct Access Memory (DMA) Mechanism. Then, NIC schedules hardware interrupt to notify the kernel that a packet is available. Central Processing Unit (CPU) responds to this by calling the driver's interrupt handler. Since using kernel version 2.4.20, New API (NAPI) is used by drivers [8].

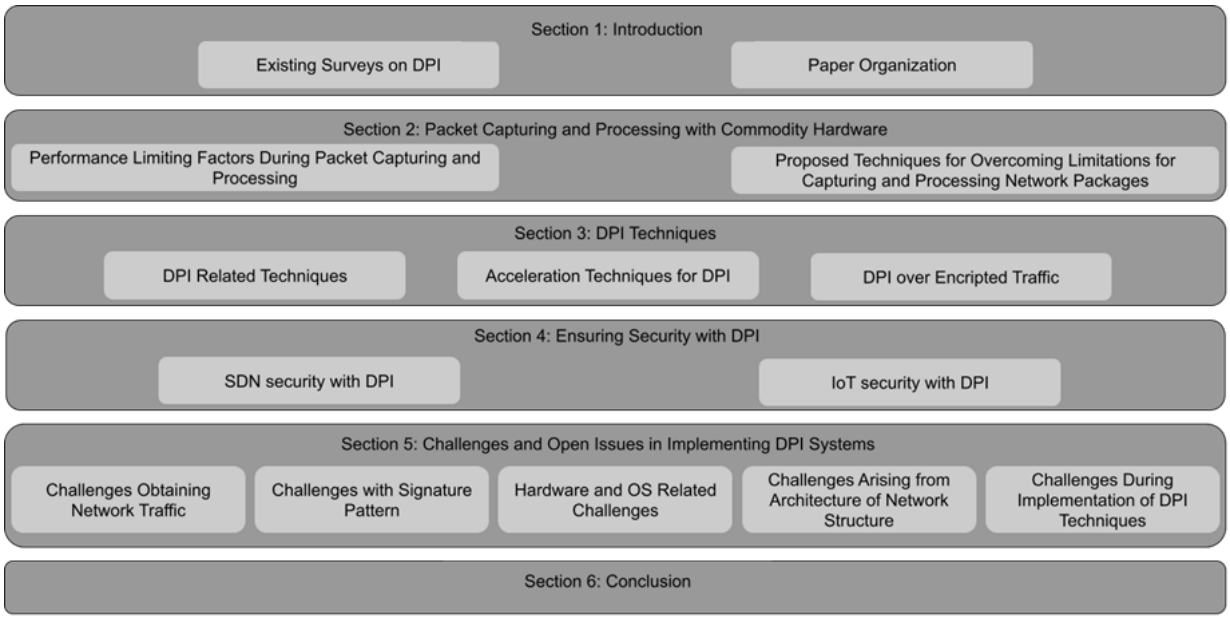

Figure 1. The structure of the survey

NIC is added to a poll list, and a soft interrupt is scheduled by the interrupt handler. Then, each of the devices in the list is polled to receive network packets from the *ring buffer* by the CPU. After the packets are processed at the network and transport layers, they are forwarded to the application layer. The *sk-buff* data is copied to the user space by using socket API. Driver must load the packet descriptor into the transmitting *ring buffer* to be transmitted of a network packet. Then, the driver notifies NIC that packets are ready to be transmitted. Finally, NIC informs the CPU via an interrupt to release the *sk-buff* structure [14,15].

NAPI contributes to the acceleration of the capturing of network packets process with two principles [16]:

Interrupt mitigation. Accepting high-speed network traffic using the traditional structure causes large number of interrupts per second. These interrupts cause the CPU to become overloaded and therefore to lose performance. The NAPI-aware driver initiates the interrupt routine when a packet RX/TX interrupt occurs to solve this problem. Unlike the traditional approach which copies and queues a packet directly, the interrupt routine uses a *poll()* function to disable similar interrupts in the future. *Polling* mode consumes more CPU cycles compared to interrupt-driven while a load of network traffic is low. However, the performance of the *polling* mode increases at higher speeds. NAPI-aware drivers fit themselves into the network load to improve performance in any case.

Packet Throttling. Packets must be dropped when high-speed traffic exceeds system capacity. Legacy drivers that do not use the NAPI approach drop network packets at the kernel level. This causes CPU cycle consumption unnecessarily. NAPI-aware drivers drop packets at the network adapter through flow control mechanisms, avoiding performance losses from redundant CPU cycles.

2.1 Performance limiting factors during packet capturing and processing

Linux networking stack is designed for general-purpose networks. Besides being used as a router, it supports many protocols at network or transmission layers. Although this design choice is suitable for running applications with a speed of 1 Gbit/s, the operating system starts to drop packets and reaches the limit when approaching 10 Gbit/s speed since it cannot process more packets [17]. NAPI technique is not suitable for capturing high-speed network traffic. In this direction, architectural problems in capturing network packets from NIC, processing them in the Linux network stack and transmitting them to the application layer cause performance losses [16].

One of the major constraints that cause performance losses is the use of main memory. The *sk-buff* structure must be allocated for each packet and released when a packet is transferred to the user level or forwarded to another endpoint. This behavior consumes unnecessary CPU cycles to transfer data from main memory to CPU. In addition, the effort to make the network stack compatible with many protocols resulted in a complex *sk-buff* structure. The *sk-buff* structure contains metadata for several protocols that are not required for packet processing. This complexity results in the creation of a very large data structure, slowing down the process. Another problem with main memory usage is that a packet has to go through different points until it arrives the userspace. This results in at least two copies for each network packet. After the packets are accepted by the NIC, the packets are copied from the DMA-capable memory area to the \textit {sk-buff} structure and then to the user-level application. A single copy of data can spend up to 2000 CPU cycles, depending on network packet length. Also, *sk-buff* conversion operations and memory allocation consume 1200 CPU cycles per network packet, and 1100 cycles are required to free the buffer [18]. *sk-buff* operations spend 63 % of the CPU cycles in receiving a single 64B packet [19].

Context switches that switch between user and kernel modes affect performance significantly. The user-level application has to make a system call for the context switch when it requires to send or receive a packet. These operations can spend up to 1000 CPU cycles per network packet [18].

The important step in the development of modern NICs is the Receive Side Scaling (RSS) Technology [19]. RSS Technology takes a load of network traffic from the NIC and shares it among the cores of a multi-core system. In this way, the load between system resources can be balanced. This allows avoiding bottlenecks in packet processing by using a single core and optimizing cache [16]. After capturing network packets using RSS technology, all network packets are concatenated at a certain point, and analyzed in the transport layer. The merging of traffic at a single point causes a bottleneck. This may also cause packet disordering [20]. System performance is adversely affected as the acceleration achieved at the driver level is lost at the user level.

Spinlock is another performance-limiting factor. During the transmission of a network packet, two spinlocks are required that control the NIC's access to the transmission queue [21]. These locks may cause congestion by preventing parallel processing when using multiple CPUs.

The main bottleneck of software architecture is about inefficient memory access. Non-Uniform Memory Architecture (NUMA) is widely used in the process of capturing and processing high-speed network traffic. NUMA architecture distributes system memory among different Symmetric Multi-Core Processors (SMPs) by assigning a memory segment to each. This architecture increases system efficiency from the point of cache misses and memory accesses [22]. However, scheduling of tasks must be done carefully when using this architecture. When NIC is plugged to the PCIe slot reserved to a NUMA node, the threads assigned to capture packets must be run on the dedicated cores for this NUMA node. Assigning these threads to another NUMA node causes to transfer of data between processors. Therefore, it reduces performance because of cache misses and access latency [16]. On the other hand, the first access to the DMA-capable memory area causes cache misses as it invalidates cache lines of DMA operations. Such cache miss spends 13.8% of the CPU cycles for a single 64B packet [19].

2.2 *Proposed techniques for overcoming limitations in capturing and processing network packets*

Various techniques are developed to overcome the problems encountered in the process of capturing network packets and transmitting them to the user area [16, 23]. These techniques are listed below:

- Preallocation and reuse of memory resources to avoid bottleneck generated by per packet allocation/releasing buffers
- *Zero copy*: Mapping the DMA-capable memory region directly accessed by NIC where an application can read and write to these regions without intermediate copies
- *Batch processing*: Copying packets into kernel or user memory by grouping them in a buffer to avoid the overhead resulted from additional calculations and function accesses such as system calls and context switches
- *Prefetching*: Loading memory locations that may be used in the processor's cache in the near future to reach them faster in case of need
- *CPU Affinity*: Determining the execution region for using the threads
- *Memory Affinity*: Determining the memory space for using the threads
- *Lock-Free Multi-Threading*: Avoiding the performance issues associated with the use of synchronization techniques such as mutexes and semaphores to ensure lock-free operation by using multiple hardware queues to allow threads to run on independent subsets of traffic
- *Compute Batching*: Applying network functions to handle a group of packets rather than single packets in order to decrease the overhead caused by additional calculations and function accesses such as context-switch and stack initialization
- *Parallel Direct Paths*: Using RSS queues and direct parallel paths between applications via the allocation of certain cores for both receiving packets from RSS queues and transferring them to the user space

There are many platforms based on the proposed solution to the problems encountered in packet processing. While the software-based platforms [23-28] developed as fast packet processing architectures use only the processing power of CPU in the packet processing, the hardware-based platforms [19, 29-32] aim to provide performance gain by executing part of the packet processing on specialized hardware such as FPGA or GPU.

Click [24] is one of the first modular software architecture used to build routers. Although it supports Linux interrupt structure which causes performance losses in packet transmission, Click uses *polling* instead of interrupt. Also, this architecture does not use *zero-copy*. Click-based Snap [29] is the hardware-based platform that offloads the part of the computational overhead to GPUs. Network packets in a batch may have different paths in Click. This separation may happen before packets reach GPU or inside GPU. This process, which causes unnecessary copying of packets that are not processed on GPU, is time-consuming due to the limited PCIe bandwidth. Snap copies only the required packets to a contiguous memory space at the host level, and then creates a group of packets that are sent to GPU memory in a single transfer over PCIe. Indeed, it is aimed to avoid bottlenecks that may occur by transferring only certain parts of the network packet processing process to GPU. Snap adds predictive bits to each packet to solve the problem of the occurence of different paths within GPU. In Snap, the *zero copy* technique is not implemented due to deviations before reaching GPU and copying only the necessary parts of the network packet for processing. PacketShader [19] is another hardware-based platform that benefits from GPU to reduce the computational load for fast packet processing. It uses a *batch processing* technique to reduce the processing load for each packet. PacketShader uses copying, which allows flexibility of the user buffer, instead of the *zero-copy* technique for better abstraction. It also facilitates the recycling of large packet buffer cells.

Netmap [25] is a userspace packet handler that does not requires special hardware and minimizes packet processing cost by using techniques such as *resource preallocation*, *batch processing*, and *zero-copy*. This framework maps the NIC rings to an equivalent number of network map rings so that the load is spread across multiple CPU cores without lock contention. GASPP [30] is the hardware-based platform that benefits from GPU to perform fast packet processing. This architecture uses the Netmap library for I/O operations in packet processing. In this way, it is possible to avoid network packet copies and context switches that cause additional overhead. In this framework, reassembling TCP streams and flow management are entirely performed on GPU. Additionally, a packet scheduling technique that eliminates load imbalance and controls flow irregularity is applied for GPU threads. Also, the *zero-copy* technique which increases throughput between devices is applied between GPU and NICs. Data Plane Development Kit (DPDK) [26] is another userspace packet handler to perform fast packet processing. This framework is developed for Intel's multi-core processors that deliver packets to applications by using acceleration techniques such as *resource preallocation*, *batch processing*, and *zero-copy*. The forwarding process of packets called the data plane operation, is performed by DPDK libraries that forward network packets to the application network stack directly without any Linux kernel overhead [33]. In addition to the libraries, DPDK also includes Poll Mode Drivers (PMD) that accesses RX and TX descriptors without any interrupts to receive, process and transmit packets to the user space. This decreases the overhead caused by interrupt operations in high speed scenarios. APUNet [31] is a hardware-based platform that leverages the power of integrated GPUs for network packet processing and uses DPDK infrastructure for packet forwarding. APUNet uses the *zero-copy* technique in the entire processing steps. This structure implements persistent GPU kernel execution to reduce communication latency between CPU and GPU. Thus, GPU threads run in parallel for constant input network packet flows. For the

solution to the cache coherency problem between CPU and GPU, synchronization of cache memory access technique is suggested by integrated GPU to present GPU's processing results to CPU at a low cost. FastClick [27] is a high-speed userspace packet processor that integrates both DPDK and Netmap into Click. Fastclick benefits from these two versions to increase packet processing speed. It also increases the efficiency of Click by using *zero-copy*, *multi-queue support*, *I/O* and *computation batching*.

PF_RING [28] performs active traffic analysis on commodity hardware and provides a performance improvement in direct proportion to the increased number of cores. This library uses PF_RING ZC drivers that implement a *zero-copy* technique to achieve maximum packet processing speeds. Packets are read directly from NIC by using these drivers. The network packets are polled from NICs by using NAPI in PF_RING architecture. Packets are copied from the NIC to the circular buffer, and the user area application reads the packets from the ring. Accordingly, the CPU is used both to transmit from the NIC and to process network packets. In PF_RING DNA (Direct NIC Access) [34], NIC NPU (Network Process Unit) architecture is used instead of NAPI. In this structure, NIC maps memory in userspace. As a result, the CPU is only used to process packets.

Vector Packet Processing (VPP) [23] is a framework capable of high-speed packet processing in the user space to benefit from general-purpose CPU architectures. DPDK or Netmap can be used as I/O nodes in the VPP. VPP resources are organized into two basic groups composed of a low-level libraries used to implement specialized network packet processing applications and high-level libraries, called plugins, that perform a specific processing task. VPP master code and plugins create a forwarding graph that defines the possible paths of a packet. Vectors are arrays located in pre-allocated contiguous memory segments, and per-vector processing is a basic principle in VPP. Vectors are efficiently managed by VPP in reusable lists before they are released. Vectors are reused and managed efficiently without releasing in reusable lists by VPP. The main innovation of VPP is that it provides performance gain by processing network packets in vectorized format. Each node of the VPP processes all packets in the vector instead of allowing each packet to traverse the entire graph. The underlying assumption of the vectorized process is that subsequent packets require similar operations. Since the instruction cache is only loaded for the first packet in the vector, other packets in the vector tend to be processed at high performance. Also, this approach provides an efficient prefetching strategy. It is known which packet data is required to process the specific feature when the node is called. Thus, (i+1)'th packet's data can be prefetched while the node is processing the i'th packet. VPP also uses the multi-loop approach to take advantage of low-level hardware support. The approach of multi-loop can be defined as a function written to process N packets in parallel. In this process, the computations for all network packets are independent of each other. With the help of this approach, CPU pipelines are allowed to fill up constantly, and the latency caused by cache miss is shared with N packets instead of a single packet.

ClickNP [32] is a hardware-based platform that aims to provide performance gain by executing part of the packet processing in FPGA. FPGAs are programmed in complex low-level Hardware Description Languages (HDL) such as Verilog or VHDL. This may cause low productivity and debugging difficulties in FPGAs. Therefore, ClickNP applies three basic approaches to overcome the programming difficulties on FPGA. In the first approach, each complicated network function is decomposed and determined as well-defined elements via ClickNP's modular architecture. Another approach is to write ClickNP elements in a high-level language that is easier to understand. The final approach is to use the high-performance PCIe channel between CPU and FPGA. This channel works with low latency and high efficiency, allowing cooperation in the processing of network packets on CPU and FPGA. Additionally, ClickNP uses a *batch processing* approach to reduce DMA overhead.

3 DPI techniques

Today's network traffic model has complex behaviors due to device mobility and network heterogeneity [1]. Analyzing this complex network traffic requires more efficient mechanisms by which multiple techniques are combined to create hybrid systems. The packet payload is used with or separately from the packet header in DPI applications such as content-based recognition, traffic classification or IDS whereas the packet header is used with its fixed format or statistical character for analysis of network traffic [5]. For this reason, it is more accurate to examine the DPI method together with other techniques used in the analysis of network traffic. In the first sub-section, DPI techniques and DPI-related techniques are examined. State-of-the-art network traffic with complex behavior causes DPI implementations to be computationally intensive and time-consuming. In this direction, improving DPI performance becomes an important working area. In the second sub-section, recommended software and hardware-based methods which improve DPI performance are discussed. On the one hand, it is important to enhance the existing literature with a well-detailed analysis of DPI applications on the network composed of encrypted traffic mostly [35]. Indeed, the methods that directly use or do not use the DPI approach or limit the usability of this approach are examined, and the advantages and disadvantages of these methods are discussed in the process of analyzing the encrypted network traffic in the last sub-section.

A classification of the literature based on DPI techniques is presented in Figure 2. This classification which focuses on the application of DPI techniques is created for the requirements of a state-of-the-art network which has complex behavior and encrypted traffic mostly. In this direction, three main categories are identified for classification DPI techniques, acceleration techniques for DPI and techniques for performing DPI on encrypted traffic. Within the first category DPI techniques, two subcategories are identified: Pattern matching and protocol decoding

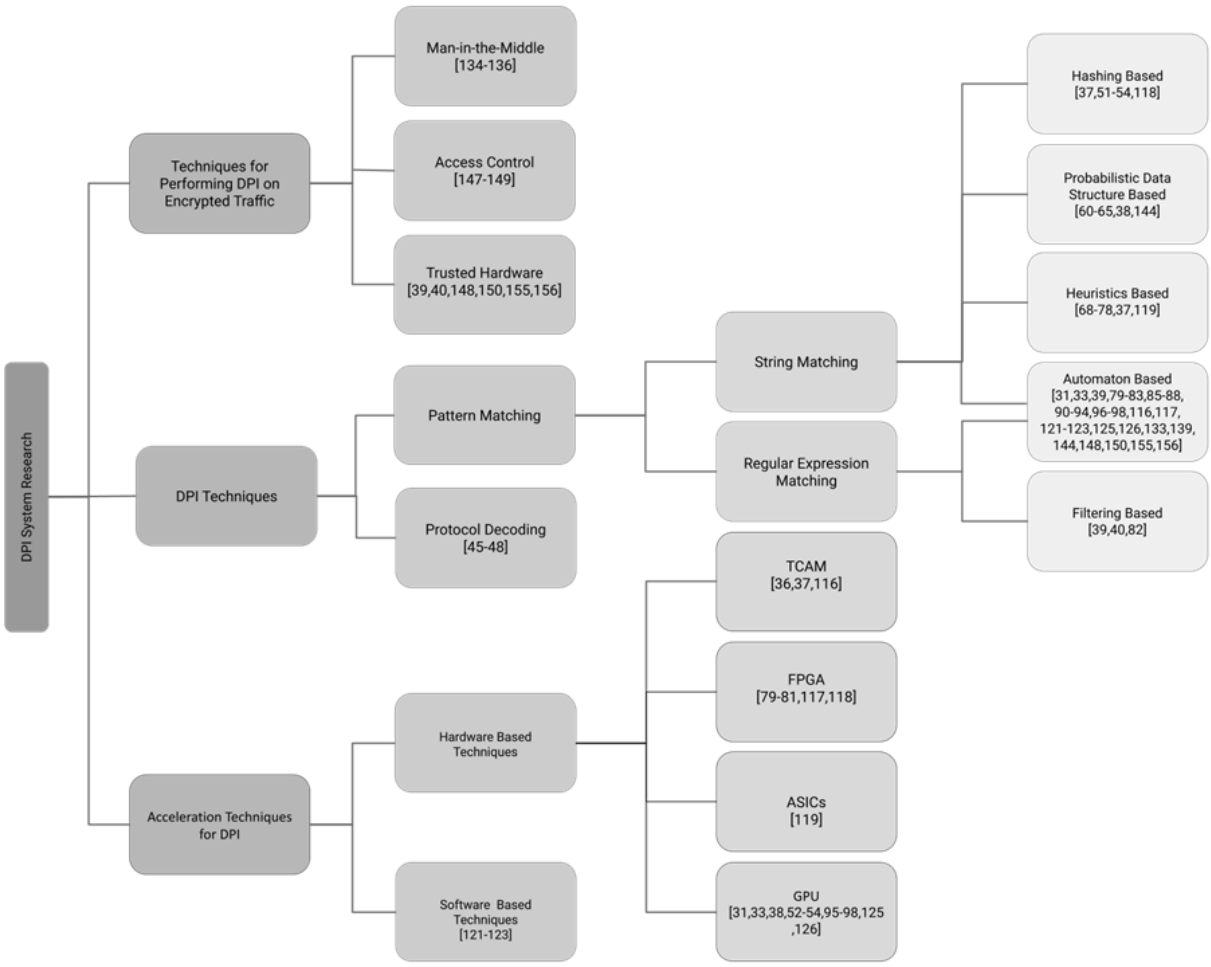

Figure 2. Overview of the surveyed research works classified according to the proposed classification

techniques. In the second category, acceleration techniques for DPI, two subcategories are identified: Hardware-based and software-based techniques. In the third category, techniques for performing DPI on encrypted traffic, three subcategories are identified: Man-in-the-Middle Attack (MITM), Access Control (AC) and Trusted Hardware (TH) techniques. With the help of this classification, it is aimed to facilitate the reader's access to references that examine a specific field. However, as a natural consequence of this classification, there are many studies in which techniques in different categories are used together. The categorization of studies numbered [36-39] and [40] can be given as an example of this case. The studies numbered [39] and [40] performed filter-based pattern matching by using the TH technique are examined in the categories of both DPI techniques and techniques for performing DPI on encrypted traffic. Likewise, the study numbered [38] performed by using Cuckoo filter (CF) [41] on GPU is examined in the categories of both DPI techniques and acceleration techniques for DPI. The study numbered [37], which proposes Ternary Content Addressable Memory (TCAM) based multiple-pattern matching algorithm that uses a hybrid model of pattern matching techniques is examined in the three categories. These categories are acceleration techniques for DPI, hashing and heuristic-based pattern matching techniques. The study numbered [36] which proposes TCAM based multiple-pattern matching algorithm is only examined in the category of acceleration techniques for DPI as it does not use any of the available DPI techniques. As seen in the classification, the most commonly used DPI technique is the automata-based pattern matching technique, whereas the filter-based pattern matching technique is the least used. Among the special-purpose hardware used to accelerate the packet processing in DPI applications, the most used hardware is the GPU, whereas Application Specific Integrated Circuit (ASIC) is the least used hardware. This classification is expanded by adding other methods related to DPI techniques. Accordingly, determining the necessary parameters for the construction of powerful mechanisms to examine the complex network traffic is an important aim of this paper.

3.1 DPI related techniques

In the study numbered [5], DPI is classified as narrow and general scope. According to this classification, generalized DPI includes an examination of both the packet payload and header. In the narrow scope, DPI represents only payload-based detection, and the detection is performed by matching the payload with signatures. On the other hand, this classification, which accepts port-based and statistical-based

analysis as a DPI technique, is not valid because it ignores the task of examining packet payload, which is the most basic feature of DPI. Therefore, DPI is considered as a part of network traffic analysis (NTA) methods in this paper. DPI and DPI-related methods are shown with references to studies numbered [5, 11] and [12] (Figure 3).

3.1.1 Port based technique

Using the port-based technique in NTA is the most traditional technique for application protocol detection. This approach is used to detect the protocol using the port fields in the TCP/UDP headers. However, reasons i.e peer to peer (P2P) applications using random port numbers, the emergence of encrypted protocols, some applications using ports assigned to other protocols for deception and replacement of one protocol by another indicate that this method is not safe for identifying the application protocol [5,10]. HTTP protocol may be considered as an example of this case. The HTTP protocol is actively used to download or upload files. Hence, it replaces FTP, which is designed specifically for downloading and uploading files. Also, many P2P applications (skype etc.) use the HTTP protocol to bypass the firewall if other ports are blocked. Additionally, HTTP is used by social networks, geomaps, and video streaming services [42]. The fact that the port-based approach is insecure and insufficient to determine the application protocol leads to the application of this technique as an auxiliary technique in the analysis of network traffic. For example, the nDPI library [42] uses the port-based approach to determine the appropriate protocol decoder. By means of this approach, protocol detection time can be reduced.

3.1.2 Protocol decoding technique

Protocol decoding is another method of analyzing network traffic. This method may be considered as the DPI technique since it performs payload inspection. Protocols can be detected by using the protocol behavior as well as the characteristic protocol headers in this method. Therefore, this technique is based upon re-establishment sessions with captured packets at the application layer [5]. Different verification methods can be applied in the protocol decoding processes. Syntactical verification aims to check the accuracy of the transferred data in terms of syntactical. For instance, the HTTP headers must be present if there is an HTTP payload. Verifiers must decode the message and ensure that the message is well-formed. Another method is protocol conformance. The process of confirming that the HTTP GET request is answered by the server in a valid manner may be considered as an example of this method. This method is more valid because it can verify the runtime behavior of the protocol when it compares to the canonical state machine. Inspecting the semantic integrity of the data is another verification method. For example, confirming whether an image object transferred by the HTTP protocol is actually an image or some other form of content [43]. The protocol decoding method, which needs a deep understanding of the application protocol, achieves high accuracy with a low false-negative rate. However, protocol decoding process is computationally intensive and time-consuming [5,12,42]. In this direction, the application of hybrid approaches in the analysis process of network traffic can be applied to improve performance.

The commercial tool PACE [44] software and the open-source nDPI library are examples of hybrid use of protocol decoding technique. PACE tool which has capable of detecting encrypted protocols uses ML techniques along with behavioral and heuristic analysis in the analysis of network traffic. nDPI is an open-source library that uses port-based approach, protocol decoding and pattern matching technique. In order to analyze encrypted network traffic, the nDPI library can perform protocol detection by using strings that match the metadata obtained from the network stream. In addition, nDPI supports DPDK which is a kernel bypass technology to minimize performance losses caused by the hardware or the operating system. There are many studies comparing the accuracy of nDPI and PACE libraries according to the degree of granularity in terms of detection of the application protocol [45-48]. According to these studies, nDPI is the library with the highest accuracy among the open-source classifiers, except for the study numbered [47]. In the study numbered [47], in which performances are evaluated according to different classification levels, nDPI is the best performing classifier at the protocol level, whereas PACE is the most successful technique at the application level. PACE is a commercial tool that cannot be accessed by the entire research community. nDPI is the most successful classifier among the open-source tools.

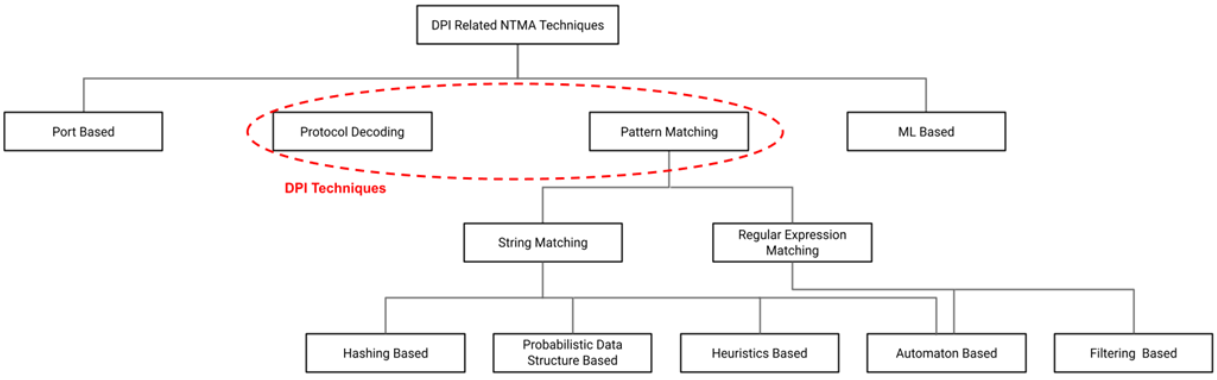

Figure 3. DPI related network traffic analysis techniques

3.1.3 *Pattern matching-based technique*

Pattern matching-based DPI methods can be implemented using hashing, heuristics, automaton, filtering or probabilistic data structure-based algorithms. These algorithms are based on string or *RE Matching* methods.

Hashing-Based Technique. Hash-based algorithms [49, 50] use the *string matching* method. This approach compares hashes instead of the contents of the packet payload with the pattern character by character. A hash value is calculated for each pattern of length m. At the same time, the hash value of the examined m-long substring is also calculated. If any pattern-substring matches are detected while the pattern is sliding over the target text, the pattern and substring are compared on a byte basis for verification. In addition to the study [51] in which the matching performance of hash-based Rabin-Karp (RK) [50] algorithm on IDSs is evaluated, there are some DPI applications that evaluates the matching performance of the parallel application of this algorithm on GPU [52, 53]. In the multi-pattern matching version of the RK algorithm performed on the GPU, each thread compares the hash code for all patterns starting from the position corresponding to the thread index. Thus, the matching process lasts too long, and RK is not an efficient algorithm for multi-pattern matching [53]. Additionally, the performance comparison of the RK from the point of execution time for variable-size network traffic is presented for both serial and parallel implementation in the study numbered [54]. RK algorithm parallelized on GPU outperforms the serial implementation encoded on CPU and increases the pattern matching speed.

Probabilistic Data Structure Based Technique. Bloom filter (BF) [55] is a probabilistic data structure used to represent a set for the purpose of performing membership testing. Using this data structure makes possible to query whether an element is present in the set at a low cost. BF does not produce false negatives. Indeed, if BF produces a result such as "The element is not present in this set.", it is true. However, there is also the possibility of producing false positives. BF may produce a result such as "Element is present in this set." for an element that is not present in the set. BF does not support delete operation. There are improved versions of BF which support deletion provide a better location or reduce the cost of space [56, 58]. A quotient filter (QF) [59] is a hash table that records fingerprints of elements to support deletion. In this data structure, encoding each entry is required additional metadata. This requires 10-25% more space than BF. In this data structure, table entry codes must be decoded before reaching the target element. Whereas the hash table is filled, operations increase at a similar rate. When the fill rate of the hash table exceeds 75%, the matching performance of the data structure drops dramatically [41]. CF is the membership query data structure in which elements can be dynamically added or removed. The biggest challenge facing CF performance is the use of three hash functions which causes additional computations. Quotient-based Cuckoo filter (QCF) [60] which uses only two hash functions has less computational overhead than standard CF. This filter has higher insertion, query and deletion capability than CF.

Probabilistic data structures are used as a matching tool in many DPI applications [60-65]. In the study numbered [61], Prefix BF (PBF) and Chain Heuristic methods are proposed, which allow pattern matching without defragmentation in order to reduce the required storage space. The PBF data structure allows the detection of the prefixes of patterns. In this way, it is possible to detect patterns on more than one network packet. Chain heuristic increases system throughput by reducing the false positive rate of PBFs without using any additional memory. In the study numbered [62], a new BF architecture is proposed by using the pipelining technique which notably reduces the overall power consumption of the BF. In the first stage of a two-stage pipelined BF, hashes are always calculated. If a match is found between the input and the pattern, the hash values are calculated in the second stage. Implementing a small number of hash functions in the first step increases power savings. In the study numbered [63], a new BF is proposed in which both two memory addresses are compressed into one I/O block of main memory. With the help of this data structure, the number of memory I/O required for the membership query is reduced. Accordingly, the average query latency is also significantly reduced. In the study numbered [64], the DPI application is performed by using QF. The results obtained from the real dataset show that QF achieves higher efficiency (30%-75%) and improves false positive rate compared to BF. In the study numbered [65], CF is used as a DPI matching tool. The developed system provides a significant time saving of 93% compared to BF and 87% compared to QF. In the study numbered [60], CF performance is tried to be increased with a new proposal called QCF. The Analysis shows that applying QCF in a DPI application results in time savings of up to 77% at CF and up to 98% at BF and QF.

Heuristic Based Technique. Heuristic-based matching algorithms use string matching. The primary principle of this approach is to jump as many payload characters as possible by using some heuristics to speed up the matching. Single-pattern matching algorithm Boyer-More (BM) [66] and multi-pattern matching algorithm Wu-Manber (WM) [67] are examples of heuristic-based matching algorithms. BM is a heuristic-based algorithm that improves the performance of the search model by making situational skips. BM algorithm which carries out control from the right to the left performs shifts according to the rules of “Good-Suffix” (matched suffix of target text and pattern) and “Bad-Character” (unmatched character of target text and pattern). Besides the studies focusing on reducing the number of character comparisons to increase the performance and efficiency of IDSs [68-70], the studies applying the BM algorithm to detect known attack patterns [71-74] are presented as an improved version of the BM algorithm. An important aspect that limits the performance of the BM algorithm is that it cannot process multiple patterns in parallel. In this direction, WM which is developed as an advanced version of the BM algorithm has the ability to process more than one pattern

simultaneously. WM consists of two phases called preprocessing and scanning. In the preprocessing phase, basic calculations required for the scanning phase are made, and three tables are created namely SHIFT, HASH and PREFIX. The created tables are used for pattern matching in the scanning phase. In order to improve the performance of IDSs, the studies numbered [75], [76] and [77] focus on reducing the number of CPU cycles by reducing the number of unnecessary matching attempts, and they are presented as an improved version of the WM algorithm. In addition, the study numbered [78] is presented as an advanced version of the WM algorithm in order to reduce the performance losses caused by short patterns that result from short shift distance. The study aims to reduce the effect of short patterns that limit performance by splitting patterns into the different pattern clusters according to their lengths and processing these clusters sequentially.

Automaton Based Technique. Automaton-based approaches can use both *string matching* and *RE Matching* methods. The matching process in DPI is computationally intensive and time-consuming as processing each byte in the payload requires one or more memory accesses. This situation is negative for the DPI process. Therefore, the DPI performance is highly dependent on the pattern matching throughput indeed the performance of the FSM [5].

There are two types of FSMs; Nondeterministic Finite Automata (NFA) and Deterministic Finite Automata (DFA). In fact, the two FSMs are equivalent. An equivalent DFA with NFA can be created, and this DFA accepts the same set of patterns. The main difference that distinguishes DFA from NFA is that any DFA state has a single pass for each character leaving to the specific state. Any NFA state can switch to different states more than once for the same character. Accordingly, a DFA can have only one active state at any one time while a NFA can have more than one active state. As a result, a NFA and a DFA have completely opposite characteristics in memory bandwidth requirement. A DFA is a memory-intensive structure while a NFA is a computationally intensive structure. Most current research aims to strike a balance between storage and performance [79-81].

RE defines a search pattern such as languages, or a set of strings. This structure can represent a set of exact strings while the exact string can represent only one string [5]. REs are widely used in many open source and commercial DPI applications under favour of their powerful and flexible detection capability [82-84]. A new RE-based DPI system that can process out of order packets without performing packet buffering and stream reassembly is proposed in the study numbered [85] that aims to improve the accuracy and speed of pattern matching, besides the studies numbered [86], [87] and [88] focusing on the creation of memory-efficient architectures for the RE pattern matching process. Exact-match strings, the simplest type of REs, are fixed-size patterns. Automaton-based Aho-Corasick (AC) [89] algorithm, which has a faster matching power than complex REs, is widely used for *string matching* by means of its easy implementation. Application of the AC algorithm in the pattern matching process makes the cache space useless for large state transition tables. As a result of this situation, the matching speed decreases for large pattern datasets. Compressing a transition table in order to reduce the memory requirement and effectively use cache is one of the research topics of the AC algorithm. Whereas some improved AC algorithms [90-93] focus on reducing memory space required for storage of automata, the study numbered [94] proposes a variable stride pipelined Aho-Corasick Deterministic Finite Automaton (AC-DFA) to reduce the number of memory accesses and energy consumption in the pattern matching process. In addition, DPI is applied by using the AC algorithm in IDSs developed on the study numbered [29] and [31] platforms. However, the memory requirement of the AC algorithm using the large state transition tables and the slowing of the matching speed for large pattern datasets show that this algorithm cannot be a suitable solution especially for GPU-accelerated DPI applications as discussed in the studies numbered [95-97] and [98].

Filtering-based Technique. A filtering-based approach relies on excluding the parts of the input data that do not match the pattern. Multi-pattern matching algorithm DFC [99] which increases pattern matching performance by significantly reducing the number of memory accesses and cache miss compared to the AC algorithm is an example of filtering-based matching algorithms. DFC consists of three stages called initial filtering, progressive filtering and verification. In the initial filtering stage, A direct filter (DF) that does not require hash computation and uses a sliding window is constructed to exclude parts of the input data that do not match the pattern. In the progressive filtering stage, multiple layers of DF are constructed to categorize patterns based upon length and filter the window incrementally. In the last phase, the input is compared with the patterns to verify whether an exact match occurs. The studies numbered [39] and [40] in which the inspection is executed in the secure enclave can be given as examples of the use of the DFC algorithm. Also, in the study numbered [82], it is used a filtering system for *RE Matching* to exclude flows that include no segment characteristics of RE.

3.1.4 Machine learning based technique

Some legal restrictions to prevent access to the packet payload due to some reasons such as protecting the privacy of users [100] encourages researchers to use different methods for examining network traffic. The statistical methods collect payload-independent variables i.e port numbers, packet length, flow start/stop timestamp and inter-arrival time of packets in a stream to analyze the network traffic and predict which application or protocol the traffic may belong to [5, 101]. In many studies, Machine Learning (ML) algorithms are used [102-104] besides statistical methods [105-107] to classify network traffic. The studies numbered [103] and [104] use TLS header information and DNS data as well as flow metadata in the analysis of network traffic. ML is a subset of Artificial Intelligence (AI). AI approach aims to implement human-like AI by creating a set of rules. Although this approach successfully completed well-defined tasks, it is insufficient to perform more

complicated processes i.e image processing. ML is developed to overcome such challenges. DL approach which is a subdomain of ML uses Deep Neural Network (DNN) to get data representation at each layer [108]. The relationship between AI, ML and DL is shown in Figure 4. The number of layers used to model the data is defined as the depth of the created model. DL models may have hundreds of consecutive layers to handle complex tasks. Traditional ML algorithms are inadequate for the analytical requirements of modern networks. This situation increases the popularity of DL in the application of NTMA techniques [1].

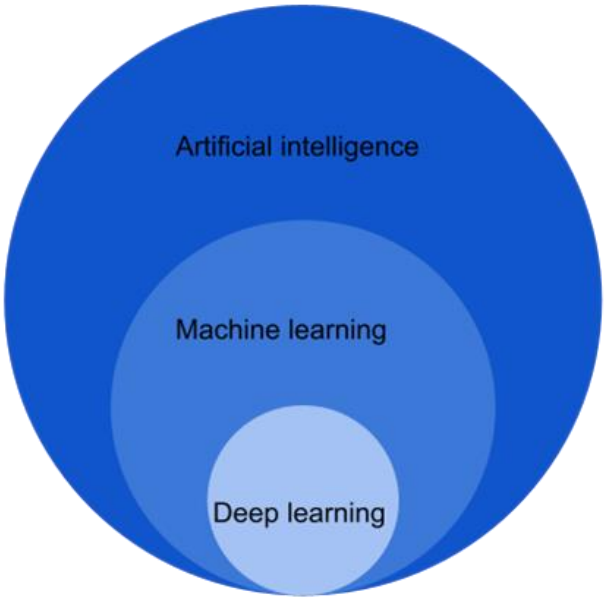

Figure 4. Relation between AI, ML, and DL

In a ML system, it requires expertise to create a feature extractor that converts raw data into a convenient representation [109]. Because DL algorithms use automatic feature learning, they remove the requirement for feature engineering in traditional methods. This characteristic of DL models is important for NTMA methods as a major part of the network traffic are unlabeled or semi-labeled [110]. Some useful features that cannot be detected by manual feature engineering can be detected by DL algorithms. Accordingly, many NTMA applications [111-114] implement by using DL algorithms.

There are many disadvantages that affect performance in the process of performing NTMA techniques using ML approaches. The success of ML methods highly depends on the quality of data used for training. The networks trained with low-quality data result in unsuccessful NTMA implementations. In this process, DL algorithms need large amounts of network traffic data. The most of data are unlabeled or semi-labeled in the network applications [110]. However, labeling this data is time-consuming and computationally labor-intensive. Accordingly, training DL algorithms with a great number of training data and variables requires rich devices in computing, memory and power resources. On the other hand, resource-constrained AI-based devices such as IoT devices are insufficient to fulfill this requirement. Also, ML-based models need to be retrained at frequent intervals to adapt to new situations in the network such as security violations or network behavior changes. Besides, high complexity in the training phase causes DL models to consume too many resources and time. Accordingly, DL models need to be improved from the point of time and resource consumption. Another issue to be evaluated in process of training the networks is about the use of the dataset. DL techniques must be trained by specific datasets collected from the network traffic and labeled with high accuracy due to the heterogeneous nature of networks. Accuracy of DL techniques trained by public datasets may decrease in several networks [1,115].

3.2 Acceleration techniques for DPI

Many techniques are proposed to accelerate packet processing in DPI applications. The hardware-based methods use special-purpose hardware such as FPGA, TCAM, ASIC, and GPU to reach high matching speed with device parallelism. The methods that propose TCAM as a hardware acceleration tool are based on TCAM's parallel processing capability [36,37,116]. The studies numbered [37] and [116], which propose TCAM based multiple pattern matching algorithms that allow multiple characters to be processed at once, aim to increase the matching speed by reducing the number of TCAM searches. The study numbered [36] is another schema that proposes TCAM based multiple-pattern matching algorithm. Unlike the algorithms proposed in studies numbered [37] and [116], this algorithm does not use any of the existing DPI techniques for packet processing. In this algorithm, patterns with a pattern length less than or equal to the specified TCAM width are classified as simple patterns. The pattern matching process for the simple patterns is as follows: The first w byte in a packet is mapped into TCAM to detect a match. Then, one byte is shifted, and the process is repeated. This iteration is performed for the entire packet. The first step of the pattern matching process for long patterns is to identify the prefix and the suffix patterns. Then the prefix patterns are combined with the corresponding suffix patterns. Three tables are stored in memory to perform this process. These tables are: Pattern table, Partial Hit List and Matching Table. FPGA is another hardware solution used for pattern matching. FPGAs consist of programmable logic blocks and interconnections between these logic blocks. These logic blocks and interconnects can be reprogrammed according to the desired purpose. Thanks to its reprogrammable and parallel processing capability, FPGA is one of the important solutions used for pattern matching. In order to achieve high pattern matching speed with device parallelism, many studies using FPGA aim to maintain a balance between storage and performance [79-81,117,118]. ASIC is an integrated circuit designed to perform a specific task, unlike general-purpose microprocessors. ASIC can run faster compared with programmable logic devices or logic integrated circuits. Despite their small size and low energy consumption, ASIC production is an expensive and time-consuming process. In the study numbered [119] developed ASIC for IDSs, FNP multiple pattern matching algorithm

which reduces the number of memory accesses and improves pattern matching performance is presented. FPGA has a more flexible structure than ASIC. However, programming difficulties of the FPGA prevent its widespread use [120].

The software-based methods using general-purpose processors provide greater flexibility and programmability compared with the hardware-based solutions. In the studies numbered [121,122] and [123], approaches that combine the advantages of NFA and DFA applications are proposed in order to benefit from multi-core architectures efficiently. It is aimed to increase the matching performance by using the existing parallelism provided by multi-core processors with the algorithm proposed in the study numbered [121] in which complex REs are divided and assigned to different cores. In the study numbered [122], a new pattern matching algorithm HBM is proposed by combining DFA and NFA. According to this algorithm, the pattern matching process has two stages: Head DFA (H-DFA) and body NFA (B-NFA). While H-DFA processes the pattern up to a certain length, it uses less memory space than AC algorithm. In the B-NFA phase, the matching process is applied over the entire pattern by using a variable stride data structure. Also, Single-Instruction Multiple Data (SIMD) approach is used for accelerating the matching process. In this direction, the HBM algorithm focuses on reducing the used memory space and increasing the matching speed. Contrary to the HBM algorithm which creates the head-body finite automaton according to the predefined depth values, the FHBM algorithm proposed in the study numbered [123] divides the head and body parts according to the head size. Accordingly, this study proposes an algorithm focusing on increasing the efficiency obtained in the pattern matching process by providing a more flexible structure in terms of AC-DFA partitioning.

After the development of high-speed networks, traditional approaches using CPUs become inadequate in meeting network packet processing speed requirements. Accordingly, GPU with superior parallel processing capability can be used to provide high matching speed compared to CPUs [97]. GPU is a multi-threaded and multi-core processor with high computational power. Therefore, GPU is well-suited to handle parallel computing problems with high arithmetic intensity. In 2006, NVIDIA introduced CUDA which is a parallel computing architecture to solve many complicated computing problems more efficiently than CPU. CUDA leverages the GPU's capabilities to increase computational performance [124]. CUDA-supported GPU cards consist of a Set of Stream Multiprocessors (SM), and each SM contains a Set of Stream Processors (SP). SMs are designed with a Single Instruction Multiple Thread (SIMT) architecture in order to run hundreds of threads simultaneously. In any clock cycle, each SP executes the same instruction by processing different data. The threads are organized according to warps. The warp is a structure that consists of 32 parallel threads, and it is the time unit of SM. A warp performs only one command at a time. Therefore, a high level of performance can be reached when all threads in the warp have the same instruction path.

GPU has performance sensitivities as well as its superior performance in computationally intensive tasks and parallel computing capability. The organization of warps significantly affects performance. The maximum performance can be achieved when threads in the same warp execute the same instructions. Otherwise, computations in the warp are done sequentially, resulting in processing latency. The warp executes each branch serially if threads in the warp have different execution paths due to any sort of divergent conditional branching. This situation which is called as thread divergence causes increasing the total time of executing instructions in the warp. Bank conflict is another factor that causes performance sensitivity. The multiple threads requesting access to the same bank at the same time results in bank conflict, and this situation increases execution time. Another factor limiting performance is the difference in access latency of GPU memory areas used in the packet processing. CUDA threads can access different memory areas while they are executing. Each memory area has its own individual purpose, accessibility and speed of access. Each thread block has a shared memory that can be accessed by all threads which belong to that block has the same lifetime as the block. It is possible for all threads to access the same global memory. The shared memory has less memory than the global memory. Like any device memory hierarchy, the local memories on GPU have less access latency than the global memories. Accordingly, executing the packet processing in shared memory instead of global memory reduces packet processing time [98].

In DPI applications, high-density computing and the speed factor are important. Therefore, GPU usage is common due to its high computational power and convenience for parallel computing problems [38,96-98,125,126]. In DPI applications based on GPU, performance sensitivities of this hardware are mostly emphasized, and it is aimed to make maximum use of GPU's parallel processing capability. Whereas the studies numbered [98] and [125] focus on reducing the used memory space, another study numbered [97] aims to increase the performance by reducing the processing load of GPU by using a pre-filtering mechanism on the CPU. In addition, BF, QF, CF and QCF probabilistic data structures are used as matching tools in many DPI applications. In the study numbered [38] performed by using CF on GPU, the parts of global memory that threads in the same block frequently access are detected and transferred to the shared memory. Then, it is aimed to reduce the execution time of threads by using the shared memory instead of the global memory. This approach only detects memory regions that threads access frequently. Infrequently accessed memory regions are accessed by using global memory. As a result, this approach cannot guarantee that all threads access only shared memory. Therefore, memory access latency in the study numbered [38] is much higher than the study numbered [98] that uses the P3FSM algorithm encoding the DFA state transition table to fit in the shared memory of GPU.

3.3 DPI over encrypted traffic

According to Google's September 2022 transparency report [35], 95% of traffic using the Chrome platform is HTTPS. The applications that require detailed analysis of the packet payload are completely disabled by the TLS protocol. Examples of these applications that are negatively affected by HTTPS are content filtering, IDS/ Intrusion Prevention System (IPS), Data Loss Prevention (DLP), fraud detection, parental control, ad blocker, transcoding and compression [127]. Additionally, the inability to detect user and session identifiers, URLs or timestamps due to encrypted traffic reduces the efficiency of repair services in troubleshooting difficulties or application layer problems. RFC8404 [128] points out that information provided by the application is insufficient in the absence of network packets to analyze. Therefore, new approaches are required to use existing techniques such as DPI to analyze encrypted traffic. Another research topic in the inspection of encrypted network traffic is the security of Middle Boxes (MB). With the advent of Network Function Virtualisation (NFV), dependence on specialized and expensive hardware decreases in the distribution of MBs responsible for performing network functions such as IDS or firewalls. The distribution of software-based MB functions starts to gain importance. Accordingly, MBs move from a hardware-based device to a cloud infrastructure that provides more flexibility. However, transferring data that is internally examined to the MB provider for processing raises security and privacy concerns [129,130]. This prompts researchers to develop different methods for examining network traffic.

This section focuses on suggested methods to perform an inspection on encrypted traffic. In this regard, inspection methods on encrypted network traffic are shown with reference to the studies numbered [127] and [131] (Figure 5).

The classification in Figure 5 is based on whether the encrypted network traffic is decrypted or not. MITM, AC, and TH methods analyze network traffic by decrypting encrypted traffic, whereas Searchable Encryption (SE) and ML approaches perform their investigations over encrypted network traffic. The details of these approaches are presented in this sub-section, with the exception of the ML approach. The application of NTMA techniques using ML approaches is discussed in sub-section 3.1.4. However, it is important to evaluate the ML technique together with techniques developed for the analysis of encrypted network traffic. Fort his reason, in this sub-section, a comparison of the ML technique with the others is presented. In addition, a comparison of all the techniques developed for the analysis of encrypted network traffic is presented

3.3.1 MITM technique

MITM technique is implemented by establishing two TLS sessions, both between client-MB and MB-server. This approach requires the client to install the MB's root certificate [132]. The root certificate allows the MB to identify itself as a server to the client by copying and signing a new certificate based on the server's credentials. In this way, encrypted traffic originating from the client can be decrypted and analyzed by MB [133]. Then, MB re-encrypts data on behalf of the client and transmits it to the server via the second TLS session. MITM technique for encrypted traffic analysis is widely used in the applications such as antivirus and parental control, also incorporates network solutions [134-136]. There are also widely open-source tools such as MitMProxy [137] and SSLSplit [138]..

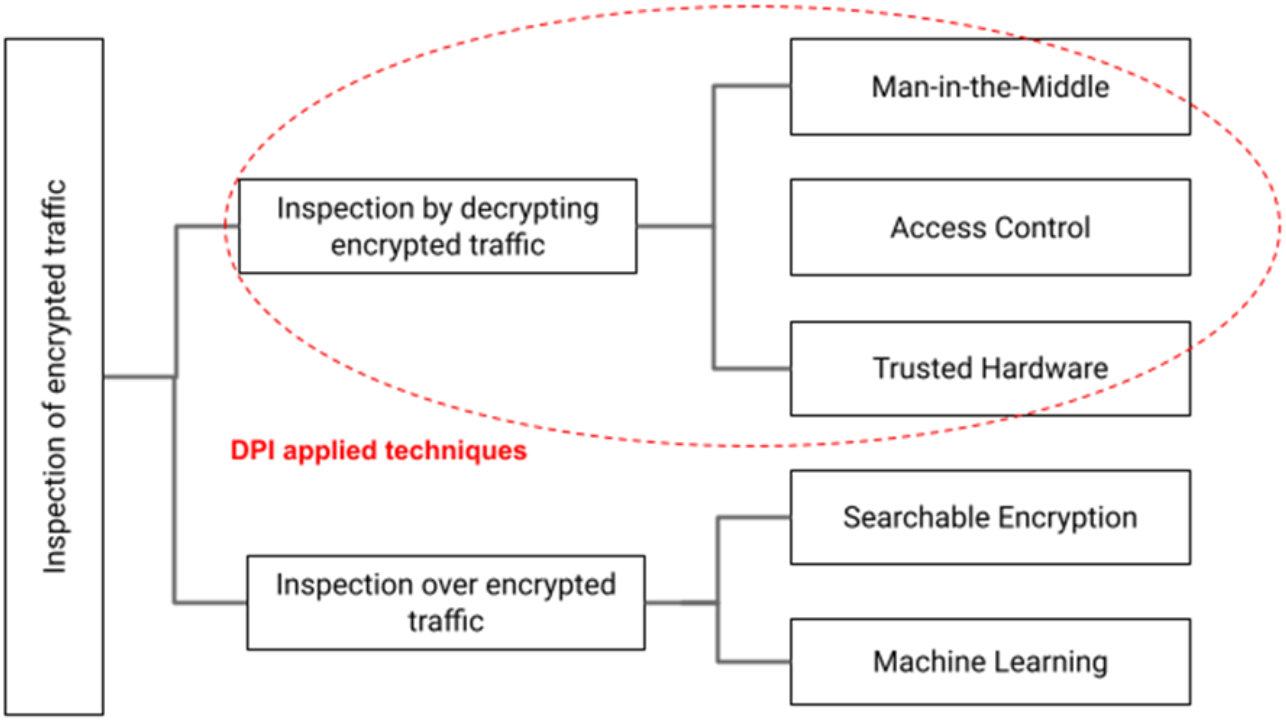

Figure 5. Methods for inspection encrypted traffic

3.3.2 SE technique

SE method tries to detect malicious traffic via token matching without decrypting the encrypted traffic. In this technique, the SE schema is used to perform a mapping between encrypted keywords and rule sets [127]. The study numbered [139] is the first DPI schema using the SE technique and protecting privacy. In this schema, the client establishes a TLS connection with the server and a second connection for token matching. Both of these connections are routed through MB. MB contains a rule set that is encrypted using a key derived from the session key of the TLS session. The client tokenizes and encrypts the message by using the same key. Then, it transmits the tokenized message by using the second connection. MB tries to match tokenized traffic with encrypted rule sets. When a match is found, traffic is considered malicious and blocked. In this schema, MB inspects only tokenized data. Thus, the SE technique has very limited functionality compared to the inspection performed on the unencrypted traffic. The study numbered [130] is designed as the first system handing over MBs to cloud providers while keeping network traffic private. In the study numbered [139], PrefixMatch which is a new and fast encryption schema is proposed in order to reduce the overheads arising from the cryptographic computations. In addition, the limited functionality of this study is extended by supporting different SE schemas that can be used for different MB services such as parental filter, IDS or firewall. The study numbered [140] which reduces the overhead from cryptographic computations in the study numbered [139] and memory space needed to perform MB functions uses decryptable SE [141] schema based on public-key cryptography. This schema replaces the existing TLS protocol by using a different approach to deploying the public key operation. As for the study numbered [142] presents another schema by using public-key encryption. Unlike the symmetric encryption schemas used in the study numbered [139], public-key encryption schemas increase functionality and reduce efficiency and overhead from the cryptographic computations during the setup phase. In contrast to the study numbered [140], the study numbered [142] uses public-key operation only in ML-based inspection for malware detection. The study numbered [139] uses a garbled circuit to generate encrypted rules used to examine encrypted traffic. However, setup latency and overhead size caused by garbled circuits are high. In this approach, computational and communication overheads are repeated as encrypted rules are created for each new session. In the study numbered [133], an obfuscated rule generation technique is proposed that provides better performance in encrypted rule generation and does not use garbled circuits. The intermediate values which can be reused in subsequent sessions are generated for repeated tokens by using a reusable obfuscation mechanism in this approach. Accordingly, this approach provides performance gain by reusing the rules, and it is aimed to accelerate the encrypted rule generation process by reducing the computational and communication overheads in the study numbered [139]. The study numbered [143] provides hiding of rule sets from MB with significantly improved performance over the studies numbered [133] and [139]. Rule hiding is enabled when MBs are in untrusted cloud providers. Two schemas that host MBs in cloud providers and implement different approaches in these providers are presented in the studies numbered [144] and [145]. The study numbered [145] is based on two cloud systems where each rule is XORed with a random string and is split into many blocks to MBs located in one of the cloud systems. The cloud systems compute blocks together to be able to inspect the network traffic. The schema which has a two-layered architecture presented in the study numbered [144] is deployed on two non-colluding servers. The first layer filters the legitimate packets using BF. The second layer supports the exact rule matching for network packet analysis by using the conjunctive SE schema [146]. In addition, the results of the inspection are verified with the cuckoo hashing method.

3.3.3 AC technique

AC technique that allows traffic to be decrypted is an approach based on the client and the server accountability. In this technique, the client and the server are aware of all the MBs deployed between each other. The prominent feature of this technique is that MBs are visible to endpoints which decide MBs' right to access encrypted traffic. A flexible control mechanism that is provided with AC decides whether the data can be decrypted or not [147]. The study numbered [147] allows to exchange read and write secret keys besides session keys by modifying existing TLS protocol so that the client, MB and server establish a secure and authenticated channel. In this approach, MBs' access rights to encrypted traffic are determined by read/write keys. The inspection is performed on segmented encrypted traffic by using these keys. However, the main problem with this approach is that it changes TLS protocol. The study numbered [148] which does not change the underlying TLS protocol provides scalability for applications to use cloud-based MBs. It also leverages the secure enclave of the Intel SGX trusted hardware to isolate MB from cloud infrastructure. Another schema that aims to make MBs visible and auditable is presented in the study numbered [149]. MB certificates used to encrypt the channel are defined for each TLS segment to eliminate insecure practices i.e installing private root certificates by users or sharing private keys by servers with third parties. These certificates are logged on an MB transparency log server so that MBs can be audited. In addition to auditable MBs, the MaTLS protocol which ensures the security of MBs is designed in this study. This protocol includes security objectives such as server, MB and data source authentication, segment and individual secrecy and path integrity.

3.3.4 TH technique

TH is a technique where inspection is performed in the secure enclave of the Intel SGX trusted hardware while maintaining confidentiality. The basic principle in this approach is that the client or the server shares the session key with the secure enclave in the MB. The process of

decryption, inspection, and re-encryption are implemented in the secure enclave. MBs and service providers holding these MBs cannot access decrypted data [131]. The studies numbered [39] and [150] can be given as examples of the efficient schemas in which the inspection is executed in the secure enclave. In these studies, DPI is applied as the MB function. In the study numbered [150], an intrusion detection and prevention system (IDPS) is implemented by using the AC algorithm. In the study numbered [39], IDS is implemented by using DFC algorithm, which increases pattern matching performance by significantly reducing the number of memory accesses and cache miss compared to the AC algorithm, and PCRE2 library [151] for RE matching. In the study numbered [150] which presents a Click-based interface for designing and implementing network functions, the inspection of network traffic is performed in the secure enclave in MB located on the client, unlike schemas where MB is deployed between the client and the server. However, a decentralized system model which requires SGX-enabled hardware support for all client systems has high deployment costs [39]. The study numbered [39] which offers programming abstraction for MB developers to securely process encrypted traffic uses high-level APIs in a safe language, RUST [152] and boundary checking mechanism [153] to mitigate potential memory safety attacks [154] on enclave code. There are also studies aimed at the efficient deployment of MBs to untrusted cloud providers and providing secure MB functions by applying the TH technique in the NFV environment [40,155,156]. The study numbered [155] which uses the DPDK library for I/O operations in network packet processing uses the AC algorithm as a MB function in DPI implementation. This schema that allows the cloud provider to see only encrypted traffic protects rule sets and network function codes from the cloud provider. The study numbered [156] which presents a Click-based interface by using ready-to-use elements and C++ extensions to design and implement various network functions leverages SCONE [157] which is a shielded execution framework based on Intel SGX to securely handle network packets. Additionally, the Hyperscan RE library [158] is used to implement DPI as a MB function in this study which uses the DPDK library for I/O operations in the network packet processing. The study numbered [40] improves functionality and security of TH-based schemas such as the studies numbered [148], [155] and [156], and SE-based schemas such as the studies numbered [130], [139], [140], [142] and [145]. It maintains confidentiality of the network traffic metadata such as packet size, count and timestamps in addition to the L4 payload of the network packet. The study which uses the DFC pattern matching algorithm in the application of DPI as MB function suggests using DPDK and Netmap libraries for I/O operations in the network packet processing. In addition, the study numbered [148] using the AC technique benefits from Intel SGX trusted hardware to isolate MB from the cloud infrastructure.

3.3.5 Comparison

MITM technique is safe as long as a root certificate is securely stored, a current TLS version is used, and decryption is done in a controlled manner through whitelists that contain data not to be decrypted. However, this approach may weaken TLS security on several counts. The proxies that implement the MITM technique are responsible for the certificate validation process while communicating with the server. The issues such as accepting any certificate which certificate chain cannot be verified or trusting an expired Certificate Authority (CA) list may weaken TLS security. Some distributions may be insecure due to weaknesses in implementing core protocols, i.e allowing deprecated cipher suites, and MITM proxies not being updated with patches against newly discovered vulnerabilities. Also, simple MITM attacks are possible when the same key is used on all systems using the same product. Creating product-specific root certificates dynamically and protecting the associated private keys are important for this reason. Another security problem of the MITM technique is related to accessing decrypted data. It is difficult to determine devices to allow access and monitor network traffic in a large network with heterogeneous network devices [133, 134, 159].

There are many studies examining the effect of network solutions by using the MITM technique on TLS security [134-136]. In the study numbered [134], a TLS testing framework is proposed to analyze antivirus and parental control applications by using the MITM technique for encrypted traffic analysis. The security vulnerabilities of TLS proxies analyzed against known attacks are detected. The support of proxies for different TLS versions is tested, and the shortcomings of these proxies in the certificate validation process are investigated. Additionally, the matters of whether applications dynamically generate product-specific root certificates and protect the corresponding private keys are investigated in this study. The study numbered [135] examines the prevalence and impact of HTTPS interception. TLS handshakes generated by browsers, security products and malware are characterized. The heuristics are created that enable web servers to detect HTTPS interception and TLS proxies. Then, these heuristics are implemented in the different networks. The negative effects of solutions that weaken TLS security by using the MITM technique for encrypted traffic analysis such as corporate MBs and antivirus control applications on connection security are also discussed, and the security vulnerabilities of these solutions are emphasized. The study numbered [136] presents a framework to detect potential security vulnerabilities by analyzing TLS Proxy features of different network devices. The study which focuses on the risks and vulnerabilities of using TLS proxies explores security issues related to the protection of private keys, patching against known attacks, certificate validation and the use of appropriate TLS version and CA trusted lists.

AC technique provides more flexibility to protect the confidentiality of data compared with the MITM technique. It does not require special cryptographic primitives as in SE. On the other hand, it changes the TLS protocol as the client, the server and all MBs must agree on schema to be used. Moreover, it is not clear how to set the access policy for MBs in the AC technique alike the MITH technique [131].

TH method does not require changes in TLS protocol like SE, MITH, and ML techniques. However, the session key must be securely transferred by the client or the server to trusted hardware. Without the disadvantages of the MITM technique, this method allows MB to examine encrypted traffic in a protected environment. Therefore, decrypted packets do not leak out of the environment. The secure enclave has limited storage, memory and computational capabilities. For this reason, it is not practical for TH technology to perform any inspection in the secure enclave. The main concern of TH-based solutions is the security vulnerabilities due to the success of attacks such as side-channel [160] against Intel SGX. The hardware deployment and the cost of trusted hardware are also among factors limiting to use of this technique [131,155].

SE-based techniques cause additional overhead in the setup process as they require the generation of tokens and encryption of rule sets. However, the matching is done in only MB, so the setup phase of SE techniques is more inconvenient than the inspection phase due to the performance losses. SE technique except for the study numbered [140] performs a confidential inspection without any changes to the underlying protocols [131]. Inspections using this technique which not require decryption and re-encryption only require the comparison of rulesets to tokenized data. This approach provides a limited control mechanism as inspection is not performed on decrypted packets. However, the studies numbered [133], [139] and [144] using this approach can also perform DPI processing for deeper analysis of traffic when suspicious flows are detected. Alike the schemas in the study numbered [139] require a separate channel to transmit tokenized traffic. In these cases, the client and the server must have the installed schema so that the MB can inspect the inbound and outbound traffic. That is, it requires to change in the structure of the existing encrypted traffic setup. In addition, security and privacy concerns about the use of this technique are resulted from attacks such as leakage-abuse [161], reconstruction [162], inference [163] and passive [164] on SE technique.

ML technique does not perform decryption while analyzing network traffic. This approach represents an ideal solution in terms of the security and application settings as it does not require any changes to the existing setup. The technique is advantageous compared to the AC technique which changes the TLS protocol or MITH, SE and TH techniques which require changes in the client and server settings. The inspection using ML techniques has inherent limitations on data that can be analyzed. The main concern about the ML technique is whether it completely meets security requirements. The inspection on headers and metadata only is insufficient for cases of using that also require analyzing over payloads [127]. Attackers carry out their attacks over encrypted channels by using encrypted traffic to disguise themselves [140, 165]. The ability to have information from both the packet header and the payload provides control over communication between two endpoints. The analyzes performed on the packet header only are insufficient to detect whether the packet contains malware. For this reason, it is necessary to examine the packet payload to detect attacks that can be performed on encrypted channels. This approach helps to neutralize the danger of hidden threats [132]. Therefore, it creates a motivation to access plain text data. Accordingly, MITH, AC, TH and some SE schemas [133, 139, 144] achieve a similar efficiency with analyzes that are performed on unencrypted traffic data.

4 Ensuring security with DPI

Identifying issues that threaten the security of the network is an important and current issue for DPI. IoT and SDN, which are architectures that generate state-of-the-art network traffic, have many difficulties in ensuring security due to their structure. In this section, difficulties in ensuring the security of IoT and SDN structures are emphasized, and the role of IDSs is evaluated in this regard. In addition, applications of the DPI method on IoT and SDN architectures are examined, and the DPI method is evaluated together with the other techniques commonly used in IDSs. DPI can be applied as a single technique in IDSs, or it can be used as a component of a hybrid system by combining with other techniques. In Figure 6, the classification of IDSs as for the using method is showed by giving references to the studies numbered [166], [167] and [168].

In DPI-based IDS systems, IDS alerts when the attack pattern matches the input stream. A signature is defined for each attack, and as regard to the number of these attacks increases the cost of storage increases. Anomaly-based IDS has an initial stage in which data is collected about the usual behavior of the observed system. A threshold value is set when suspicious behavior is encountered in order to alarm IDS. Unlike the signature-based method, this ML-based technique can detect unknown attacks. However, as mentioned earlier, it has many disadvantages such as the lack of labeled data, the computational cost, the difficulties in retraining or poor quality data [1]. This technique has a very high false-positive rate as there may be deviations from the threshold value. It also has a relatively high false-negative rate since attacks may show small deviations that are considered within the norm. It is recommended to overcome the disadvantages of the two methods by creating a hybrid system. The hybrid IDS system created by the combination of the anomaly and the signature-based techniques aims to balance high storage cost and the limited attack detection of the signature-based technique with high computational cost and false positive rate of the anomaly-based techniques [169]. In the hybrid schema, the signature-based technique is used to detect known attacks, whereas the anomaly-based technique is used to detect unknown attacks [170]. Specification-based IDSs can detect intrusions when a deviation from usual behavior occurs alike anomaly-based IDSs. The specifications are developed manually and usual system behavior is detected. These behaviors indicate usual system behavior, and new behavior is verified according to the specified operations [171].

4.1 SDN security with DPI

In traditional networks, all network functions are performed by network devices such as switches, routers or MB. These devices can be supplied by a single vendor or

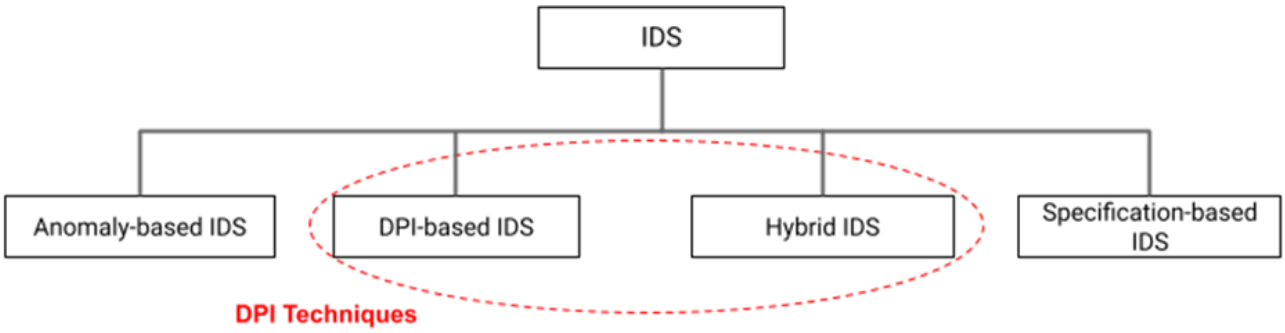

Figure 6. Classification of IDS

multiple vendors. Using network devices from a single vendor reduces innovation and makes it harder to change on the network. The dependency on multiple vendors can increase the complexity of the network [130, 172]. As a solution to these problems, SDN and NFV technologies are developed. SDN is a new networking model in which routing hardware is distinguished from the control plane. It significantly simplifies network management while facilitating innovation and evolution. Network intelligence is centralized in the software-based control plane. The network devices become simple programmable packet routing devices, forming the data plane [173]. NFV implements the network functions via software virtualization techniques and executes them on commodity hardware. These virtual appliances eliminate both the cost of dedicated hardware and energy reasons why running a separate device for each function in various parts of the network. In this way, NFV is an innovative step towards the implementation of the lower-cost agile network infrastructure [174]. However, one of the major barriers in NFV is ensuring that the network performance becomes as good as the purpose-built hardware implementations. Virtualization can cause unusual latency and throughput instability even when network infrastructure is not used at full capacity [175]. The agility provided by a programmable network infrastructure under central control, the fast updating and the easy maintenance facilitate the implementation of virtualized network functions. In this respect, SDN and NFV are complementary technologies [172].

Some security vulnerabilities may arise in network components of SDN architecture and in their relationships due to natural changes [176]. Introducing new interfaces and protocols also leads to the occurrence of new attack interfaces and exploitable targets. The network structure created with SDN may be insecure in case of not taking precautions against the security threats originating from SDN architecture [177]. The determination of preservation architectures for the control channel or another interface in SDN architecture must be made by considering services and protocols that use the channel. The traffic on the communication channels may be intercepted, and this traffic data may be used by the attackers to compromise the assets in the network when the channels and interfaces used for data exchange are not sufficiently secured [178, 179].

Adoption of using cloud infrastructure and virtualization of network functions contributes to the improvement of SDN security by ensuring integration of different security functions. The hardware-based security functions such as IDS, IPS or firewalls begin to be implemented as software-based network functions on commodity hardware by virtualizing in SDN intertwined with NFV technology [180]. Whereas these security services can run anywhere in the network topology, they can also be shared and moved between different security clouds. Accordingly, virtualized security contributes to the scalability of SDN security. The network elements avoid the additional resource cost required to run complicated and resource-consuming security functions with the help of this approach. There is a reduction in the controller processing load when the processing of security functions is transferred to the external data processing systems. This contributes to improve SDN performance [172].

DPI techniques are widely used in IDS applications created to ensure network security in SDN. In addition to the studies that present the systems created by applying DPI engines that perform signature detection on SDN controllers [181-183], there are also some studies supported by an anomaly-based IDS to determine unknown or zero-day attacks such as the studies numbered [13] and [184]. In the study numbered [13], which points to the use of the DPI technique as a security solution for smart grids, a SDN-based security monitoring framework by using a hybrid model of the DL approach and DPI technique is presented. A hybrid system design for the classification of QoS categories is also presented in the study numbered [185] different from the studies which purpose of network security. In this study, a SDN flow classification framework by using DPI and ML-based classifiers is proposed. DPI technique is used to label flows with specific applications to create a partially labeled dataset. Then, the classifier is trained with this dataset and used to identify QoS categories corresponding to different application flows.

In traditional networks, DPI engines implemented in the hardware MBs can be virtualized and dynamically deployed as parts of the software on the commodity hardware by using software-defined networks intertwined with NFV technology. Using such virtual DPI engines is costly from the point of license fee and power consumption [172]. Accordingly, it is important to design low-cost DPI engine

deployment strategies that supply with cybersecurity operational constraints. In order to that, a method based on genetic algorithms and optimizing the cost of DPI engine deployment is proposed in the study numbered [186]. In this direction, an approach reduces the DPI engine number, the global network load, and the number of unanalyzed flows. As a result of analyzes performed with different traffic types, the cost can be reduced up to 58% with this multi-purpose optimization approach. However, this approach is not scalable for the larger networks. Therefore, his problem is handled with the integer linear programming (ILP) method in the study numbered [187]. The validity of designed centrality-based greedy algorithm is evaluated by comparing with the ILP solution, and the complexity in the study numbered [186] is reduced. The experiments using real traffic data show that the proposed approach is 20-25 times faster than ILP, and it is a scalable solution that can be applied for large networks. The proposed cost-optimization-based approaches determine the number and location of DPI engines to be deployed to the network. This method can also be used at runtime to dynamically adapt DPI features.

4.2 IoT security with DPI

IoT composes large number of interconnected devices. These applieances accessed over the internet are interconnected by sensing, communication and computing capabilities. There are many terms that form the basis of IoT i.e machine-to-machine (M2M) communication and sensor networks. These terms are a form of communication that machines interact with each other without any human interaction [188]. The importance of security and privacy for providers and end-users increases the interest in the classification of the network traffic in IoT. The classification of the network traffic also consists of many solutions for the other fields i.e intelligent home systems and health care [10].

The scale of network traffic generated is much larger than in other scenarios because the IoT network has a great number of devices. The variety of services that led to this large scale also caused the traffic flows to grow rapidly. In addition, new obfuscation techniques resulting in more sophisticated and malware traffic allow malware to reach their targets in less time. For these reasons, the traffic in IoT is more complicated than other types of network traffic. This causes great difficulties in the examination of the network traffic [189].

IoT devices in large-scale network traffic are faced with serious security problems. The security violations on interconnected IoT devices can expose confidential information such as audio and video recording, email and passwords. Moreover, poorly designed devices allow various commands to be executed on them and reconfiguration of the firmware [190]. IoT devices have a wide distribution and interact with each other. Therefore, such malicious behavior may endanger the security of the entire network. Mirai is an example of IoT-specific developed malware [191]. This cyber-attack leads to a large-scale DDoS attack by using compromised IoT devices.

The host-based security solutions are inadequate to protect themselves for most smart devices. These shortcomings arise from a lack of physical resources i.e power and computational resources. One of the essential solutions that preserve IoT devices from cyber attacks is network-based security solutions. However, the implementation of these security solutions involves many difficulties due to the structure of IoT networks [10].

It is very difficult to gather and identify signatures from all device actions and interactions owing to the heterogeneous and large-scale structure of IoT networks. Additionally, the firmware releases can affect signatures and behaviors generated by IoT devices. This makes it difficult to identify and filter malicious activity in network traffic for IDS. However, implementing complex IDSs is difficult due to the resource and energy constraints of IoT devices. The low-cost and the low-quality sensor nodes have inflexible constraints i.e computing capacity, memory and energy. Most of the existing IDSs cause a high computational cost and memory requirements for data analysis and storage. Reducing resource consumption by reducing the memory used for storage and computation to more reasonable levels is an important starting point in the researches conducted for IDSs [10, 166]. Developing an IDS that can protect a high detection rate whereas keeping the false alarm rate not high is another challenge. Accordingly, determining the normal behaviors for large-scale sensor applications is an extremely difficult task [192].

The sensor nodes in IoT networks are sensitive to environmental influences. Therefore, the data collected from the sensor nodes for analysis is often insecure. In large-scale IoT networks, a great number of sensor nodes are distributed in harsh and unattended environments that are not easily accessible. The data which is noisy, erroneous, inaccurate, or unnecessary may occur as a result of these nodes running out of battery power or fault. This leads to the insecurity of analyzes performed with low-quality data. Also, a sensor network deployed for long periods in unattended environments is susceptible to dynamic network topology. IoT devices may attach or leave the network from anywhere. A network topology becomes dynamic with the ability to add and remove devices regardless of time or location. It is difficult to adapt to such sudden network topology changes for IDSs. The sensor nodes with different sensing and processing capabilities may move between different locations. In fact, each sensor node may contain a different number and type of sensors. This dynamism and heterogeneity increase the complexity in designing and applying analysis techniques [193].

Similar to SDN, the DPI method may also be applied to IDSs in IoT networks, either alone [194-196] or as a component of a hybrid system. Enabling anomaly detection technique on every resource-constrained IoT device results in high energy consumption, so it is necessary to balance between accuracy detection and energy consumption. The ability of responding to unknown attacks besides increasing the accuracy in detecting known attacks is one of the important goals of hybrid NIDS. For this reason, many hybrid studies use anomaly detection technique to detect the unknown attack signatures whereas the DPI-based technique is used for known attack types [169, 170, 197, 198].

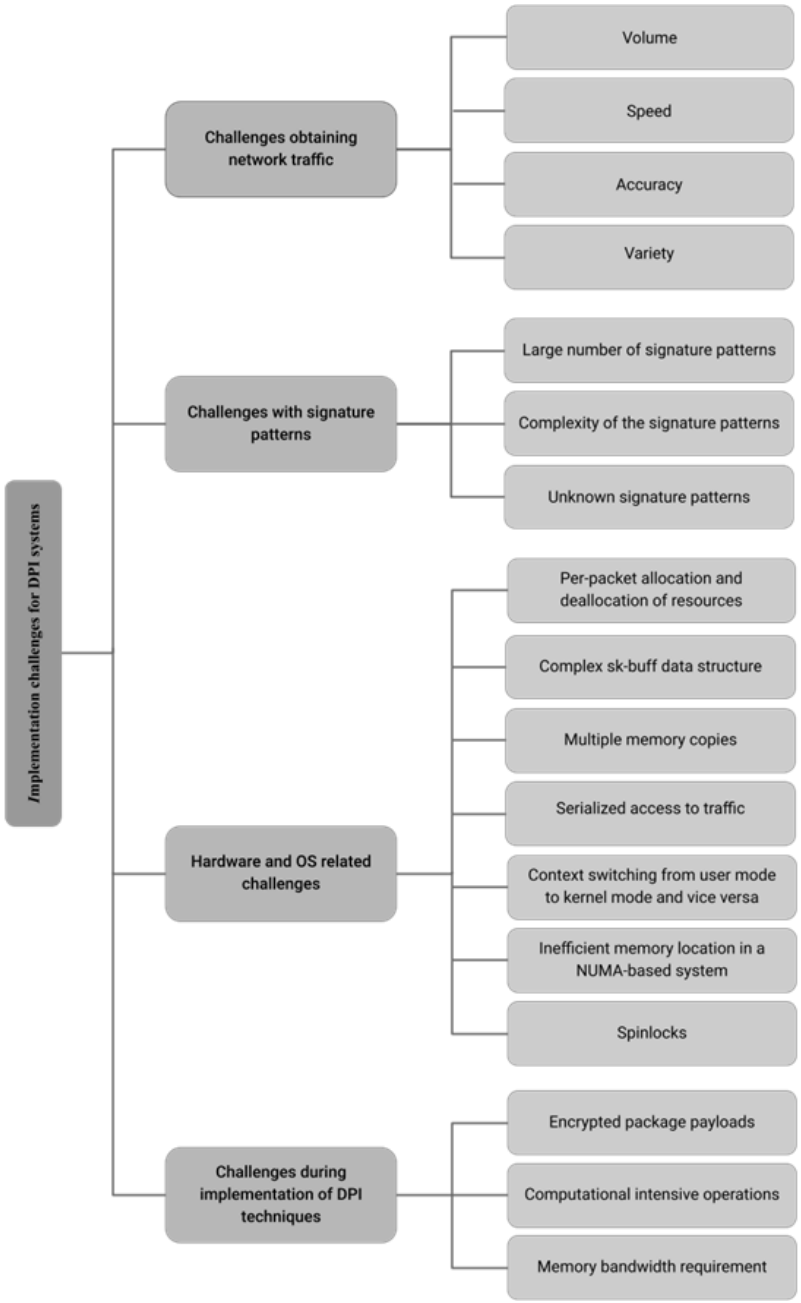

Figure 7. Implementation challenges for DPI systems

In addition, the study numbered [13] which proposes a hybrid model of the ML approach and DPI technique presents a security monitoring framework in which IoT and SDN architectures are combined with each other.

5 Challenges and open issues in implementing DPI systems

Designing efficient DPI mechanisms can be accomplished by individually evaluating and optimizing all processes from acquisition to inspection of network traffic. A review on DPI implementation challenges is presented in this section [1, 10, 11, 16]. Figure 7 summarizes the factors affecting system performance in the entire DPI process. In addition to the application difficulties of the DPI technique, the research issues of the field are discussed in this section.

5.1 Challenges obtaining network traffic

Performing data analysis on modern communication systems and networks includes challenges such as ensuring accuracy and efficient analysis of big data in real-time. Especially in the cellular networks, traffic pattern exhibits complex behavior due to the varied factors i.e device mobility and network heterogeneity. The aforementioned difficulties are related to main difficulties such as the volume, speed, accuracy and diversity of data encountered in the process of obtaining valuable information from the structure called big data [1, 115]. The collection and evaluation of this heterogeneous network traffic resulting from increasing network complexity require efficient mechanisms created by designing scalable and distributed

applications that perform the real-time analysis of large amounts of data. Some platforms developed for processing big data are presented in the studies numbered [199], [200] and [201]. On the other hand, it is unclear whether NTMA applications fully benefit from big data solutions [115].

5.2 Challenges with signature patterns

The increasing number of new systems, services and applications containing malware result in the growth of signature sets that need to be examined by DPI. Although the large signature sets are important for traffic identification, this degrades the overall performance of DPI systems [11]. Also, DPI applications have more complicated signature datasets compared to the others. For instance, the rules of Snort which are the implementation of DPI are more complex than a XML filter using RE for pattern detection [202]. Accordingly, reducing the computational complexity and memory requirement for DPI applications developed by using complex and databases with many signatures rank among the important research topic. On the other hand, signature-based systems cannot be used for the detection of unknown attacks or zero-day attacks. Therefore, it is necessary to create hybrid systems that can respond to unknown attacks while aiming to increase accuracy in detecting known attacks. In addition, these hybrid systems allow the updating of databases used for signature-based detection by detecting new signatures.

5.3 Hardware and OS related challenges

In the process of obtaining network packets, there are many factors that limit performance associated with an operating system such as problems with the use of main memory, spinlocks, context switches, inefficient memory accesses or serialization problem in accessing network traffic [16]. In section 2, the problems encountered in receiving network packets from NIC and processing are detailed, and solutions developed for these problems are presented. In addition, the hardware factors such as processor, memory, hard disk and their communication over the system bus also affect the processing performance. Most of the devices are not physically linked to each other. In this case, main memory becomes the common interface between devices for data communication. The latency in situations that occurs when DMA structure cannot be used, and data must be transmitted from the main memory causes bottlenecks. The time of the NIC is not performing DMA to GPU memory is counted as an example of this situation. If data has to be transmitted from the main memory, a bottleneck may be occurred even though the bus is fast enough [11, 203]. Moreover, the bottlenecks may be occurred in the transfer process of packets due to the limited PCIe bandwidth for the hardware-based solutions. This bottleneck can be reduced by the use of a pre-filtering mechanism that prevents all packets from being forwarded to the hardware for processing.

5.4 Challenges arising from the architecture of the network structure

Contemporary architectures such as IoT or SDN have many difficulties in ensuring security. The resource and energy constraints of the devices in IoT networks make it difficult to implement complex IDSs on these devices, and the host-based security solutions are insufficient to secure these devices. The poor design of devices causes security vulnerabilities leaving them vulnerable to any cyber attack. Considering the span of IoT devices and their interaction with each other, such malicious behavior may have negative effects on the security of the entire network [10, 166, 191]. The security threats originating from SDN architecture may cause insecurity of network structure created by the architecture. The new interfaces and protocols may lead to the occurrence of new attack interfaces and exploitable targets. Accordingly, the network traffic on the communication channels can be accessed by the attackers, and this traffic data can be used to endanger the assets in the network when channels and interfaces used for information exchange are not secure enough [177-179]. Therefore, it is extremely important to develop security strategies on SDN and IoT architectures which security threats affect a wide area. Designing deployment strategies of the DPI engine which is costly in terms of the license fees and power consumption is another challenge arising from the SDN architecture in the development of security strategies [172]. On the other hand, the implementation of DPI engines as virtualized network functionality on SDN architecture may lead information leakage during the implementation of security functions such as IDS, IPS or firewalls [129].

5.5 Challenges during implementation of DPI techniques

Nowadays, almost all network traffic is encrypted due to security and privacy concerns. Therefore, it is important to evaluate the DPI technique over encrypted traffic. In this direction, the difficulties encountered in the application of DPI techniques are examined in this section. The application of DPI techniques in analyzing encrypted network traffic requires cryptographic computations. This requirement causes additional overhead for DPI systems. In addition, the intensive computation required in process of matching complex patterns on plaintext data increases this overhead [11]. Additively, the requirement of reducing memory space required for algorithms that are applied in the pattern matching process is an important research topic about DPI applications that use hardware such as GPUs that accelerate the processing of network packets, especially with parallel computing capabilities. Accordingly, it is a great challenge to develop well-performed systems for DPI applications with high computational complexity and memory space requirement [98]. As summarized in Figure 5, the methods that apply DPI by decoding encrypted traffic and perform an inspection on fully encrypted traffic are discussed together for an accurate comparison. In this direction, the inspection techniques are evaluated in terms of security, performance and functionality.

5.5.1 Security

The successful side-channel attacks on the Intel SGX used in the TH approach and attacks on the SE technique such as inference attack, leakage-abuse attack, reconstruction attacks and passive attacks require further research on the security of these techniques. AC technique provides more flexibility in protecting the confidentiality of

data compared to the MITM technique. However, setting access policies and monitoring network traffic for both MITH and AC techniques are the significant security challenge in a large network with heterogeneous network devices. Moreover, the main concern of techniques that perform inspection by decrypting network traffic on MBs in insecure cloud environments is the risk of information leakage [130, 131]. The main concern of the ML technique which does not decrypt the network traffic is whether it fully meets security requirements [127]. The fact that almost all of today's network traffic is encrypted causes attackers to use encrypted channels to hide themselves [165]. The inspections of headers and metadata only are insufficient to detect whether a packet contains malware.

5.5.2 Performance

The most important factor limiting performance for the SE technique is the additional cryptographic calculations at the setup phase. The research focus of schemas implementing this technique is to increase efficiency by reducing computations. In the TH technique, one of the most important factor limiting the performance is the secure enclave which is limited in terms of memory, storage space and computational capability [131]. Therefore, future researches in this area must focus on balancing storage and performance. Also, all network traffic that routes to the secure enclave causes inspection on all packets. An efficient pre-filtering mechanism enables a more efficient mechanism without unnecessary investigations on the resource-constrained region.

For ML-based models that need to be retrained frequently due to security breaches or network behavior changes, the high complexity in the training phase consumes lots of resources and time, limiting performance [1]. Accordingly, the optimization of ML models from the point of the time complexity and the resource consumption is required.

5.5.3 Functionality

In MITH, AC, TH, and some SE schemas [133, 139, 144], performing inspections on the decrypted data means full control over communication between two endpoints. This provides full functionality for NTMA applications. For ML-based schemas in which inspections are performed on the encrypted traffic, the main challenge is to create ML models that enable the successful implementation of NTMA applications without examining the packet payload. Also, ML technique does not require any changes on the existing setup compared to AC technique that changes TLS protocol, or MITH, SE, and TH techniques that require changes in the client and the server settings. Accordingly, ML technique offers the most advantageous solution in terms of application settings [131].

6 Conclusion

In this paper, a comprehensive review of DPI implementation challenges is made for the scenarios that generate state-of-the-art network traffic. A detailed analysis is presented as regards the determination and the improvement of the parameters that limit the performance in all processes from the collection of the network traffic to the analysis with DPI. The structures of the frameworks containing DPI techniques in the current literature are examined. In addition, this paper points the other techniques that complement the DPI technique instead of focusing only on this technique in determining the procedures to be applied in future mechanisms. The discussion of the application based on DPI technique in the network security field and the analysis of this technique over encrypted traffic fulfil an important deficiency in the literature toward the increasing concerns about security and privacy. For this reason, the security issues of the current architectures such as IoT and SDN, whose main focus is security concerns, are discussed, and DPI implementation difficulties on these architectures are evaluated. In addition, a classification of the proposed methods is presented to perform an inspection on encrypted traffic, and the advantages and disadvantages of the techniques that directly use or do not use the DPI approach or limit the usability of this approach are discussed. In conclusion, the aim of this paper is to evaluate integration of DPI technique into mechanisms aiming to analyze network traffic efficiently by determining the requirements of state-of-the-art network traffic.

Conflict of interest

The author declares that there is no conflict of interest.

Similarity rate (iThenticate): %14

References

[1] M. Abbasi, A. Shahraki, A. Taherkordi, Deep learning for network traffic monitoring and analysis (ntma): A survey, Computer Communications 170 (10), 19–41, 2021. https://doi.org/10.1016/j.comcom.2021.01.021.

[2] G. A. Pimenta Rodrigues, R. de Oliveira Albuquerque, F. E. Gomes de Deus, G. A. De Oliveira J´unior, L. J. Garc´ıa Villalba, T.-H. Kim, et al., Cybersecurity and network forensics: Analysis of malicious traffic towards a honeynet with deep packet inspection, Applied Sciences 7 (10), 1082, 2017. https://doi.org/10.3390/app7101082.

[3] C. Parsons, Deep Packet Inspection in Perspective: Tracing its lineage and surveillance potentials, Citeseer, 2011.

[4] C. Parsons, The politics of deep packet inspection: What drives surveillance by internet service providers?, Ph.D. thesis, 2013.

[5] C. Xu, S. Chen, J. Su, S.-M. Yiu, L. C. Hui, A survey on regular expression matching for deep packet inspection: Applications, algorithms, and hardware platforms, IEEE Communications Surveys & Tutorials 18 (4), 2991–3029, 2016. https://doi.org/10.1109/COMST.2016.2566669.

[6] R. Bendrath, M. Mueller, The end of the net as we know it? deep packet inspection and internet governance, New Media & Society 13 (7), 1142–1160, 2011. https://doi.org/10.1177/1461444811398031.

[7] P. Renals, G. A. Jacoby, Blocking skype through deep packet inspection, in: 2009 42nd Hawaii International Conference on System Sciences, IEEE, pp. 1–5, 2009.

[8] R. M. Topolski, F. Press, P. Knowledge, Nebuad and partner isps: Wiretapping, forgery and browser hijacking, Washington DC: FreePress.
[9] M. R. Shahid, G. Blanc, Z. Zhang, H. Debar, Iot devices recognition through network traffic analysis, in: 2018 IEEE international conference on big data (big data), IEEE, pp. 5187–5192, 2018.
[10] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, N. B. Anuar, The rise of traffic classification in iot networks: A survey, Journal of Network and Computer Applications 154, 102538, 2020. https://doi.org/10.1016/j.jnca.2020.102538.
[11] R. Antonello, S. Fernandes, C. Kamienski, D. Sadok, J. Kelner, I. Godor, G. Szabo, T. Westholm, Deep packet inspection tools and techniques in commodity platforms: Challenges and trends, Journal of Network and Computer Applications 35 (6), 1863–1878, 2012. https://doi.org/10.1016/j.jnca.2012.07.010.
[12] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, K. Hanssgen, A survey of payload-based traffic classification approaches, IEEE Communications Surveys & Tutorials 16 (2), 1135–1156, 2013. https://doi.org/10.1109/SURV.2013.100613.00161.
[13] G. D. L. T. Parra, P. Rad, K.-K. R. Choo, Implementation of deep packet inspection in smart grids and industrial internet of things: Challenges and opportunities, Journal of Network and Computer Applications 135, 32–46, 2019. https://doi.org/10.1016/j.jnca.2019.02.022.
[14] W. Wu, M. Crawford, M. Bowden, The performance analysis of linux networking–packet receiving, Computer Communications 30 (5), 1044–1057, 2007. https://doi.org/10.1016/j.comcom.2006.11.001.
[15] R. Rosen, Linux kernel networking: Implementation and theory, Apress, 2014.
[16] J. L. Garc´ıa-Dorado, F. Mata, J. Ramos, P. M. S. del R´ıo, V. Moreno, J. Aracil, High-performance network traffic processing systems using commodity hardware, in: Data traffic monitoring and analysis, Springer, pp. 3–27, 2013. http://dx.doi.org/10.1007/978-3-642-36784-7_1.
[17] D. Scholz, A look at intels dataplane development kit, Network 115. http://dx.doi.org/10.2313/NET-2014-08-1_15.
[18] G. Liao, X. Znu, L. Bnuyan, A new server i/o architecture for high speed networks, in: 2011 IEEE 17th International Symposium on High Performance Computer Architecture, IEEE, pp. 255–265, 2011.
[19] S. Han, K. Jang, K. Park, S. Moon, Packetshader: a gpu-accelerated software router, ACM SIGCOMM Computer Communication Review 40 (4), 195–206, 2010. https://doi.org/10.1145/1851275.1851207.
[20] W. Wu, P. DeMar, M. Crawford, Why can some advanced ethernet nics cause packet reordering?, IEEE Communications Letters 15 (2), 253–255, 2010. https://doi.org/10.1109/LCOMM.2011.122010.10 022.
[21] C. Benvenuti, Understanding linux network internals, o'relly media, Inc., Sebastopol, CA.
[22] M. Dobrescu, K. Argyraki, S. Ratnasamy, Toward predictable performance in software packet-processing platforms, in: 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12), pp. 141–154., 2012.
[23] D. Barach, L. Linguaglossa, D. Marion, P. Pfister, S. Pontarelli, D. Rossi, High-speed software data plane via vectorized packet processing, IEEE Communications Magazine 56 (12), 97–103, 2018. https://doi.org/10.1109/MCOM.2018.1800069.
[24] E. Kohler, R. Morris, B. Chen, J. Jannotti, M. F. Kaashoek, The click modular router, ACM Transactions on Computer Systems (TOCS) 18 (3), 263–297, 2000. https://doi.org/10.1145/354871.354874.
[25] L. Rizzo, netmap: a novel framework for fast packet i/o, in: 21st USENIX Security Symposium (USENIX Security 12), pp. 101–112, 2012. https://doi.org/10.1145/354871.354874.
[26] INTEL DPDK, https://www.dpdk.org/, Accessed 3 October 2022.
[27] T. Barbette, C. Soldani, L. Mathy, Fast userspace packet processing, in: 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), IEEE, pp. 5–16, 2015.
[28] PFRING, http://www.ntop.org/products/packet-capture/pf_ring/, Accessed 3 October 2022.
[29] W. Sun, R. Ricci, Fast and flexible: Parallel packet processing with gpus and click, in: Architectures for Networking and Communications Systems, IEEE, pp. 25–35, 2013.
[30] G. Vasiliadis, L. Koromilas, M. Polychronakis, S. Ioannidis, {GASPP}: A gpu-accelerated stateful packet processing framework, in: 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14), pp. 321–332, 2014.
[31] Y. Go, M. A. Jamshed, Y. Moon, C. Hwang, K. Park, Apunet: Revitalizing {GPU} as packet processing accelerator, in: 14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17), pp. 83–96, 2017.
[32] B. Li, K. Tan, L. Luo, Y. Peng, R. Luo, N. Xu, Y. Xiong, P. Cheng, E. Chen, Clicknp: Highly flexible and high performance network processing with reconfigurable hardware, in: Proceedings of the 2016 ACM SIGCOMM Conference, pp. 1–14, 2016.
[33] Intel DPDK Performance on the SAU5081I Server, https://www.accton.com/Technology-Brief/inteldpdk-performance-on-the-sau5081i-server/, Accessed 3 October 2022.
[34] L. Rizzo, L. Deri, A. Cardigliano, 10 gbit/s line rate packet processing using commodity hardware: Survey and new proposals, 2012.
[35] Google Transparency Report, https://transparencyreport.google.com/https/overview, Accessed 3 October 2022.
[36] F. Yu, R. H. Katz, T. V. Lakshman, Gigabit rate packet pattern-matching using tcam, in: Proceedings of the 12th IEEE International Conference on Network

Protocols, 2004. ICNP 2004., IEEE, pp. 174–183, 2004.
[37] J.-S. Sung, S.-M. Kang, Y. Lee, T.-G. Kwon, B.-T. Kim, A multi-gigabit rate deep packet inspection algorithm using tcam, in: GLOBECOM'05. IEEE Global Telecommunications Conference, Vol. 1, IEEE, 2005.
[38] T. Ho, S.-J. Cho, S.-R. Oh, Parallel multiple pattern matching schemes based on cuckoo filter for deep packet inspection on graphics processing units, IET Information Security 12 (4), 381–388, 2018. https://doi.org/10.1049/iet-ifs.2017.0421.
[39] J. Han, S. Kim, D. Cho, B. Choi, J. Ha, D. Han, A secure middlebox framework for enabling visibility over multiple encryption protocols, IEEE/ACM Transactions on Networking 28 (6), 2727–2740, 2020. https://doi.org/10.1109/TNET.2020.3016785.
[40] H. Duan, X. Yuan, C. Wang, Lightbox: Sgx-assisted secure network functions at near-native speed. corr abs/1706.06261, arXiv preprint arXiv:1706.06261, 2017.
[41] B. Fan, D. G. Andersen, M. Kaminsky, M. D. Mitzenmacher, Cuckoo filter: Practically better than bloom, in: Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, pp. 75–88, 2014.
[42] L. Deri, M. Martinelli, T. Bujlow, A. Cardigliano, ndpi: Open-source high-speed deep packet inspection, in: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE, pp. 617–622, 2014.
[43] F. Risso, M. Baldi, O. Morandi, A. Baldini, P. Monclus, Lightweight, payload-based traffic classification: An experimental evaluation, in: 2008 IEEE International Conference on Communications, IEEE, pp. 5869–5875, 2008.
[44] Protocol and application classification with metadata extraction (PACE) , https://www.ipoque.com/products/dpi-engine-rsrpace-2, Accessed 3 October 2022.
[45] T. Bujlow, V. Carela-Espanol, Comparison of deep packet inspection (dpi) tools for traffic classification.
[46] S. Alcock, R. Nelson, Measuring the accuracy of open-source payload-based traffic classifiers using popular internet applications, in: 38th Annual IEEE Conference on Local Computer Networks-Workshops, IEEE, pp. 956–963, 2013.
[47] T. Bujlow, V. Carela-Espa˜nol, P. Barlet-Ros, Independent comparison of popular dpi tools for traffic classification, Computer Networks 76, 75–89, 2015. https://doi.org/10.1016/j.comnet.2014.11.001.
[48] G. B. Satrya, F. E. Nugroho, T. Brotoharsono, Improving network security-a comparison between ndpi and l7-filter, International Journal on Information and Communication Technology (IJoICT) 2 (2), 11–11, 2016. https://doi.org/10.21108/IJOICT.2016.22.77.
[49] R. Muth, U. Manber, Approximate multiple string search, in: Annual Symposium on Combinatorial Pattern Matching, Springer, pp. 75–86, 1996.
[50] R. M. Karp, M. O. Rabin, Efficient randomized pattern-matching algorithms, IBM journal of research and development 31 (2), 249–260, 1987. https://doi.org/10.1147/rd.312.0249.
[51] V. Gupta, M. Singh, V. K. Bhalla, Pattern matching algorithms for intrusion detection and prevention system: A comparative analysis, in: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, pp. 50–54, 2014.
[52] N. Shoaib, J. Shamsi, T. Mustafa, A. Zaman, J. ul Hasan, M. Gohar, Gdpi: Signature based deep packet inspection using gpus, Int. J. Adv. Comput. Sci. Appl 8 (11), 210–216, 2017. https://doi.org/10.14569/IJACSA.2017.081128.
[53] M. Ramesh, H. Jeon, Parallelizing deep packet inspection on gpu, in: 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), IEEE, pp. 248–253, 2018.
[54] J. Sharma, M. Singh, Cuda based rabin-karp pattern matching for deep packet inspection on a multicore gpu, International Journal of Computer Network and Information Security 7 (10), 70–77, 2015. https://doi.org/10.5815/ijcnis.2015.10.08.
[55] B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (7), 422–426, 1970. https://doi.org/10.1145/362686.362692.
[56] L. Fan, P. Cao, J. Almeida, A. Z. Broder, Summary cache: a scalable wide-area web cache sharing protocol, IEEE/ACM transactions on networking 8 (3), 281–293, 2000. https://doi.org/10.1109/90.851975.
[57] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, G. Varghese, An improved construction for counting bloom filters, in: European Symposium on Algorithms, Springer, pp. 684–695, 2006.
[58] F. Putze, P. Sanders, J. Singler, Cache-, hash-and space-efficient bloom filters, in: International Workshop on Experimental and Efficient Algorithms, Springer, pp. 108–121, 2007.
[59] D. E. Knuth, The art of computer programming: Sorting and searching, Vol. 3, Addison-Wesley Publishing Company.
[60] M. Al-Hisnawi, M. Ahmadi, Qcf for deep packet inspection, IET Networks 7 (5), 346–352, 2018. https://doi.org/10.1049/iet-net.2017.0037.
[61] N. S. Artan, H. J. Chao, Multi-packet signature detection using prefix bloom filters, in: GLOBECOM'05. IEEE Global Telecommunications Conference, 2005., Vol. 3, IEEE, 2005.
[62] T. Kocak, I. Kaya, Low-power bloom filter architecture for deep packet inspection, IEEE Communications Letters 10 (3), 210–212, 2006. https://doi.org/10.1109/LCOMM.2006.1603387.
[63] Y. Chen, A. Kumar, J. J. Xu, A new design of bloom filter for packet inspection speedup, in: IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference, IEEE, pp. 1–5, 2007.

[64] M. Al-Hisnawi, M. Ahmadi, Deep packet inspection using quotient filter, IEEE Communications Letters 20 (11), 2217–2220, 2016. https://doi.org/10.1109/LCOMM.2016.2601898.

[65] M. Al-Hisnawi, M. Ahmadi, Deep packet inspection using cuckoo filter, in: 2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT), IEEE, pp. 197–202, 2017.

[66] R. S. Boyer, J. S. Moore, A fast string searching algorithm, Communications of the ACM 20 (10), 762–772, 1977. https://doi.org/10.1145/359842.359859.

[67] S. Wu, U. Manber, et al., A fast algorithm for multi-pattern searching, University of Arizona. Department of Computer Science, 1994.

[68] Y. Wang, H. Kobayashi, An improved technology for content matching intrusion detection system, in: 2006 International Conference on Software in Telecommunications and Computer Networks, IEEE, pp. 238–241, 2006.

[69] A. A. Hasan, N. A. A. Rashid, Hash-boyer-moore-horspool string matching algorithm for intrusion detection system, in: International Conference on Computer Networks and Communication Systems, IPCSIT, 35, pp. 12–16, 2012.

[70] S. Sharma, M. Dixit, Single digit hash boyer moore horspool pattern matching algorithm for intrusion detection system, International Journal of Future Generation Communication and Networking 9 (9), 169–180, 2016. https://doi.org/10.14257/ijfgcn.2016.9.9.15.

[71] R. Padmashani, S. Sathyadevan, D. Dath, Bsnort ips better snort intrusion detection/prevention system, in: 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), IEEE, pp. 46–51, 2012.

[72] S. Gupta, Efficient malicious domain detection using word segmentation and bm pattern matching, in: 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), IEEE, pp. 1–6, 2016.

[73] T. F. A. Rahman, A. G. Buja, K. Abd, F. M. Ali, Sql injection attack scanner using boyer-moore string matching algorithm., J. Comput. 12 (2), 183–189, 2017. https://doi.org/10.17706/jcp.12.2.183-189.

[74] Y. Otoum, A. Nayak, As-ids: Anomaly and signature based ids for the internet of things, Journal of Network and Systems Management 29 (3), 1–26, 2021. https://doi.org/0.1007/s10922-021-09589-6.

[75] Q. Zheng, An improved multiple patterns matching algorithm for intrusion detection, in: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, Vol. 2, IEEE, pp. 124–127, 2010.

[76] C. Ke-Qin, D. Lin, W. Hui, An improved multi-pattern matching algorithms in intrusion detection, in: 2013 Fifth International Conference on Measuring Technology and Mechatronics Automation, IEEE, pp. 203–205, 2013.

[77] M. Aldwairi, K. Al-Khamaiseh, F. Alharbi, B. Shah, Bloom filters optimized wu-manber for intrusion detection, Journal of Digital Forensics, Security and Law 11 (4), 5, 2016. https://doi.org/10.15394/jdfsl.2016.1427.

[78] B. Zhang, X. Chen, X. Pan, Z. Wu, High concurrence wu-manber multiple patterns matching algorithm, in: Proceedings. The 2009 International Symposium on Information Processing (ISIP 2009), Citeseer, p. 404, 2009.

[79] D. Luchaup, L. De Carli, S. Jha, E. Bach, Deep packet inspection with dfa-trees and parametrized language overapproximation, in: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, IEEE, pp. 531–539, 2014.

[80] M. ˇCeˇska, V. Havlena, L. Hol´ık, O. Leng´al, T. Vojnar, Approximate reduction of finite automata for high-speed network intrusion detection, International Journal on Software Tools for Technology Transfer 22 (5), 523–539, 2020. https://doi.org/10.1007/978-3-319-89963-3_9.

[81] M. Ceˇska, V. Havlena, L. Hol´ık, J. Korenek, O. Leng´al, D. Matouˇsek, J. Matouˇsek, J. Semric, T. Vojnar, Deep packet inspection in fpgas via approximate nondeterministic automata, in: 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), IEEE, pp. 109–117, 2019.

[82] M. Roesch, et al., Snort: Lightweight intrusion detection for networks., in: Lisa, Vol. 99, 1999, pp. 229–238.

[83] R. Sommer, Bro: An open source network intrusion detection system, Security, E-learning, E-Services, 17. DFNArbeitstagung¨uber Kommunikationsnetze. https://doi.org/10.1007/978-3-319-89963-3_9.

[84] Cisco IOS Intrusion Prevention System (IPS) , https://www.cisco.com/c/en/us/products/security/iosintrusion-prevention-system-ips/index.html, Accessed 3 October 2022.

[85] X. Yu, W.-c. Feng, D. Yao, M. Becchi, O3 fa: A scalable finite automata–based pattern-matching engine for out–of–order deep packet inspection, in: 2016 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), IEEE, pp. 1–11, 2016.

[86] C. Yin, H. Wang, X. Yin, R. Sun, J. Wang, Improved deep packet inspection in data stream detection, The Journal of Supercomputing 75 (8), 4295–4308, 2019. https://doi.org/10.1007/s11227-018-2685-y.

[87] R. Sun, L. Shi, C. Yin, J. Wang, An improved method in deep packet inspection based on regular expression, The Journal of Supercomputing 75 (6), 3317–3333, 2019. https://doi.org/10.1007/s11227-018-2517-0.

[88] S. Nagaraju, B. Shanmugham, K. Baskaran, High throughput token driven fsm based regex pattern matching for network intrusion detection system, Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2021.04.028.

[89] A. V. Aho, M. J. Corasick, Efficient string matching: an aid to bibliographic search, Communications of the ACM 18 (6), 333–340, 1975. https://doi.org/10.1145/360825.360855.
[90] M. Norton, Optimizing pattern matching for intrusion detection, Sourcefire, Inc., Columbia, MD.
[91] N. Tuck, T. Sherwood, B. Calder, G. Varghese, Deterministic memory-efficient string matching algorithms for intrusion detection, in: IEEE INFOCOM 2004, 4, IEEE, pp. 2628–2639, 2004.
[92] L. Tan, T. Sherwood, A high throughput string matching architecture for intrusion detection and prevention, in: 32nd International Symposium on Computer Architecture (ISCA'05), IEEE, pp. 112–122, 2005.
[93] T.-H. Lee, N.-L. Huang, A pattern-matching scheme with high throughput performance and low memory requirement, IEEE/ACM Transactions on Networking 21 (4), 1104–1116, 2012. https://doi.org/10.1109/TNET.2012.2224881.
[94] H. Kim, A scalable architecture for reducing power consumption in pipelined deep packet inspection system, Microelectronics Journal 46 (10), 950–955, 2015. https://doi.org/10.1016/j.mejo.2015.08.002.
[95] X. Zha, S. Sahni, Multipattern string matching on a gpu, in: 2011 IEEE Symposium on Computers and Communications (ISCC), IEEE, pp. 277–282, 2011.
[96] C.-H. Lin, C.-H. Liu, L.-S. Chien, S.-C. Chang, Accelerating pattern matching using a novel parallel algorithm on gpus, IEEE Transactions on Computers 62 (10), 1906–1916, 2012. https://doi.org/10.1109/TC.2012.254.
[97] C.-L. Lee, Y.-S. Lin, Y.-C. Chen, A hybrid cpu/gpu pattern-matching algorithm for deep packet inspection, PloS one 10 (10), e0139301, 2015. https://doi.org/10.1371/journal.pone.0139301.
[98] C.-L. Hsieh, L. Vespa, N. Weng, A high-throughput dpi engine on gpu via algorithm/implementation co-optimization, Journal of Parallel and Distributed Computing 88, 46–56, 2016. https://doi.org/10.1016/j.jpdc.2015.11.001.
[99] B. Choi, J. Chae, M. Jamshed, K. Park, D. Han, {DFC}: Accelerating string pattern matching for network applications, in: 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16), pp. 551–565, 2016.
[100] D. C. Sicker, P. Ohm, D. Grunwald, Legal issues surrounding monitoring during network research, in: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pp. 141–148, 2007.
[101] T. T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE communications surveys & tutorials 10 (4), 56–76, 2008. https://doi.org/10.1109/SURV.2008.080406.
[102] A. Finamore, M. Mellia, M. Meo, D. Rossi, Kiss: Stochastic packet inspection classifier for udp traffic, IEEE/ACM Transactions on Networking 18 (5), 1505–1515, 2010. https://doi.org/10.1109/TNET.2010.2044046.
[103] B. Anderson, D. McGrew, Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity, in: Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining, pp. 1723–1732, 2017.
[104] B. Anderson, S. Paul, D. McGrew, Deciphering malware's use of tls (without decryption), Journal of Computer Virology and Hacking Techniques 14 (3), 195–211, 2018. https://doi.org/10.1007/s11416-017-0306-6.
[105] A. Yamada, Y. Miyake, K. Takemori, A. Studer, A. Perrig, Intrusion detection for encrypted web accesses, in: 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), 1, IEEE, pp. 569–576, 2007.
[106] J. Y. Chung, B. Park, Y. J. Won, J. Strassner, J. W. Hong, Traffic classification based on flow similarity, in: International Workshop on IP Operations and Management, Springer, pp. 65–77, 2009.
[107] E. Rocha, P. Salvador, A. Nogueira, Detection of illicit network activities based on multivariate gaussian fitting of multi-scale traffic characteristics, in: 2011 IEEE International Conference on Communications (ICC), IEEE, pp. 1–6, 2011.
[108] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.
[109] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553), 436–444, 2015. https://doi.org/10.1038/nature14539.
[110] M. A. Alsheikh, D. Niyato, S. Lin, H.-P. Tan, Z. Han, Mobile big data analytics using deep learning and apache spark, IEEE network 30 (3), 22–29, 2016. https://doi.org/10.1109/MNET.2016.7474340.
[111] B. J. Radford, L. M. Apolonio, A. J. Trias, J. A. Simpson, Network traffic anomaly detection using recurrent neural networks, arXiv preprint arXiv:1803.10769.
[112] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, M. Tornatore, Network traffic prediction based on diffusion convolutional recurrent neural networks, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, pp. 246–251, 2019.
[113] K. Ding, S. Ding, A. Morozov, T. Fabarisov, K. Janschek, On-line error detection and mitigation for time-series data of cyber-physical systems using deep learning based methods, in: 2019 15th European Dependable Computing Conference (EDCC), IEEE, pp. 7–14, 2019.
[114] W. Zhong, F. Gu, A multi-level deep learning system for malware detection, Expert Systems with Applications 133, 151–162, 2019. https://doi.org/10.1016/j.eswa.2019.04.064.
[115] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, P. Casas, A survey on big data for network traffic monitoring and analysis, IEEE Transactions on

Network and Service Management 16 (3), 800–813, 2019. https://doi.org/10. 1109/TNSM.2019.2933358.

[116] M. Alicherry, M. Muthuprasanna, V. Kumar, High speed pattern matching for network ids/ips, in: Proceedings of the 2006 IEEE International Conference on Network Protocols, IEEE, pp. 187–196, 2006.

[117] H. Kim, K.-I. Choi, A pipelined non-deterministic finite automaton-based string matching scheme using merged state transitions in an fpga, PloS one 11 (10), e0163535, 2016. https://doi.org/10.1371/journal.pone.0163535.

[118] I. Sourdis, D. N. Pnevmatikatos, S. Vassiliadis, Scalable multigigabit pattern matching for packet inspection, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16 (2), 156–166, 2008. https://doi.org/10.1109/ TVLSI.2007.912036.

[119] R.-T. Liu, N.-F. Huang, C.-H. Chen, C.-N. Kao, A fast string-matching algorithm for network processor-based intrusion detection system, ACM Transactions on Embedded Computing Systems (TECS) 3 (3), 614–633, 2004. https://doi.org/10.1145/1015047.1015055.

[120] D. F. Bacon, R. Rabbah, S. Shukla, Fpga programming for the masses, Communications of the ACM 56 (4), 56–63, 2013. https://doi.org/10.1145/2436256.2436271.

[121] Y. Sun, H. Liu, V. C. Valgenti, M. S. Kim, Hybrid regular expression matching for deep packet inspection on multi-core architecture, in: 2010 Proceedings of 19th International Conference on Computer Communications and Networks, IEEE, pp. 1–7, 2010.

[122] Y.-H. E. Yang, V. K. Prasanna, Robust and scalable string pattern matching for deep packet inspection on multicore processors, IEEE Transactions on Parallel and Distributed Systems 24 (11), 2283–2292, 2012 https://doi.org/10.1109/TPDS.2012.217.

[123] C.-L. Lee, T.-H. Yang, A flexible pattern-matching algorithm for network intrusion detection systems using multi-core processors, Algorithms 10 (2), 58, 2017. https://doi.org/10.3390/a10020058.

[124] CUDA C PROGRAMMING GUIDE , https://docs.nvidia.com/cuda/archive/9.1/pdf/CUDA_C_Programming_Guide.pdf, Accessed 3 October 2022.

[125] R. Smith, N. Goyal, J. Ormont, K. Sankaralingam, C. Estan, Evaluating gpus for network packet signature matching, in: 2009 IEEE International Symposium on Performance Analysis of Systems and Software, IEEE, pp. 175–184, 2009.

[126] M. Ramesh, Network traffic anomaly-detection framework using gpus, Ph.D. thesis, San Jose State University, 2017.

[127] X. d. C. de Carnavalet, P. C. van Oorschot, A survey and analysis of tls interception mechanisms and motivations, arXivpreprint arXiv:2010.16388.

[128] K. Moriarty, A. Morton, Effects of pervasive encryption on operators, draft-mm-wg-effect-encrypt-25 (work in progress).

[129] K. Bhargavan, I. Boureanu, A. Delignat-Lavaud, P.-A. Fouque, C. Onete, A formal treatment of accountable proxying over tls, in: 2018 IEEE Symposium on Security and Privacy (SP), IEEE, pp. 799–816, 2018.

[130] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, Z. Liu, Embark: Securely outsourcing middleboxes to the cloud, in: 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16), 2016, pp. 255–273.

[131] G. S. Poh, D. M. Divakaran, H. W. Lim, J. Ning, A. Desai, A survey of privacy-preserving techniques for encrypted traffic inspection over network middleboxes, arXiv preprint arXiv:2101.04338.

[132] L. S. Huang, A. Rice, E. Ellingsen, C. Jackson, Analyzing forged ssl certificates in the wild, in: 2014 IEEE Symposium on Security and Privacy, IEEE, pp. 83–97, 2014.

[133] J. Ning, G. S. Poh, J.-C. Loh, J. Chia, E.-C. Chang, Privdpi: Privacy-preserving encrypted traffic inspection with reusable obfuscated rules, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1657–1670, 2019.

[134] X. de Carn´e de Carnavalet, M. Mannan, Killed by proxy: analyzing client-end tls interception software https://doi.org/10.3390/a10020058.

[135] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. A. Halderman, V. Paxson, The security impact of https interception., in: NDSS, 2017.

[136] L. Waked, M. Mannan, A. Youssef, To intercept or not to intercept: Analyzing tls interception in network appliances, in: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp. 399–412, 2018.

[137] MitMProxy, https://mitmproxy.org/, Accessed 3 October 2022.

[138] SSLSpit, https://www.roe.ch/, Accessed 3 October 2022.

[139] J. Sherry, C. Lan, R. A. Popa, S. Ratnasamy, Blindbox: Deep packet inspection over encrypted traffic, in: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, pp. 213–226, 2015.

[140] S. Canard, A. Diop, N. Kheir, M. Paindavoine, M. Sabt, Blindids: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 561–574, 2017.

[141] T. Fuhr, P. Paillier, Decryptable searchable encryption, in: International Conference on Provable Security, Springer, pp. 228–236, 2007.

[142] J. Fan, C. Guan, K. Ren, Y. Cui, C. Qiao, Spabox: Safeguarding privacy during deep packet inspection at a middlebox, IEEE/ACM Transactions on Networking 25 (6), 3753–3766, 2017. https://doi.org/10.1109/TNET.2017.2753044.

[143] J. Ning, X. Huang, G. S. Poh, S. Xu, J.-C. Loh, J. Weng, R. H. Deng, Pine: Enabling privacy-preserving deep packet inspection on tls with rule-hiding and fast

connection establishment, in: European Symposium on Research in Computer Security, Springer, pp. 3–22, 2020.

[144] H. Ren, H. Li, D. Liu, G. Xu, N. Cheng, X. S. Shen, Privacy-preserving efficient verifiable deep packet inspection for cloud-assisted middlebox, IEEE Transactions on Cloud Computing. https://doi.org/10.1109/TCC.2020.2991167.

[145] H. J. Asghar, L. Melis, C. Soldani, E. De Cristofaro, M. A. Kaafar, L. Mathy, Splitbox: Toward efficient private network function virtualization, in: Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization, pp. 7–13, 2016.

[146] S. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S.-F. Sun, D. Liu, C. Zuo, Result pattern hiding searchable encryption for conjunctive queries, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 745–762, 2018.

[147] D. Naylor, K. Schomp, M. Varvello, I. Leontiadis, J. Blackburn, D. R. L´opez, K. Papagiannaki, P. Rodriguez Rodriguez, P. Steenkiste, Multi-context tls (mctls) enabling secure in-network functionality in tls, ACM SIGCOMM Computer Communication Review 45 (4), 199–212, 2015. https://doi.org/10.1145/2829988.2787482.

[148] D. Naylor, R. Li, C. Gkantsidis, T. Karagiannis, P. Steenkiste, And then there were more: Secure communication for more than two parties, in: Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, pp. 88–100, 2017.

[149] H. Lee, Z. Smith, J. Lim, G. Choi, S. Chun, T. Chung, T. T. Kwon, matls: How to make tls middlebox-aware?, in: NDSS, 2019.

[150] D. Goltzsche, S. R¨usch, M. Nieke, S. Vaucher, N. Weichbrodt, V. Schiavoni, P.-L. Aublin, P. Cosa, C. Fetzer, P. Felber, et al., Endbox: Scalable middlebox functions using client-side trusted execution, in: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, pp. 386–397, 2018.

[151] Perl Compatible Regular Expressions Library (PCRE2), https://ftp.pcre.org/pub/pcre/, Accessed 3 October 2022.

[152] N. D. Matsakis, F. S. Klock, The rust language, ACM SIGAda Ada Letters 34 (3), 103–104, 2014. https://doi. org/10.1145/2692956.2663188.

[153] D. Kuvaiskii, O. Oleksenko, S. Arnautov, B. Trach, P. Bhatotia, P. Felber, C. Fetzer, Sgxbounds: Memory safety for shielded execution, in: Proceedings of the Twelfth European Conference on Computer Systems, pp. 205–221, 2017.

[154] L. Szekeres, M. Payer, T. Wei, D. Song, Sok: Eternal war in memory, in: 2013 IEEE Symposium on Security and Privacy, IEEE, pp. 48–62, 2013.

[155] R. Poddar, C. Lan, R. A. Popa, S. Ratnasamy, Safebricks: Shielding network functions in the cloud, in: 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18), pp. 201–216, 2018.

[156] B. Trach, A. Krohmer, F. Gregor, S. Arnautov, P. Bhatotia, C. Fetzer, Shieldbox: Secure middleboxes using shielded execution, in: Proceedings of the Symposium on SDN Research, pp. 1–14, 2018.

[157] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'keeffe, M. L. Stillwell, et al., {SCONE}: Secure linux containers with intel {SGX}, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pp. 689–703, 2016.

[158] hyperscan, https://www.hyperscan.io/, Accessed 3 October 2022.

[159] J. M. Sherry, Middleboxes as a cloud service, Ph.D. thesis, UC Berkeley, 2016.

[160] Y. Lindell, The security of intel sgx for key protection and data privacy applications.

[161] D. Cash, P. Grubbs, J. Perry, T. Ristenpart, Leakage-abuse attacks against searchable encryption, in: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp. 668–679, 2015.

[162] G. Kellaris, G. Kollios, K. Nissim, A. O'neill, Generic attacks on secure outsourced databases, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1329–1340, 2016.

[163] M. S. Islam, M. Kuzu, M. Kantarcioglu, Access pattern disclosure on searchable encryption: ramification, attack and mitigation., in: Ndss, Vol. 20, Citeseer, p. 12, 2012.

[164] J. Ning, J. Xu, K. Liang, F. Zhang, E.-C. Chang, Passive attacks against searchable encryption, IEEE Transactions on Information Forensics and Security 14 (3), 789–802, 2018. https://doi.org/10.1109/TIFS.2018.2866321.

[165] Cisco Encrypted Traffic Analytics White Paper, https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/enterprise-network-security/nb-09-encrytd-traf-anlytcs-wp-cte-en.html, Accessed 3 October 2022.

[166] S. Hajiheidari, K. Wakil, M. Badri, N. J. Navimipour, Intrusion detection systems in the internet of things: A comprehensive investigation, Computer Networks 160, 165–191, 2019. https://doi.org/10.1016/j.comnet.2019.05.014.

[167] C. Birkinshaw, E. Rouka, V. G. Vassilakis, Implementing an intrusion detection and prevention system using softwaredefined networking: Defending against port-scanning and denial-of-service attacks, Journal of Network and Computer Applications 136, 71–85, 2019. https://doi.org/10.1016/j.jnca.2019.03.005.

[168] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, K.-Y. Tung, Intrusion detection system: A comprehensive review, Journal of Network and Computer Applications 36 (1), 16–24, 2013. https://doi.org/10.1016/j.jnca.2012.09.004.

[169] S. Raza, L. Wallgren, T. Voigt, Svelte: Real-time intrusion detection in the internet of things, Ad hoc networks 11 (8), 2661–2674, 2013. https://doi.org/10.1016/j.adhoc.2013.04.014.

[170] H. Sedjelmaci, S. M. Senouci, M. Al-Bahri, A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology, in: 2016 IEEE international conference on communications (ICC), IEEE, pp. 1–6, 2016.

[171] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, S. Zhou, Specification-based anomaly detection: a new approach for detecting network intrusions, in: Proceedings of the 9th ACM conference on Computer and communications security, pp. 265–274, 2002.

[172] S. Demirci, M. Demirci, S. Sagiroglu, Virtual security functions and their placement in software defined networks: A survey, Gazi University Journal of Science 32 (3), 833–851, 2019. https://doi.org/10.35378/gujs.422000.

[173] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, IEEE Communications surveys & tutorials 16 (3), 1617–1634, 2014. https://doi.org/10.1109/SURV.2014.012214.00180.

[174] B. Han, V. Gopalakrishnan, L. Ji, S. Lee, Network function virtualization: Challenges and opportunities for innovations, IEEE Communications Magazine 53 (2), 90–97, 2015. https://doi.org/10.1109/MCOM.2015.7045396.

[175] G.Wang, T. E. Ng, The impact of virtualization on network performance of amazon ec2 data center, in: 2010 Proceedings IEEE INFOCOM, IEEE, pp. 1–9, 2010.

[176] S. Scott-Hayward, S. Natarajan, S. Sezer, A survey of security in software defined networks, IEEE Communications Surveys & Tutorials 18 (1), 623–654, 2015. https://doi.org/10.1109/COMST.2015.2453114.

[177] J. C. C. Chica, J. C. Imbachi, J. F. B. Vega, Security in sdn: A comprehensive survey, Journal of Network and Computer Applications 159, 102595, 2020. https://doi.org/10.1016/j.jnca.2020.102595.

[178] L. Schehlmann, S. Abt, H. Baier, Blessing or curse? revisiting security aspects of software-defined networking, in: 10th International Conference on Network and Service Management (CNSM) and Workshop, IEEE, pp. 382–387, 2014.

[179] M. Liyanage, M. Ylianttila, A. Gurtov, Securing the control channel of software-defined mobile networks, in: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, IEEE, pp. 1–6, 2014.

[180] Y. Jarraya, A. Shameli-Sendi, M. Pourzandi, M. Cheriet, Multistage ocdo: Scalable security provisioning optimization in sdn-based cloud, in: 2015 IEEE 8th International Conference on Cloud Computing, IEEE, pp. 572–579, 2015.

[181] M. Sainz, I. Garitano, M. Iturbe, U. Zurutuza, Deep packet inspection for intelligent intrusion detection in softwaredefined industrial networks: A proof of concept, Logic Journal of the IGPL 28 (4), 461–472, 2020.

[182] A. Bremler-Barr, Y. Harchol, D. Hay, Y. Koral, Deep packet inspection as a service, in: Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, pp. 271–282, 2014.

[183] Y. Li, R. Fu, An parallelized deep packet inspection design in software defined network, in: Proceedings of 2nd International Conference on Information Technology and Electronic Commerce, IEEE, pp. 6–10, 2014.

[184] A. Abubakar, B. Pranggono, Machine learning based intrusion detection system for software defined networks, in: 2017 seventh international conference on emerging security technologies (EST), IEEE, pp. 138–143, 2017.

[185] C. Yu, J. Lan, J. Xie, Y. Hu, Qos-aware traffic classification architecture using machine learning and deep packet inspection in sdns, Procedia computer science 131, 1209–1216, 2018. https://doi.org/10.1016/j.procs.2018.04.331.

[186] M. Bouet, J. Leguay, V. Conan, Cost-based placement of virtualized deep packet inspection functions in sdn, in: MILCOM 2013-2013 IEEE Military Communications Conference, IEEE, pp. 992–997, 2013.

[187] M. Bouet, J. Leguay, T. Combe, V. Conan, Cost-based placement of vdpi functions in nfv infrastructures, International Journal of Network Management 25 (6), 490–506, 2015. https://doi.org/10.1002/nem.1920.

[188] J. Kim, J. Lee, J. Kim, J. Yun, M2m service platforms: Survey, issues, and enabling technologies, IEEE Communications Surveys & Tutorials 16 (1), 61–76, 2013. https://doi.org/10.1109/SURV.2013.100713.00203

[189] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, Z. Han, Capsule network assisted iot traffic classification mechanism for smart cities, IEEE Internet of Things Journal 6 (5), 7515–7525, 2019. https://doi.org/10.1109/JIOT.2019.2901348.

[190] E. Bertino, N. Islam, Botnets and internet of things security, Computer 50 (2), 76–79, 2017. https://doi.org/10.1109/MC.2017.62.

[191] 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.

[192] Y. Zhang, N. Meratnia, P. Havinga, Outlier detection techniques for wireless sensor networks: A survey, IEEE communications surveys & tutorials 12 (2), 159–170, 2010. https://doi.org/10.1109/SURV.2010.021510.00088.

[193] J. Wang, Q. Kuang, S. Duan, A new online anomaly learning and detection for large-scale service of internet of thing, Personal and Ubiquitous Computing 19 (7), 1021–1031, 2015.

[194] H. Sun, X. Wang, R. Buyya, J. Su, Cloudeyes: Cloud-based malware detection with reversible sketch for resourceconstrained internet of things (iot) devices, Software: Practice and Experience 47 (3), 421–441, 2017. https://doi.org/10.1002/spe.2420.

[195] D. Oh, D. Kim, W. W. Ro, A malicious pattern detection engine for embedded security systems in the internet of things, Sensors 14 (12), 24188–24211, 2014. https://doi.org/10.3390/s141224188.

[196] S. O. Amin, M. S. Siddiqui, C. S. Hong, J. Choe, A novel coding scheme to implement signature based ids in ip based sensor networks, in: 2009 IFIP/IEEE International Symposium on Integrated Network Management-Workshops, IEEE, pp. 269–274, 2009.

[197] H. Sedjelmaci, S. M. Senouci, T. Taleb, An accurate security game for low-resource iot devices, IEEE Transactions on Vehicular Technology 66 (10), 9381–9393, 2017. https://doi.org/10.1109/TVT.2017.2701551.

[198] D. Midi, A. Rullo, A. Mudgerikar, E. Bertino, Kalis—a system for knowledge-driven adaptable intrusion detection for the internet of things, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, pp. 656–666, 2017.

[199] Y. Lee, Y. Lee, Toward scalable internet traffic measurement and analysis with hadoop, ACM SIGCOMM Computer Communication Review 43 (1), 5–13, 2012. https://doi.org/10.1145/2427036.2427038.

[200] M. Wullink, G. C. Moura, M. M¨uller, C. Hesselman, Entrada: A high-performance network traffic data streaming warehouse, in: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, IEEE, pp. 913-918, 2016.

[201] C. Orsini, A. King, D. Giordano, V. Giotsas, A. Dainotti, Bgpstream: a software framework for live and historical bgp data analysis, in: Proceedings of the 2016 Internet Measurement Conference, pp. 429–444, 2016.

[202] M. Becchi, M. Franklin, P. Crowley, A workload for evaluating deep packet inspection architectures, in: 2008 IEEE International Symposium on Workload Characterization, IEEE, pp. 79–89, 2008.

[203] F. Schneider, J. Wallerich, A. Feldmann, Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware, in: International Conference on Passive and Active Network Measurement, Springer, pp. 207–217, 2007.