Advances in Science, Technology and Engineering Systems Journal
Vol. 10, No. 6, 1-22 (2025)
www.astesj.com
Special Issue on Innovation in Computing, Engineering Science & Technology

ASTES Journal
ISSN: 2415-6698

TIMeFoRCE: An Identity and Access Management Framework for IoT Devices in A Zero Trust Architecture

Vinton Morris[*,1], Kevin Kornegay[2], Joy Falaye[1], Sean Richardson[1], Marcial Tienteu[1], Loic Jephson Djomo Tchuenkou[1]

[1]*Morgan State University, Cybersecurity Assurance and Policy Center, Baltimore, 21251, United States*

[2]*Morgan State University, Electrical and Computer Engineering, Baltimore, 21251, United States*

ARTICLE INFO

Article history:
Received: 01 October, 2025
Revised: 19 October, 2025
Accepted: 21 October, 2025
Online: 10 November, 2025

Keywords:
Zero Trust
Zero Trust Architecture
Identity and Access Management
Access Control
Fingerprinting
Authentication
Policy Enforcement Point
Policy Decision Point

ABSTRACT

Zero Trust Architecture offers a transformative approach to network security by emphasizing ”never trust, always verify.” IoT devices, while increasingly integral to modern ecosystems, pose unique challenges for identity management and access control due to their constrained processing power, memory, and energy capabilities. In a Zero Trust framework, every IoT device is treated as a potential security risk, necessitating continuous, adaptive authentication and strict access control policies. Despite their limited resources, IoT devices must undergo identity verification and operate within the principle of least privilege, granting access only to specific services and data based on each device’s role, context, and risk profile. This paper examines the challenges of implementing Zero Trust in IoT environments, focusing on scalable identity management and access control mechanisms that account for the inherent resource constraints of IoT devices while ensuring robust security against emerging threats. Previous works highlighted the inherent issues with IoT devices in a Zero Trust environment; however, they offer no viable solution. This paper proposes Time-based Identity Management and Flow Rule Control Engine (TIMeFoRCE), an identity management and access control solution that satisfies the tenet of Zero Trust for resource-constrained IoT devices through time-based authentication. This work builds upon prior research and provides metrics to show the solution’s efficacy.

1. Introduction

This paper is an extension of work originally presented in the 2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC) [1]. IoT devices, while integral to modern ecosystems, are inherently constrained by their limited computational resources, including processing power, memory, and energy capacity [2, 3, 4]. These limitations make traditional security models, which often assume more robust device capabilities, unsuitable for IoT environments [5, 6, 7]. As a result, securing these devices against an ever-increasing array of cyber threats has become a pressing concern. One promising solution to these challenges is the implementation of Zero Trust Architecture (ZTA) [5, 8, 9, 10, 11]. ZTA operates on the principle of ”never trust, always verify,” where no entity, whether internal or external to the network, is automatically trusted [5, 12, 13]. In the context of IoT, this approach is crucial because it treats every device as a potential security risk, regardless of its location on the network [8]. One of the fundamental tenets of Zero Trust is continuous adaptive authentication and strict access control for all devices, including those with limited resources [8]. Despite the inherent limitations of IoT devices, ZTA requires them to undergo identity verification and adhere to the principle of least privilege, granting access only to specific services and data based on the device’s role, context, and associated risk profile [8, 14].

This paper addresses the inherent limitations of IoT devices by proposing a novel approach to identity and access management, specifically designed for resource-constrained IoT devices within a Zero Trust framework. By leveraging behavior-based fingerprinting and time-based authentication, we aim to offer a lightweight yet effective solution that balances security with the device limitations inherent to the IoT landscape by removing the security responsibility from devices. Vendors such as Cisco, Microsoft, Palo Alto, Armis, and Zscaler [15, 16, 17, 18, 19] provide access control solutions for IoT; however, these proprietary black-box solutions with limited visibility simply exist for financial gains and require significant

[*]Corresponding Author: Vinton Morris, Email: vinton.morris@morgan.edu

financial investment and infrastructure change. TIMeFoRCE is one that can be deployed without incurring significant overhead. We performed a comprehensive analysis of the proposed architecture's efficacy through detailed metrics and performance evaluations. In doing so, we aim to advance the state of the art in securing IoT environments, ensuring that they remain resilient against emerging threats while maintaining the integrity of the underlying infrastructure.

Our contribution presents Time-based Identity Management and Flow Rule Control Engine (TIMeFoRCE), an Identity and Access Management (IAM) framework tailored to support IoT devices within a ZTA. The main contributions of this paper are:

- Proposed *TIMeFoRCE*, a novel Identity and Access Management (IAM) system that authenticates IoT devices by extracting discriminative features from network packet headers to generate unique device fingerprints, which are matched against a reference database to validate devices attempting communication.
- Introduced a continuous identity verification and dynamic access control mechanism that integrates time-based authentication, software-defined networking (SDN), and behavioral fingerprinting, thereby mitigating reliance on pre-shared credentials and static configurations.
- Presented a comprehensive evaluation using a testbed of 26 IoT devices under conditions representative of real-world network environments. The system employs machine learning to analyze traffic patterns and classify newly introduced devices in real time.
- Demonstrated that the IAM system enforces authentication and access control policies through continuous traffic monitoring and adaptive interaction with the SDN switch, enabling context-aware security decisions based on device behavior.

The remainder of this paper is structured as follows: Section 2 discusses related work. Section 3 details our methodology and experimental setup. Section 4 presents our results. Sections 5 and 6 conclude the paper and discuss future directions.

2. Related Work

This research spans several areas, including device identification, Zero Trust, microsegmentation, and software-defined networking to develop TIMeFoRCE and are organized accordingly.

2.1. *Device Identification*

Extensive research has been conducted on device identification and classification [3, 20, 21, 22, 23, 24, 25, 26, 27], and show the various ways to identify devices. However, minimal research has been conducted in the Zero Trust space on the use of such identities for identity management.

One of the more promising works in the area of IoT device identification is by [28]. In [28], the author evaluated their IoT device identification method using two publicly available datasets: the Aalto University IoT dataset and the USNW dataset. With their mixed approach, they achieved accuracies of 94% on the UNSW dataset and 83% on the Aalto dataset. Unlike [28], who grouped similar devices (TP-LinkPlugHS100 and TP-LinkPlugHS110) to form their aggregate model and improve accuracy, our work grouped identical devices (make and model). For example, in our study, two Sengled light bulbs were treated as distinct devices and not grouped because of a color difference (soft white vs. daylight). The work of [28] focused on device classification, while our work focuses on developing device identities and using them to satisfy the identity and access management tenet of a ZTA.

2.2. *Zero Trust*

While many studies [29, 30, 31, 32, 33] discuss the principles and components of Zero Trust, few provide practical implementations or concrete methods for utilizing such identities for authentication. The authors in [31], investigated global Zero Trust research and its application in IoT environments. In [32], the authors examined advancements in biometrics, blockchain, and AI to enhance IoT security through improved authentication, and explored the integration of Zero Trust principles into IoT frameworks to mitigate cyber threats and strengthen resilience; offering insights from recent research and real-world implementations on the evolution of secure IoT authentication. In [33], the authors sought to reinterpret Zero Trust from a network security perspective by exploring how elements of trust still exist within Zero Trust frameworks; defining the concept and key characteristics of trust in Zero Trust systems, establishing foundational principles, and discussed future research trends and applications across various scenarios. These authors focused primarily on providing a comprehensive review of current literature and Zero Trust trends.

In [34], the authors present results from a cybersecurity test bed, which incorporates elements of a Zero Trust data communication network through the use of autonomic OODA loops. They demonstrate results from experiments in which identity management was integrated with automated threat response and packet-based authentication, alongside the dynamic management of eight unique network trust levels. In their testing environment, they deployed a dynamic orchestration of firewall access control lists (ACLs) and incorporated authentication gateways, each configured with its own dynamic trust levels. The authentication mechanisms employ First Packet Authentication (FPA) alongside Transport Access Control (TAC), with explicit trust established by generating a network identity token at the start of each session. They reported that the system successfully detected and mitigated DDoS attacks, blocking over 100 unauthorized access attempts within 60 seconds. However, despite their effectiveness, IoT devices typically operate in an insecure manner and lack identities that can be tied to external authentication providers.

In [5], the authors conducted a comprehensive review of the literature on Zero Trust, identifying current knowledge and research gaps. They presented relevant literature on Zero Trust. They aim to systematically find new research streams and organize the body of knowledge on Zero Trust. They determined that the majority of the literature they reviewed highlighted the benefits of Zero Trust but did not discuss its potential shortcomings. They noted that although experts contend that implementing Zero Trust principles produces a highly scalable infrastructure, no design for such massive networks has been put forth, proven, or tested to date. They also highlighted that they found no research into sectors such as healthcare or energy, where security is paramount. While they conducted an in-depth review, no relevant solution was proposed to address the inherent lack of identity and access management (IAM) in IoT devices.

The authors in [9], presented the core principles of Zero Trust using a descriptive approach and also reviewed a variety of approaches to effectively implement this paradigm. In addition to providing a thorough analysis of cutting-edge authentication and access control methods across various contexts, they explain the role of these technologies in ZTA. They also go into great detail about standard encryption methods, security automation, and microsegmentation that can be used to implement a ZTA. The paper also examines several challenges that may impede the practical implementation of Zero Trust, including issues with contemporary authentication methods, access control systems, trust and risk assessment frameworks, microsegmentation strategies, and software-defined perimeter solutions. They also identified potential future research directions for the successful realization of zero trust in critical infrastructures. Our work focuses on developing a method for authenticating devices using discrete identities.

In [6], the authors sought to bridge the current knowledge gap and investigate the complexities of the Zero Trust framework. Like [5], [9], [31], and [33] they reviewed the ZTA literature and offered a basic analysis of its application and efficacy in light of earlier research. They examined the advantages and disadvantages of the Zero Trust security architecture, provided a general overview of the model, and addressed the knowledge gap regarding the effectiveness of implementing a Zero Trust philosophy. They supported the widely held belief that Zero Trust has numerous meanings. Like the previous authors, they also focus on the component that equates to a robust Zero Trust policy, but no solution to address the inherent problems.

In [35], the authors presents ZTA-IoT, a novel Zero Trust Architecture tailored for IoT systems, designed to extend and adapt the NIST ZTA framework for cloud-enabled IoT environments. They further proposes the ZTA-IoT Access Control Framework (ZTA-IoT-ACF) to manage diverse interactions among IoT layers and components, to reduce implicit trust and reinforcement of authentication and authorization mechanisms. Additionally, they presents the Object-Level Zero Trust Score-Based Authorization Framework (ZTA-IoT-OL-SAF), which governs access to devices and data objects through dynamic, context-aware authorization based on real-time calculated trust scores. The framework supports continuous authorization decisions by evaluating multiple attributes related to actors, targets, and actions. The study culminates in the UCONIoT model, a formally defined usage control policy governing user-to-object and device-to-object interactions validated through a proof-of-concept implementation on AWS IoT using Lambda functions and DynamoDB. The results highlight the potential of fine-grained, score-based access control in enhancing IoT security. Several concepts were introduced; however, they were later presented as future work. This include; Real-time dynamically calculated score and threshold values, virtual-object level, the cloud level, the application level interactions, and usability. Additionally, the study focused heavily on access control and not authentication. While the work appears to provide effective access control, it contrasts with TIMeFoRCE, which is capable of operating in real time.

The National Institute of Standards and Technology (NIST) special publications 800-207 [8] and 800-207A [36] outline the core principles of any ZTA implementation, and are the primary documents referenced by researchers and practitioners. A thorough set of Zero Trust principles and a referenced ZTA for bringing those ideas to life are outlined in NIST Special Publication 800-207. A

Table 1: Feature Comparison of prior works across device identification, microsegmentation, and SDN controller functionality externalization with TIMeFoRCE.

Work	Scalability	IoT Device Support	Behavioral Identity / SDN Integration	Performance / Strengths
IoTDevID [28]	✓	✓	✗	High classification accuracy on public IoT datasets; behaviour-based device identification; limited to offline analysis and non-dynamic enforcement.
Zero Trust Microsegmentation [29]	✓	✗	✗	Improves Zero Trust posture via fine-grained segmentation and workload isolation; achieves 60–90% reduction in lateral movement risk; scalability policy tradeoff present.
Externalization of Packet Processing in SDN [30]	✓	✗	✓	Architectural scalability through distributed control-plane microservices; minimal performance overhead; no behavioral identification or access control capability.
TIMeFoRCE (This Work)	✓	✓	✓	Provides **real-time authentication and access control** using behavioral fingerprints integrated within SDN; supports continuous Zero Trust validation; scalable to large heterogeneous IoT ecosystems.

key paradigm shift in obtaining a ZTA involves transitioning from traditional security models that enforce segmentation and isolation based on network constructs to models that enforce policy decisions primarily based on identity [8].

While the previous literature identifies some of the benefits of Zero Trust, most do not offer a meaningful solution, especially regarding authentication. Most of the Zero Trust solutions we find exist in the commercial space and are focused on financial gains. Organizations such as [16], [37], [38], and [39] offer IoT platforms; however, these are designed for IoT manufacturers as cloud platforms for user interaction with devices or as aggregation points for analytics.

2.3. Microsegmentation

In [29], the authors conducted an impact assessment to develop an analytical framework for characterizing and quantifying the efficacy of microsegmentation in enhancing network security. Their methodology employed a dual graph-feature-based framework that integrated network connectivity and attack graphs to assess network exposure and robustness. To evaluate robustness, they used MulVAL, incorporating inputs such as Nessus XML data and network firewall rules to generate the network attack graph. The study examined a range of metrics, including the number of misconfigurations, counts of shortest paths, average and minimum shortest path lengths, minimum shortest path counts, and average and maximum out-degrees, as well as average betweenness. Their findings indicate that the average and maximum out-degrees of compromised network privileges substantially illustrate the impact of microsegmentation in constraining lateral movement and exploration by attackers, thereby reducing the potential number of attacks by 93% and 69%, respectively. Furthermore, microsegmentation reduced average betweenness by over 98% and altered the network topology, resulting in a more linear distribution of betweenness. In [29], the authors focused on developing an evaluation framework for microsegmentation, a tenet of Zero Trust; however, our work focuses on identity management, which authenticates devices communicating within the network.

2.4. Software-Defined Networking

For our research framework, we examined the work of [20] and [28] in the context of device identification, the work of [29] in microsegmentation, and the contributions of [30] in the SDN domain. In [30], the authors present an architecture that decouples controller functionality and externalizes packet processing, thereby decentralizing microservices at the control plane level. Their goal was to disaggregate core subsystems of SDN controllers into cooperative microservices, allowing flexibility in programming language selection and converting a monolithic control plane into a microservice-based architecture. They also integrated support for external reactive applications beyond traditional SDN APIs. To achieve this, they propose using Kafka as an event distribution system to transmit incoming packets from network devices to external management apps. ICMP packets were transmitted to evaluate the system, and the corresponding response times were recorded for analysis. Although this work aligns closely with ours, it focuses on measuring response time rather than addressing an actual use case. Additionally, the preliminary work reported in [1] presents ongoing efforts to identify and manage access to IoT devices in a ZTA. Table 1 provides a comparison of TIMeFoRCE in terms of scalability, device type, identity methods, and performance. Table 1 compares four approaches, IoTDevID, Zero Trust Microsegmentation, Externalized SDN Processing, and TIMeFoRCE across scalability, IoT device support, behavioral identity integration, and performance. TIMeFoRCE distinguishes itself by providing real-time behavioral fingerprint-based authentication and access control within SDN, demonstrating both adaptability and scalability across diverse environments.

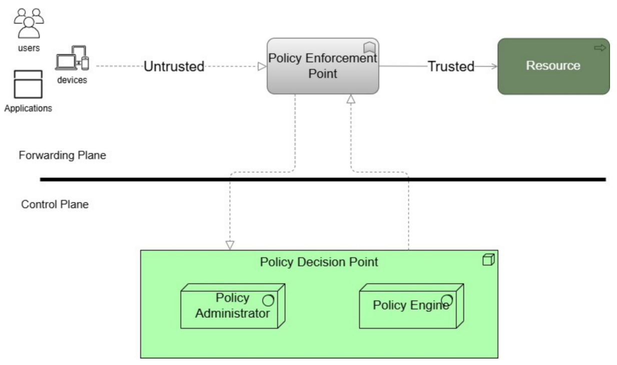


Figure 1: Zero Trust Architecture covering the policy enforcement point and policy decision point with policy administrator and policy engine.

3. Methodology

To present our Identity and Access Management (IAM) solution (i.e., TIMeFoRCE), we divided the implementation into several Phases. We first investigated and created the testbed. Second, data collection was performed on the testbed. Next, we performed device classification during the analysis phase. Following the analysis phases, we implemented the proposed solution. Finally, metrics were obtained on the performance of the proposed solution. The first three items were performed in [1] as part of a work-in-progress paper.

3.1. The Simplified Zero Trust Architecture

A ZTA is often depicted as two distinct components: the control and data plane. The control plane serves as the Policy Decision Point (PDP), where security policies are defined and decisions are made, while the data plane functions as the Policy Enforcement Point (PEP), where these policies are enforced during data transmission and access requests [8]. The PDP can be further broken down into the Policy Administrator (PA) and the Policy Engine (PE). Figure 1 portrays a simplified ZTA showing the delineation of control and forwarding/data plane. TIMeFoRCE is integrated into the PDP, where it evaluates and determines the outcomes of access requests.

3.2. IoT Testbed

For the study, the Linksys WRT 3200ACM, a dual-band MU-MIMO Gigabit Wi-Fi router [40] was utilized as the infrastructure, due to its architectural advantages, compatibility with OpenWRT, support for the Open vSwitch (OVS) package, and wireless isolation feature as outlined in [1]. For this study, primarily home and small-office IoT devices were the focus, as they are readily available as commercial-off-the-shelf (COTS) devices. A total of 26 IoT devices, representing various types, such as switches, plugs, light bulbs, sensors, media, Gateway hubs, and home assistants, typically deployed in real-world environments, comprise the testbed based on [1], are presented in Figure 2 and Table 2.

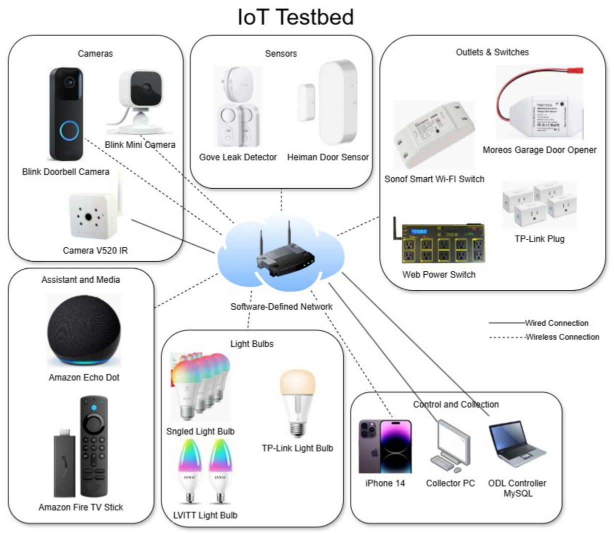


Figure 2: IoT testbed showing the various device types used in the experiment

Table 2: IoT devices in the testbed listing device type, quantity of each, and communication protocol

Device Name	Device Type	Quantity	Protocol
Amazon Alexa	Home Assistant	1	WiFi
Amazon Fire TV	Media	1	WiFi
Blink Doorbell Camera	Camera	1	WiFi
Blink Mini Camera	Camera	1	WiFi
Camera VR520 IR	Camera	1	Ethernet
Govee Gateway	Gateway	1	WiFi
Govee Leak Detector	Sensor	2	RF
Heiman Door Sensor	Sensor	3	WiFi
LVWIT Light Bulb	Light Bulb	2	WiFi
Sengled Light Bulb	Light Bulb	5	WiFi
Sonof WiFi Smart Switch	Smart Switch	1	WiFi
TP-Link Light Bulb	Light Bulb	1	WiFi
TP-Ling Plug	Smart Plug	4	WiFi
Web Power Switch	Smart Switch	1	WiFi

Figure 3 presents the internal architecture of the Linksys WRT 3200ACM after installing OVS and configuring the various ports. The Linksys WRT 3200ACM, when operating with the OpenWRT firmware version 22.03.2, utilized the Distributed Switching Architecture (DSA), which provides a Linux network interface for these user ports (lan1 - lan4 and wan), known as 'slave' interfaces [41], allowing the devices to function as a traditional layer two switch and forward all packets that it receives.

3.3. Data Collection and Analysis

3.3.1. Data Collection

Data collection began during the commissioning process of introducing the IoT devices to the testbed [1]. Data were collected for approximately 20 days over 3 months, with devices added at various stages of the capture period. During this stage, the required data volume was determined, and the most suitable features for subsequent analysis [1] were identified. Specifically, key packet header attributes essential for device identification were identified, along with additional features beyond the packet header, derived through feature engineering that could further improve this process.

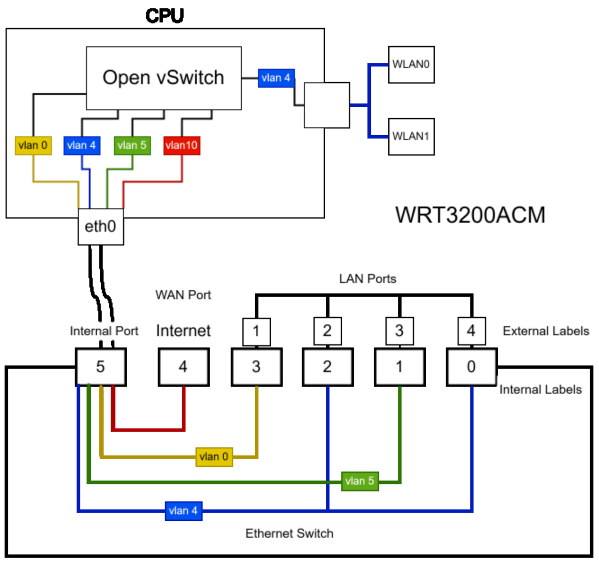


Figure 3: Linksys WRT 3200ACM Router showing the internal switch architecture after installing OVS

3.3.2. Data Analysis

Our intention was not to develop our own identification and classification algorithm but to leverage existing techniques based on previously completed scholarly work as a foundation for the research. We reviewed several such works that focused on packet-based classification [2, 28, 42, 20], as well as flow-based classification [22, 43]. The scope was narrowed to research on packet-based classification, specifically the work by [28]. However, gaps in the research prevented the use of their method as a viable solution in its current state, particularly when dealing with data the model has not encountered during its training phase. The authors in [1] provide a framework for the proposed method.

To address the limitations identified in [28], the feature set was expanded to include the z-scores of both packet size and payload for each packet. Additional binary features were introduced to indicate the presence of specific protocols, including Internet Protocol

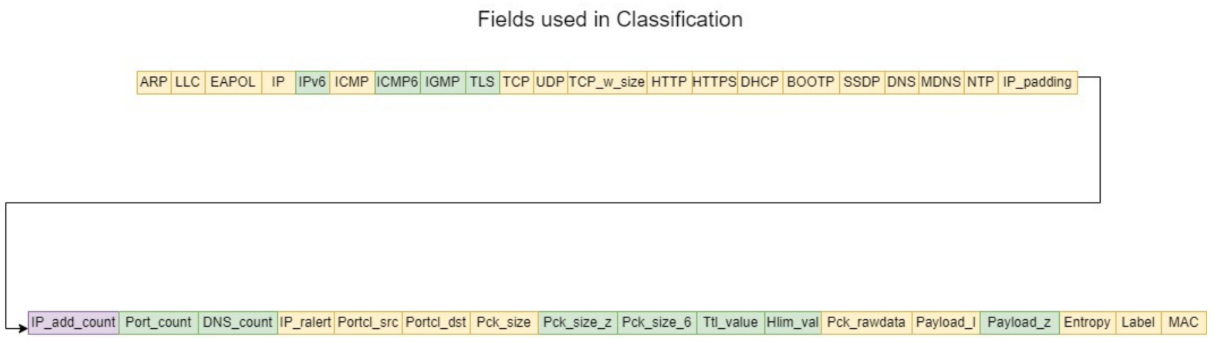


Figure 4: Feature set utilized in the classification process, showing original features and new features added.

version 6 (IPv6), Internet Group Management Protocol (IGMP), and Transport Layer Security (TLS). Furthermore, the Time-to-Live (TTL), Header Limit (HLIM), and packet size for IPv6 packets were incorporated. These IPv6-related features were added to enhance support for IPv6 traffic [1]. Figure 4 illustrates the complete set of features used in the classification process, with features from [28] shown in yellow, newly added features in green, and modified features in purple.

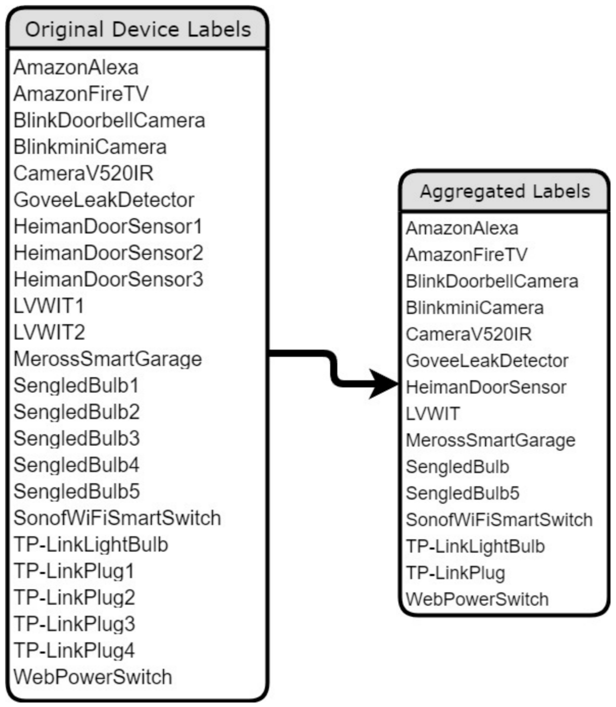


Figure 5: Original labels of devices and labels after grouping.

While adding the additional features allowed the model to perform well, utilizing the training and test data, and improving accuracy to over 99% in comparison to the 94% obtained in [28], it did not perform well with a validation set. Validation sets often contain data the trained model has not previously encountered. The validation set was a new dataset collected over eight days outside the training and test data. This is important because the identity and access management solution, TIMeFoRCE, will review this data to identify new IoT devices attempting to communicate on the network. The ”destination IP Address count” feature utilized in [28] resets to zero when parsing a new dataset. It is conceivable that, as the dataset size increases, the number of destination IP addresses will increase as well. Each time a new destination IP Address is detected, the count value is incremented by one (1). The issue is that it increments the value of the current row/packet. In a large dataset, this value can become substantial. As a result, the number of IP addresses associated with a device at the start of the dataset may differ significantly from the count observed later. This feature is essential in the classification process as a discriminatory value for each device. To overcome the limitation, we created a dictionary that maintains the counts obtained during the initial capture period. That dictionary is then used during the validation process. This dictionary updates the IP_add_count after parsing all the dataset packets. This is accomplished by using the dictionary values to update each row’s column values for specific MAC identifiers. Additionally, we added feature values for unique destination port counts and query name (QNAME) counts for each observed MAC address, further strengthening the model with more contextual information and better capturing patterns in the data. The QNAME is an important feature used during the discovery, classification, and addition of new devices. The following represents the collection of destination ports (dst_port), destination IPs (dst_ip), and query names (qname) per MAC Address, used to update their respective unique count columns.

Let:

- M be the set of unique MAC addresses.
- P_m be the set of unique destination ports visited by a MAC address m.
- I_m be the set of unique destination IPs visited by a MAC address m.
- Q_m be the set of unique query names resolved by a MAC address m.

For each observed packet t, extract:

$$(m, p, i, q) = (\text{MAC}, \text{dst\_port}, \text{dst\_ip}, \text{qname}) \tag{1}$$

$$P_m = P_m \cup \{p\}, \quad I_m = I_m \cup \{i\}, \quad Q_m = Q_m \cup \{q\} \tag{2}$$

For each MAC address m, we compute the number of unique values:

$$C_m^{IP} = |I_m|, \quad C_m^{Port} = |P_m|, \quad C_m^{Qname} = |Q_m| \tag{3}$$

where:

- C_m^{IP} represents the number of unique destination IPs visited by m.
- C_m^{Port} represents the number of unique destination ports contacted by m.
- C_m^{Qname} represents the number of unique query names resolved by m.

We achieved accuracy, precision, and recall above 99% using Random Forest (RF), as illustrated in Table 3 and based on the aggregated labels of identical devices as depicted in Figure 5. These are the labels obtained by combining identical devices into a single label. Furthermore, Figure 6 shows a comparison of dictionary utilization in TIMeFoRCE on the validation dataset. Utilizing a dictionary for the unique destination IPs and ports, and unique query names, provides for much greater accuracy. Figure 6a shows misclassifications for devices that share a similar manufacturer. For example, in Figure 6a, there is a high number of misclassifications. Several ”AmazonFireTV” devices are misclassified as ”AmazonAlexa” devices. This indicates that they share some behavioral traits. Figure 6b, on the other hand, shows that using the dictionary enables higher precision, recall, F1-score, and accuracy. Figure 6c and 6d show the classification report corresponding to Figure 6a and 6b, respectively.

Table 3: Classification Report using the aggregation method for identical model IoT Devices

Device	Precision	Recall	F1-Score	Support
AmazonAlexa	1.00	1.00	1.00	178483
AmazonFireTV	1.00	1.00	1.00	145786
BlinkDoorbellCamera	1.00	1.00	1.00	62534
BlinkminiCamera	1.00	1.00	1.00	32783
CameraV520IR	1.00	1.00	1.00	6445
GoveeLeakDetector	1.00	1.00	1.00	15755
HeimanDoorSensor	1.00	1.00	1.00	1227
LVWIT	1.00	1.00	1.00	9757
MerossSmartGarage	1.00	1.00	1.00	2938
SengledBulb	1.00	1.00	1.00	64959
SengledBulb5	1.00	1.00	1.00	2760
SonofWiFiSmart Switch	1.00	1.00	1.00	1231
TP-LinkLightBulb	1.00	1.00	1.00	3145
TP-LinkPlug	1.00	1.00	1.00	13944
WebPowerSwitch	1.00	1.00	1.00	1906
Accuracy			**1.00**	**543653**
Macro Avg	1.00	1.00	1.00	543653
Weighted Avg	1.00	1.00	1.00	543653

The classification process aims to generate unique device fingerprints, enabling each testbed device to be reliably identified by its intrinsic characteristics. Fingerprints are derived as follows. D is the set of all IoT devices. F is the set of all fingerprints. $d \in D$ is an individual IoT device represented by $f \in F$, a fingerprint. $\text{MAC}(d) \in \{0,1\}^{48}$ is the 48-bit MAC address of device d while $\vec{h} = [h_1, h_2, \ldots, h_{16}] \in \mathbb{R}^{16}$ is a vector of 16 packet header features. $\phi : \{0,1\}^{48} \times \mathbb{R}^{16} \to F$ is the fingerprint generation function that maps $\psi : F \to D$ each fingerprint to its originating device.

The fingerprint is generated as follows:

$$f = \phi(\text{MAC}(d), \vec{h}) \tag{4}$$

Where:

1. Each fingerprint is uniquely associated with a single device:
$$\forall f_1, f_2 \in F, \quad f_1 = f_2 \Rightarrow \psi(f_1) = \psi(f_2) \tag{5}$$
2. A single device may have multiple fingerprints:
$$\exists f_1, f_2 \in F, \; f_1 \neq f_2 \text{ such that } \psi(f_1) = \psi(f_2) = d \tag{6}$$
3. The fingerprint-to-device mapping is injective:
$$\forall f \in F, \quad \psi(f) = d \in D \tag{7}$$
4. The device-to-fingerprint mapping is one-to-many:
$$\forall d \in D, \quad |\psi^{-1}(d)| \geq 1 \tag{8}$$

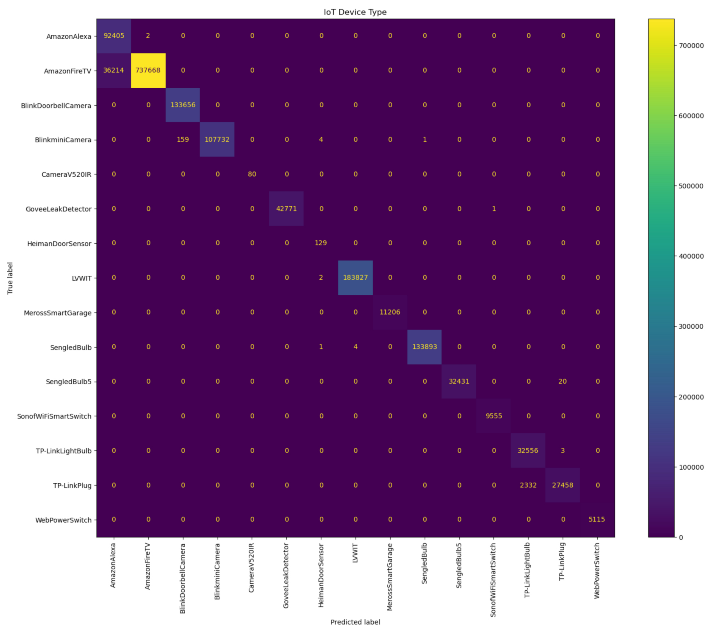
Resulting in:

$$\psi(\phi(\text{MAC}(d), \vec{h})) = d \tag{9}$$

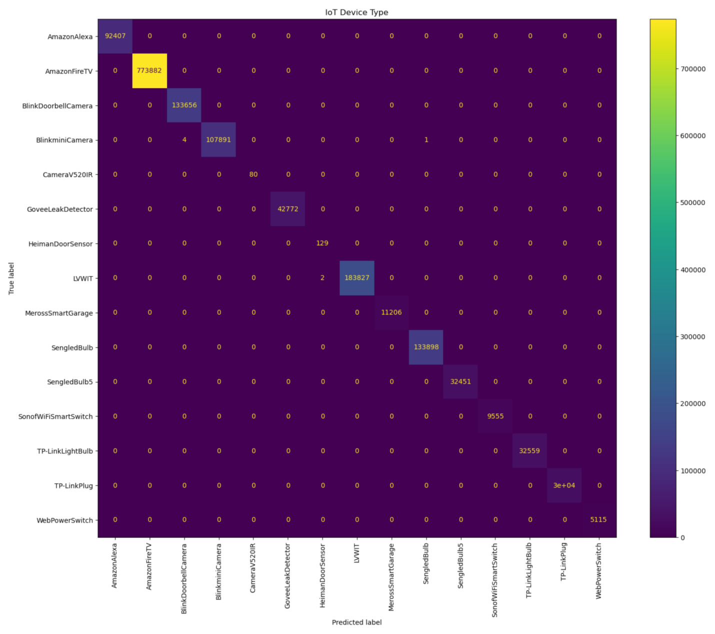
Table 4: Features used in the classification

Value Types	Feature names
Direct Values	TCP_w_size, Pck_size, Pck_size_6, payload_l
Binary Values	ARP, LLC, EAPOL, HTTP, HTTPS, DHCP, BOOTP, SSDP, DNS, MDNS, NTP, IP, IPV6, ICMP, ICMP6, IGMP, TLS TCP, UDP, IP_padding, IP_ralert, Pck_rawdata
Grouping	Portcl_src, Portcl_dst, Ttl_value, Hlim_value
Mathematical Equation	Entropy, Packet_size_z, Payload_z
Count	IP_add_count, Port_count, DNS_count

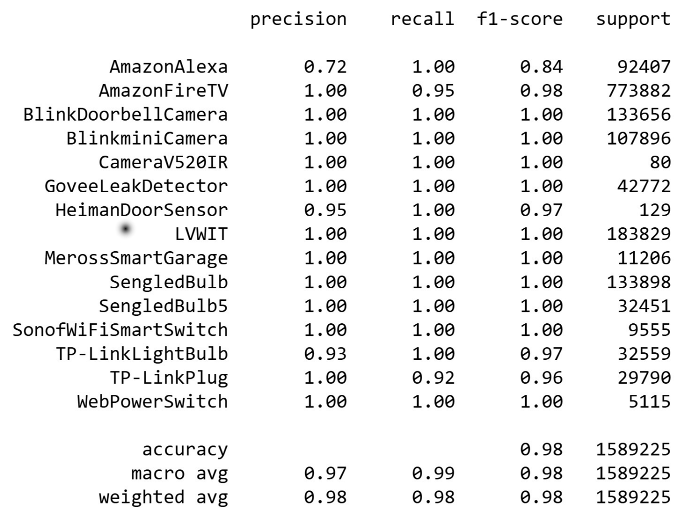
The classification was carried out using 36 features, an increase in the set of characteristics from the initial work in [1]. Table 4 presents the feature names and value types of each feature. It includes features obtained directly from the packet header as well as those derived from various feature engineering methods. Two new features added are the z-score of the packet and the payload, respectively. To obtain the z-score, the mean is needed. Given a sufficiently large dataset, the mean will reach a point where it



(a) Validation set confusion matrix without using dictionary

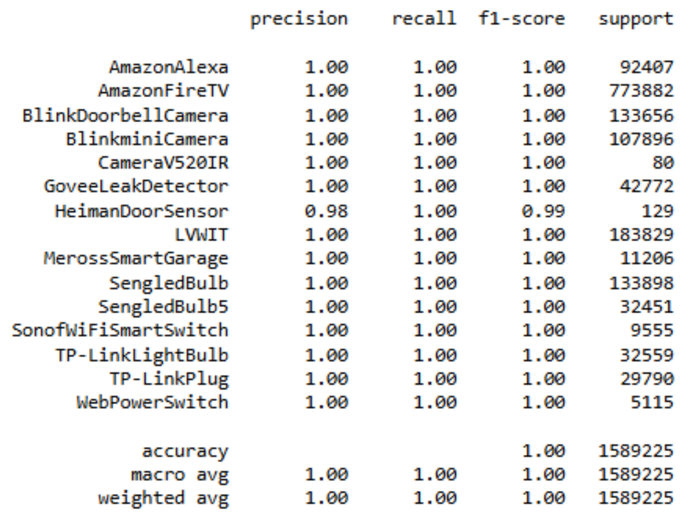


(b) Validation set confusion matrix using dictionary



```
                      precision    recall  f1-score   support

        AmazonAlexa       0.72      1.00      0.84     92407
       AmazonFireTV       1.00      0.95      0.98    773882
 BlinkDoorbellCamera       1.00      1.00      1.00    133656
     BlinkminiCamera       1.00      1.00      1.00    107896
        CameraV520IR       1.00      1.00      1.00        80
  GoveeLeakDetector       1.00      1.00      1.00     42772
   HeimanDoorSensor       0.95      1.00      0.97       129
              LVWIT       1.00      1.00      1.00    183829
  MerossSmartGarage       1.00      1.00      1.00     11206
        SengledBulb       1.00      1.00      1.00    133898
       SengledBulb5       1.00      1.00      1.00     32451
SonofWiFiSmartSwitch       1.00      1.00      1.00      9555
    TP-LinkLightBulb       0.93      1.00      0.97     32559
         TP-LinkPlug       1.00      0.92      0.96     29790
      WebPowerSwitch       1.00      1.00      1.00      5115

           accuracy                           0.98   1589225
          macro avg       0.97      0.99      0.98   1589225
       weighted avg       0.98      0.98      0.98   1589225
```

(c) Validation set classification report without using dictionary



```
                      precision    recall  f1-score   support

        AmazonAlexa       1.00      1.00      1.00     92407
       AmazonFireTV       1.00      1.00      1.00    773882
 BlinkDoorbellCamera       1.00      1.00      1.00    133656
     BlinkminiCamera       1.00      1.00      1.00    107896
        CameraV520IR       1.00      1.00      1.00        80
  GoveeLeakDetector       1.00      1.00      1.00     42772
   HeimanDoorSensor       0.98      1.00      0.99       129
              LVWIT       1.00      1.00      1.00    183829
  MerossSmartGarage       1.00      1.00      1.00     11206
        SengledBulb       1.00      1.00      1.00    133898
       SengledBulb5       1.00      1.00      1.00     32451
SonofWiFiSmartSwitch       1.00      1.00      1.00      9555
    TP-LinkLightBulb       1.00      1.00      1.00     32559
         TP-LinkPlug       1.00      1.00      1.00     29790
      WebPowerSwitch       1.00      1.00      1.00      5115

           accuracy                           1.00   1589225
          macro avg       1.00      1.00      1.00   1589225
       weighted avg       1.00      1.00      1.00   1589225
```

(d) Validation set classification report using dictionary

Figure 6: Comparison of TIMeFoRCE classification performance on unseen data with and without dictionary usage

changes little, if at all, with the addition of more packets or rows. As a result, the dataset's mean, which contains approximately 10 million packets, will be used as the classification mean in future classification.

Suppose there are n samples $x_1, x_2, \ldots, x_n$, each in the interval $[a, b]$. Define the sample mean by

$$\bar{x}_n = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{10}$$

When a new sample $x_{n+1} \in [a, b]$ is added, the updated mean is

$$\bar{x}_{n+1} = \frac{1}{n+1} \sum_{i=1}^{n+1} x_i = \bar{x}_n + \frac{x_{n+1} - \bar{x}_n}{n+1}. \tag{11}$$

Hence, the difference between the new mean and the old mean is

$$\bar{x}_{n+1} - \bar{x}_n = \frac{x_{n+1} - \bar{x}_n}{n+1}. \tag{12}$$

Since $x_{n+1}, \bar{x}_n \in [a, b]$, the numerator $|x_{n+1} - \bar{x}_n|$ is at most $b - a$, yielding

$$|\bar{x}_{n+1} - \bar{x}_n| = \left| \frac{x_{n+1} - \bar{x}_n}{n+1} \right| \leq \frac{b-a}{n+1}. \tag{13}$$

As $n \to \infty$, we have $\frac{b-a}{n+1} \to 0$, implying that once the number of samples becomes large, adding additional values in the range $[a, b]$ changes the mean by a negligibly small amount.

These are all standard practices utilized by many researchers in the IoT device identification and classification space [28, 42, 44, 45].

Creating a fingerprint for a device is akin to building a user identity in Active Directory (AD) or Azure, and using the Lightweight

Directory Access Protocol (LDAP) or Security Assertion Markup Language (SAML) as the authentication protocol. Fingerprints are proactively added to the database by taking all packets captured during the learning phase and removing certain packets, such as ARP. ARP packets are identical for all devices except for the MAC address. The dataset was reduced by removing duplicate rows and extracting features to derive the fingerprint.

3.4. Implementation

The solution was implemented using the OpenDaylight (ODL) SDN controller [46], version 0.20.1 (code name Calcium). A Flask Web Server (FWS) [47] (version 3.0.3) was employed as the external application responsible for interacting with the associated Python scripts [48]. MySQL 8 [49] served as the database (DB) repository for storing device fingerprints. An ODL Java bundle was developed using Maven [50, 51], enabling streamlined portability across compatible SDN controllers.

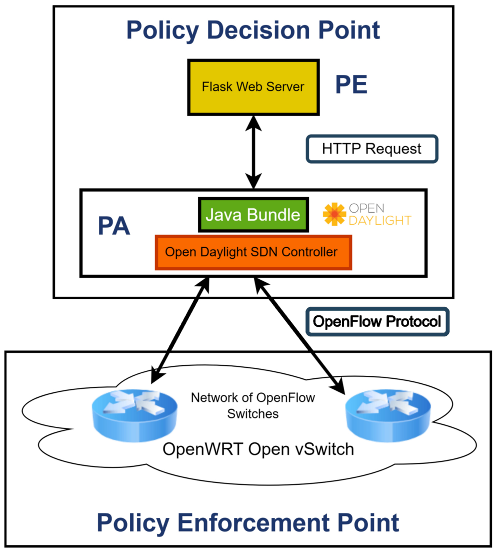


Figure 7: Proposed ZTA showing the various components that make up the architecture.

3.5. Experimental Setup

Multiple studies [27, 52, 53, 54, 55], have been completed utilizing Mininet as a virtual SDN switch; however, our study utilized the Linksys WRT 3200ACM because the testbed contains real-world devices that require a connection point [1]. A Samsung Notebook Model NP940X5J, Intel Core i7-4500U CPU @ 1.80 GHZ, 8 GB RAM with VirtualBox version 7.0.14 r161095 [56], and (2) Ubuntu 22.04.4 virtual machines functioned as the control plane of the architecture. We installed ODL Calcium 2024.01 [57] with three (3) GB RAM and one (1) CPU on the first virtual machine. MySQL 8 [49] was installed on the 2nd virtual machine with two (2) GB RAM and one (1) CPU. The MySQL 8 database was configured and populated with the fingerprints generated in section 3.3.2. This served as the fingerprints or the identity repository for the IoT devices in the testbed. Figure 7 presents the proposed ZTA showing the various components that are integrated into the PDP, while Figure 8 outlines the ZTA for the IAM solution (TIMeFoRCE), showing the process flows among the various components. The ODL controller, FWS, and DB makes up the PDP layer of the architecture. The OpenWRT OVS is the forwarding/data plane that forwards packets based on match criteria in the OpenFlow switch flow table. The control plane is the ODL SDN controller, which interacts with the OpenFlow switches via the OpenFlow protocol and the FWS. The ODL SDN controller operates via Java bundles, while the FWS uses a set of Python scripts. ODL functions as the policy administrator (PA), while FWS functions as the policy engine (PE), ultimately deciding whether to grant a given subject access to a resource. In this research, the decision is whether an IoT device is allowed to communicate.

4. Experimental Results

The previous section presented the methodology for our solution. We presented relevant literature on device identification, SDN, and Zero Trust. The team performed data analysis and completed the experimental setup. We investigated the various components that comprise the infrastructure to determine the optimal architecture. Additionally, we created the various Python scripts that the FWS (the PE) uses to perform the necessary operations. The fingerprints generated in section 3.3.2 were also added to the database. In this section, the focus is on results from the infrastructure's operational state. We examined the communication pattern of IoT devices. Additionally, several use cases are presented that demonstrate TiMeFoRCE's promise not only for authenticating network devices but also for providing access control.

4.1. Communication Patterns

An IoT device exhibits unique communication patterns. Some devices communicate quite frequently, even when not actively used, while others only communicate when a change-state event occurs. Table 5 provides the results for traffic collected over a five-hour period. 104783 packets were observed for the "AmazonFireTV" with an average interval of 0.302449 milliseconds, a maximum interval of 38.528123 seconds, and a minimum interval of 0.0000001 seconds while it was not actively utilized. Compare this to the "HeimanDoorSensor2," which observed only 35 packets with an average interval of 0.350611 seconds, a maximum interval of 3.613992 seconds, and a minimum interval of 0.000626 seconds. While the "HeimanDoorSensor2" may appear to have a lower maximum interval, it only measures traffic when it is activated; it did not see traffic again within the five-hour period. As a result, communication patterns cannot be garnered from devices like the HeimanDoorSensor2" as they operate on event-based triggers. Table 5 shows that devices can communicate from a few microseconds to several minutes.

4.2. Use Case

The following sections present three use cases for TIMeFoRCE: a legitimate device achieving successful authentication, an illegitimate device where authentication is rejected, and a new device being introduced into the network where classification is required.

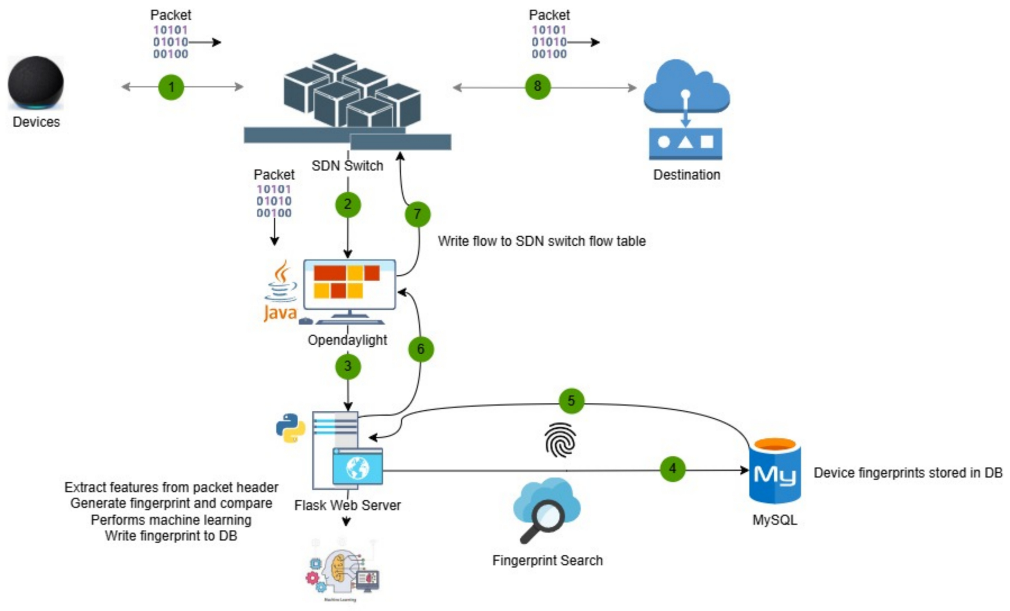


Figure 8: ZTA for Identity and Access Management (TIMeFoRCE) showing the various components that make up the architecture.

Table 5: Table showing device statistics for the number of packets observed, the average interval, the max interval, and the min interval between communication requests (intervals in seconds)

Device	Packets Sent	Avg Interval	Max Interval	Min Interval
BlinkminiCamera	8873	4.945252	65.825147	0.000005
SengledBulb2	4819	9.106290	60.037668	0.000051
SengledBulb4	8686	5.048959	49.121616	0.000054
SengledBulb3	5164	8.496532	42.142203	0.000061
AmazonAlexa	92001	0.476809	30.582975	0.000000
SengledBulb5	5289	8.272675	133.843222	0.000073
TP-LinkLightBulb	1984	22.071131	319.312205	0.000021
WebPowerSwitch	777	56.293847	502.737341	0.000066
SonofWiFiSmartSwitch	1363	32.197260	186.069789	0.000069
TP-LinkPlug1	1062	41.211247	245.402278	0.000196
TP-LinkPlug3	2612	16.782434	90.460782	0.000128
GoveeLeakDetector	3954	11.080574	102.742173	0.000147
TP-LinkPlug2	2651	16.529848	94.959352	0.000629
TP-LinkPlug4	2576	17.023061	88.578222	0.000050
MerossSmartGarage	1230	35.631615	191.624964	0.000284
BlinkDoorbellCamera	14404	3.024163	615.469474	0.000006
AmazonFireTV	104783	0.302449	38.528123	0.000000
HeimanDoorSensor2	35	0.350611	3.613992	0.000626
LVWIT1	2929	3.534108	10.117231	0.000145

4.2.1. Legitimate Device

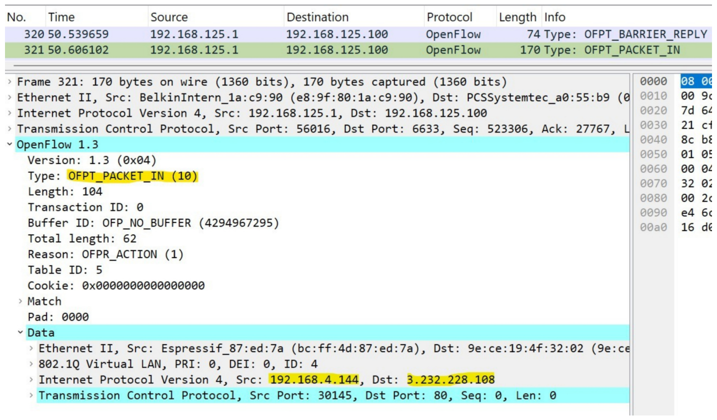
In this use case, with the ODL SDN controller, Java bundle, and FWS activated, which serves as the PDP and the central components of TIMeFoRCE, the device attempting to communicate sends a packet on the network. The OpenFlow switch or the PEP, which operates based on reactive flows, collects the packet and attempts to match it against its flow table. Due to the absence of a matching flow and a static flow rule directing certain packets to the PDP, the PEP sends the received packet to the PDP as an OFP_PACKET_IN message via the Southbound application programming interface (API). This is the original packet encapsulated in an OpenFlow packet, as illustrated in Figure 9.

The SDN controller using the Packet-Forwarder bundle (the PA) we created extracts the payload (the original packet "data") from the OpenFlow packet as a byte string, ensuring it is not a null packet. It then sends the extracted OpenFlow packet payload, "the data," to the PE via an HTTP Post request using the Northbound API. The PE receives the payload string and parses the string to obtain the packet header information. It then performs the sequence of activities as illustrated in Figure 10.

The primary activity is to generate a fingerprint from the parsed

packet header data as this is the initial task performed by TIMeFoRCE. We defined the fingerprint generated as follows:
Let $\vec{x}_i = (x_{i1}, x_{i2}, x_{i3}, \dots, x_{i16}) \in \{0, 1\}^{16}$ represent the 16 binary features extracted from the packet header of device i.
Let $\text{MAC}_i \in \{0, 1\}^{48}$ denote the 48-bit Ethernet source address of device i, obtained from the packet header.
We define the fingerprint f_i of device i as the binary concatenation of the MAC address and the feature vector:

$$f_i = \text{MAC}_i \parallel \vec{x}_i \tag{14}$$



```
No.   Time        Source          Destination       Protocol  Length Info
  320 50.539659   192.168.125.1   192.168.125.100   OpenFlow      74 Type: OFPT_BARRIER_REPLY
  321 50.606102   192.168.125.1   192.168.125.100   OpenFlow     170 Type: OFPT_PACKET_IN

› Frame 321: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits)                          0000  08 0
› Ethernet II, Src: BelkinIntern_1a:c9:90 (e8:9f:80:1a:c9:90), Dst: PCSSystemtec_a0:55:b9 (0        0010  00 9
› Internet Protocol Version 4, Src: 192.168.125.1, Dst: 192.168.125.100                             0020  7d 6
› Transmission Control Protocol, Src Port: 56016, Dst Port: 6633, Seq: 523306, Ack: 27767, L        0030  21 c
˅ OpenFlow 1.3                                                                                      0040  8c b
    Version: 1.3 (0x04)                                                                             0050  01 0
    Type: OFPT_PACKET_IN (10)                                                                       0060  00 0
    Length: 104                                                                                     0070  32 0
    Transaction ID: 0                                                                               0080  00 2
    Buffer ID: OFP_NO_BUFFER (4294967295)                                                           0090  e4 6
    Total length: 62                                                                                00a0  16 d
    Reason: OFPR_ACTION (1)
    Table ID: 5
    Cookie: 0x0000000000000000
  › Match
    Pad: 0000
  ˅ Data
    › Ethernet II, Src: Espressif_87:ed:7a (bc:ff:4d:87:ed:7a), Dst: 9e:ce:19:4f:32:02 (9e:ce
    › 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 4
    › Internet Protocol Version 4, Src: 192.168.4.144, Dst: 3.232.228.108
    › Transmission Control Protocol, Src Port: 30145, Dst Port: 80, Seq: 0, Len: 0
```

Figure 9: OpenFlow packet (OFPT_PACKET_IN) captured in Wireshark

Figure 11 presents a sample of fingerprints in the database representing the fingerprintID and deviceID for four distinct devices. A device is not limited to the number of fingerprints that it can have. The 16 features used to generate the fingerprints are listed in Table 6.

TIMeFoRCE's PE performs a series of activities. The series of activities includes parsing packets, generating fingerprints, creating a dataframe, querying the database, classifying packets, gathering values to send back to the PA, adding fingerprints to the database, generating flow IDs, and applying the table ID. These activities include:

- Parsing packets - This script parses the packet header into its various features.
- Generating fingerprints - This script generates a fingerprint by combining the MAC address and 16 binary features from the packet header.
- Create dataframe - This script creates a dataframe, which is used to classify new devices introduced into the network.
- Query the database - This script queries the database to obtain a matching fingerprint and matching deviceID.
- Classify packet - This script performs classification utilizing the previous model developed in the data analysis phase.
- Gather values to send back to PA - This script extracts information from the packet header to be sent back to the PA (i.e., the ODL controller). The values extracted from the packet header include table ID, VLAN ID, source MAC address, destination MAC address, source IP address, destination IP address, source port, and destination port.
- Generate flow ID - The PE (i.e., FWS) generates a unique flow ID that is attached to each flow. This is a randomly generated 6-digit hexadecimal string.
- Applying table ID - The PE returns the table ID value to the PA. In our example, the table ID is 5.

Table 6: Features used in fingerprint

Protocols Layers	**Protocols**
Layer 1 Protocol	MAC
Layer 2 Protocols	ARP, LLC
Layer 3 Protocols	IP, IPV6, ICMP, ICMP6
Layer 4 Protocols	TCP, UDP
Layer 7 protocols	HTTP, HTTPS, DHCP, BOOTP, SSDP, DNS, MDNS, NTP

These activities performed vary based on the observed condition of the device attempting to communicate (new vs. existing).

The MAC $M_s = \text{MAC}_{\text{src}}(P)$ and features $\mathbf{f} \in \{0, 1\}^{16}$ are obtained from the header $h(P)$ of the incoming packet P and compared to a database of know fingerprints $\mathcal{F}$, each of the form $(M_i, \mathbf{f}_i)$. Therefore, the fingerprint of P is:

$$\text{FP}(P) = (M_s, \mathbf{f}) \tag{15}$$

Define by the following match flag:

$$F = \begin{cases} 1, & \text{if FP}(P) \in \mathcal{F} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

Let M_d be the destination MAC address. If $F = 1$, install bi-directional flows:

$$\text{InstallFlow}(M_s, M_d)\text{InstallFlow}(M_d, M_s) \tag{17}$$

Upon completing the required processing, the PE returns a set of parsed values to the PA. Figure 12 illustrates the output corresponding to a successfully extracted fingerprint. The returned values include the source and destination Ethernet (MAC) addresses, source and destination IP addresses, source and destination transport-layer ports, the protocol identifier, the flow ID, and the Table ID. The flow ID is a unique, randomly generated six-digit hexadecimal string. For reverse flows, an appended character "r" is used to denote the reversed direction of the same flow identifier.

The Packet-Forwarder bundle (the PA) uses the parsed values to update the PEP flow table with bidirectional flows, as illustrated in Figure 13.

$$\begin{aligned} \text{InstallFlow}(M_s, M_d) &= \{\text{cookie} = c,\ \text{table\_id} = t,\ \dots\} \\ \text{InstallFlow}(M_d, M_s) &= \{\text{cookie} = c',\ \text{table\_id} = t',\ \dots\} \end{aligned} \tag{18}$$

It adds a flow based on the original packet and a flow with the parsed values in reverse. SDN switches operate based on bi-directional flows. Therefore, the PA must add at least two flow rules to a PEP

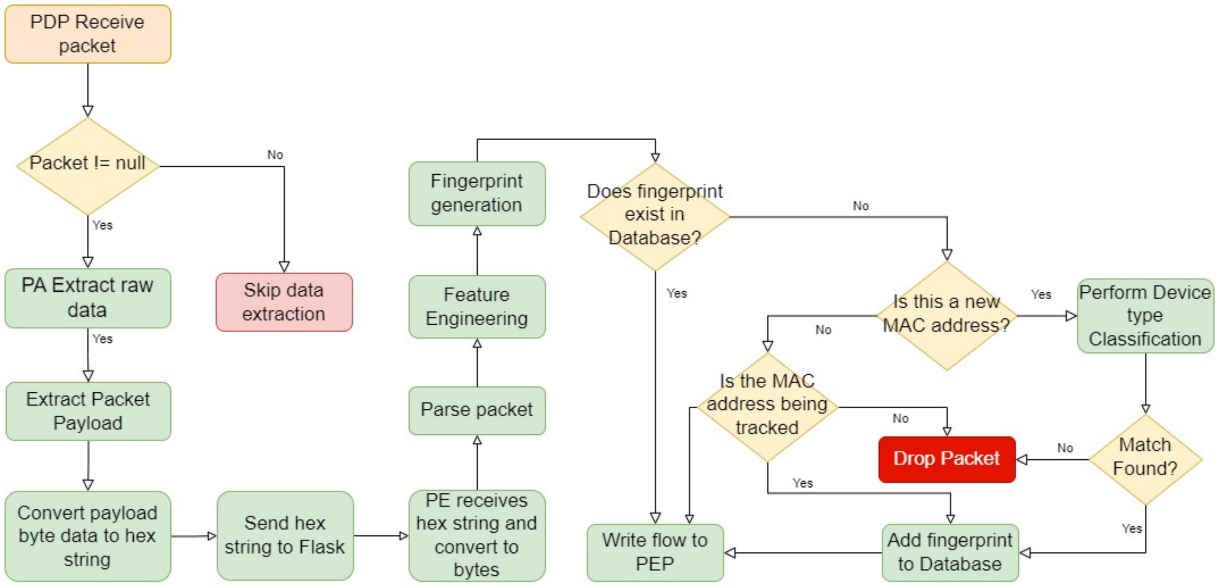
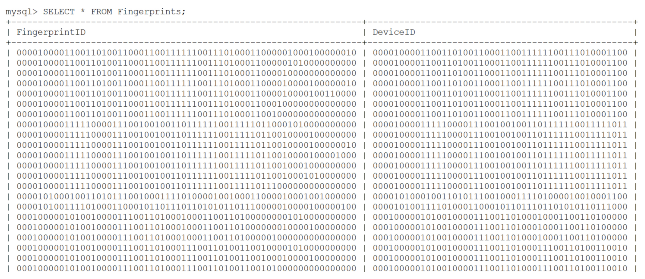
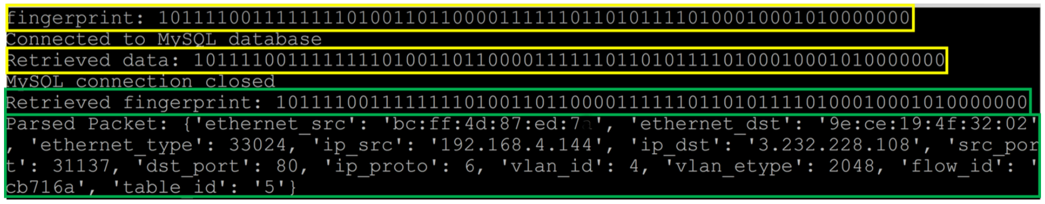


Figure 10: Flowchart depicting the sequence of activities performed by the PDP (PA and PE)



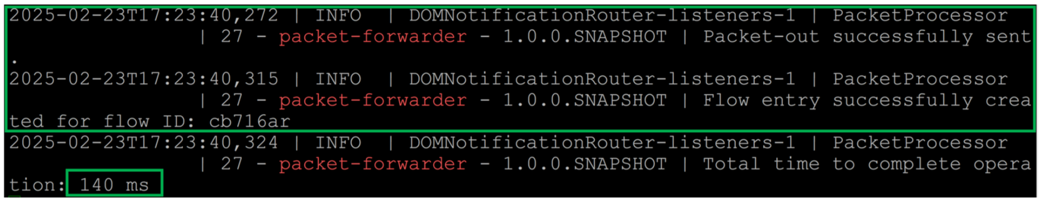
```
mysql> SELECT * FROM Fingerprints;
+------------------------------------------------------------------+------------------------------------------------+
| FingerprintID                                                    | DeviceID                                       |
+------------------------------------------------------------------+------------------------------------------------+
| 0000100001100110100110001100111111001110100011000001000100000010 | 000010000110011010011000110011111100111010001100 |
| 0000100001100110100110001100111111001110100011000001010000000000 | 000010000110011010011000110011111100111010001100 |
| 0000100001100110100110001100111111001110100011000010000000000000 | 000010000110011010011000110011111100111010001100 |
| 0000100001100110100110001100111111001110100011000010000100000010 | 000010000110011010011000110011111100111010001100 |
| 0000100001100110100110001100111111001110100011000010000100110000 | 000010000110011010011000110011111100111010001100 |
| 0000100001100110100110001100111111001110100011000100000000000000 | 000010000110011010011000110011111100111010001100 |
| 0000100001100110100110001100111111001110100011001000000000000000 | 000010000110011010011000110011111100111010001100 |
| 0000100001111100001110010010011011111100111110110001010000000000 | 000010000111110000111001001001101111110011111011 |
| 0000100001111100001110010010011011111100111110110010000100000000 | 000010000111110000111001001001101111110011111011 |
| 0000100001111100001110010010011011111100111110110010000100000010 | 000010000111110000111001001001101111110011111011 |
| 0000100001111100001110010010011011111100111110110010000100001000 | 000010000111110000111001001001101111110011111011 |
| 0000100001111100001110010010011011111100111110110010001000000000 | 000010000111110000111001001001101111110011111011 |
| 0000100001111100001110010010011011111100111110110010001010000000 | 000010000111110000111001001001101111110011111011 |
| 0000100001111100001110010010011011111100111110111000000000000000 | 000010000111110000111001001001101111110011111011 |
| 0000101000100110101110010001111010000100100011000010001001000000 | 000010100010011010111001000111101000010010001100 |
| 0000101001111010001100010110111011010101101110000010000100000100 | 000010100111101000110001011011101101010110111000 |
| 0001000000101001000011100110100010001100110100000000101000000000 | 000100000010100100001110011010001000110011010000 |
| 0001000000101001000011100110100010001100110100000000100001000000 | 000100000010100100001110011010001000110011010000 |
| 0001000000101001000011100110100010001100110100000010000000000000 | 000100000010100100001110011010001000110011010000 |
| 0001000000101001000011100110100011100110100110010000101000000000 | 000100000010100100001110011010001110011010011001 |
| 0001000000101001000011100110100011100110100110010001000010000000 | 000100000010100100001110011010001110011010011001 |
| 0001000000101001000011100110100011100110100110010100000000000000 | 000100000010100100001110011010001110011010011001 |
```

Figure 11: Sample of fingerprints in the database



```
fingerprint: 1011110011111111101001101100001111110110101111010001000101000000
Connected to MySQL database
Retrieved data: 1011110011111111101001101100001111110110101111010001000101000000
MySQL connection closed
Retrieved fingerprint: 1011110011111111101001101100001111110110101111010001000101000000
Parsed Packet: {'ethernet_src': 'bc:ff:4d:87:ed:7a', 'ethernet_dst': '9e:ce:19:4f:32:02'
, 'ethernet_type': 33024, 'ip_src': '192.168.4.144', 'ip_dst': '3.232.228.108', 'src_por
t': 31137, 'dst_port': 80, 'ip_proto': 6, 'vlan_id': 4, 'vlan_etype': 2048, 'flow_id': '
cb716a', 'table_id': '5'}
```

Figure 12: Show a successful fingerprint retrieval from the database



```
2025-02-23T17:23:40,272 | INFO  | DOMNotificationRouter-listeners-1 | PacketProcessor
                         | 27 - packet-forwarder - 1.0.0.SNAPSHOT | Packet-out successfully sent
.
2025-02-23T17:23:40,315 | INFO  | DOMNotificationRouter-listeners-1 | PacketProcessor
                         | 27 - packet-forwarder - 1.0.0.SNAPSHOT | Flow entry successfully crea
ted for flow ID: cb716ar
2025-02-23T17:23:40,324 | INFO  | DOMNotificationRouter-listeners-1 | PacketProcessor
                         | 27 - packet-forwarder - 1.0.0.SNAPSHOT | Total time to complete opera
tion: 140 ms
```

Figure 13: Shows a successful flow installation of original and reverse flow

flow table to receive return traffic. The PA adds a cookie ID to each flow and two timeout values to the flow rule: a hard timeout and a soft timeout. The PA uses the cookie ID to remove flows from the PEP after they expire, and the PA receives a FLOW_REMOVED message. The soft timeout removes the flow if no packet is received at the network device with matching criteria within a specified interval. In contrast, the hard timeout removes the flow rule after a specified amount of time that the network programmer sets, ensuring that devices will be re-authenticated periodically, fulfilling the time-based authentication requirement of a ZTA. In the experimental testing, the value of the hard timeout was twice the soft timeout. We performed several iterations to determine the optimal time for soft and hard timeouts. Traffic was captured for 20 minutes at each interval. We measured the flow table size at each interval. The bidirectional flows are written with the following parameters: cookie, table ID, idle_timeout, hard_timeout, send_flow_rem priority, Protocol, dl_vlan, dl_src, dl_dst, nw_src, nw_dst, tp_src, tp_dst, and the actions to take for the specified packet. The initial packets sent by a device provide authentication, while the flow(s) written to the PEP provide access control for the specific device. The initial packet is the only packet sent to the PDP during a communication request; subsequent packets bypass this process until the timeout expires.

The previous section outlines the steps for retrieving a successful fingerprint. Subsequent sections will present cases where the application did not obtain a successful fingerprint. This will result in one of two activities. First, if a fingerprint is not successfully identified, TIMeFoRCE will not allow communication for the specific traffic type. Second, TIMeFoRCE will further examine the device to determine if its MAC address is not in the DB.

4.2.2. *Unsuccessful Fingerprint*

The application performs the initial activities as outlined in 4.2.1 and Figure 10. In this scenario, a fingerprint was not found in the database. Figure 14 shows that it did not find a matching fingerprint. The "Retrieved fingerprint," which denotes the fingerprint returned by the query, is a 64-bit string of all zeros.

$$\mathrm{FP}(P) = \mathbf{0}_{64} \tag{19}$$

This indicates that it did not find a matching fingerprint. The next activity involves checking to see if the device has an existing presence in the database $\mathcal{D}$:

$$\mathrm{IsKnown}(M_s) = \begin{cases} 1, & \text{if } M_s \in \mathcal{D} \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

The "Retrieved data" indicates that the device is present in the database. As a result, the packet is determined not to originate from a legitimate device, or the fingerprint generated for the specific device does not match an allowed authentication fingerprint. If $\mathrm{FP}(P) = \mathbf{0}_{64}$ and $\mathrm{IsKnown}(M_s) = 1$, the device is not deemed legitimate:

$$\mathrm{Legitimate}(M_s) = 0 \tag{21}$$

The PE returns an error to the PA, which, in essence, is "null" values for the parsed data; this, in turn, skips the flow entry update of the SDN switch flow table, and the packet is subsequently dropped. Figure 15 presents the output of the PA for an unsuccessful fingerprint or illegitimate device as indicated by the skipping of the flow and reverse flow entry update.

4.2.3. *New Device*

The previous sections outline cases in which a fingerprint was successfully retrieved and a case in which it was not, for a device. Section 4.2.2 presents a case where a successful fingerprint was not identified. Part of the activities outlined in Figure 10 and 14 was to determine if the device attempting to communicate is a "new device". This is accomplished by checking the database for the device's MAC address. The device's existence in the database indicates that it is not a new device; however, $\mathrm{IsKnown}(M_s) = 0$, which demonstrates that it is, in fact, a new device. Designating a new device does not determine whether it is legitimate. Further assessment is needed.

As outlined in Figure 10, one of the questions that is asked is whether or not this is a new MAC Address. If it is determined that $\mathrm{IsKnown}(M_s) = 0$, the application will attempt to perform device type classification. The PE first extracts the features from the packet header with an initial label of "Unknown. Before device classification, it is essential to establish a ground truth reference to determine the best match accurately. Specific values result from counting and cannot be obtained from a single packet. As a result, the packet is checked against a list developed during model creation to identify specific conditions. Similar IoT devices should have similar behaviors. For example, all Amazon Alexa devices will connect to the same or a similar cloud provider. Similar devices use similar destination ports and DNS names, as presented in [22]. Using a structured approach, we can examine the similarities shared among devices and update the IP_add_count, the port_count, and the qname_count based on devices that share those traits. We used the following conditions to determine the values to assign to the device prior to classification. The match condition function C is defined as:

$$C = \begin{cases} 1 & \text{if Full MAC matches} \\ 2 & \text{if OUI} \wedge \mathrm{IP}_d \wedge P_d \wedge Q \text{ match} \\ 3 & \text{if OUI} \wedge \mathrm{IP}_d \wedge P_d \text{ match} \\ 4 & \text{if } \mathrm{IP}_d \wedge P_d \wedge Q \text{ match} \\ 5 & \text{if OUI} \wedge \mathrm{IP}_d \wedge Q \text{ match} \\ 6 & \text{if OUI} \wedge \mathrm{IP}_d \text{ match} \\ 7 & \text{if OUI} \wedge P_d \wedge Q \text{ match} \\ 8 & \text{if OUI} \wedge P_d \text{ match} \\ 9 & \text{if OUI} \wedge Q \text{ match} \\ 10 & \text{if } \mathrm{IP}_d \wedge P_d \text{ match} \\ 11 & \text{if } Q \text{ matches only} \\ 12 & \text{if OUI matches only} \\ 13 & \text{otherwise (default case)} \end{cases}$$

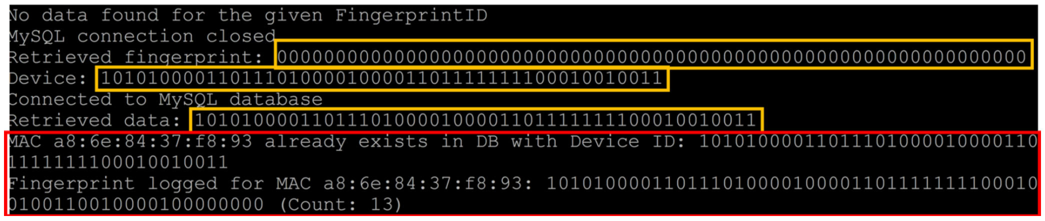
where:

- $\mathrm{MAC} \in \{0,1\}^{48}$: Full MAC address
- $\mathrm{OUI} = \mathrm{MAC}_{[0:24]}$: First 24 bits of MAC address
- IP_d: Destination IP address
- P_d: Destination port
- Q: DNS QNAME
- $C \in \{1, 2, \ldots, 13\}$: Match condition code

The packet parsing script iterates through the conditions to find the best possible match. Matching a condition does not automatically guarantee that a device will match the type whose condition is matched during the classification process.

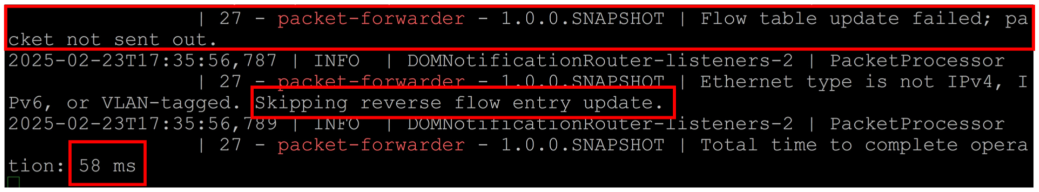
In this case, the MAC is not known, and no fingerprint is found, so device classification is performed:

$$\text{if } \mathrm{FP}(P) = \mathbf{0}_{64} \wedge \mathrm{IsKnown}(M_s) = 0 \Rightarrow \mathrm{Classify}(M_s) \tag{22}$$



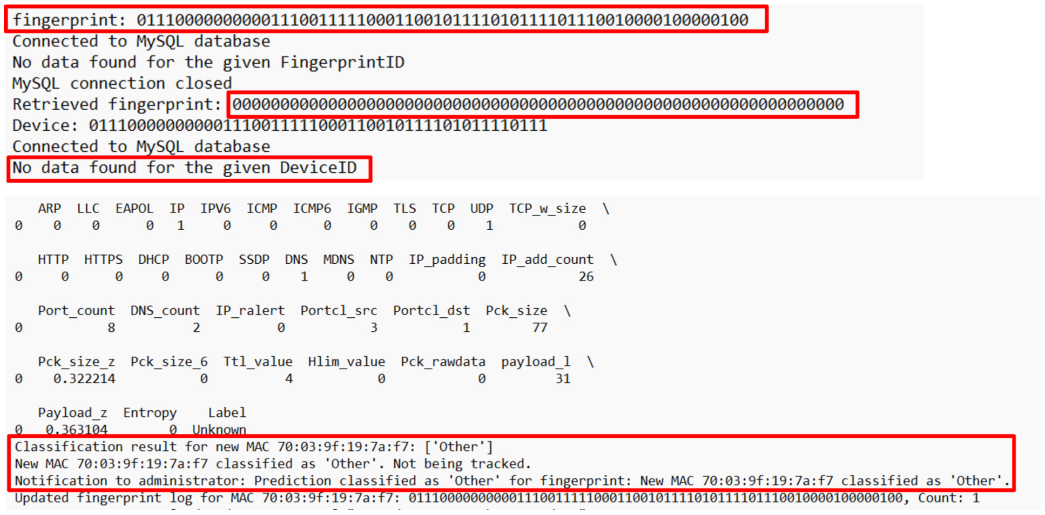
```
No data found for the given FingerprintID
MySQL connection closed
Retrieved fingerprint: 0000000000000000000000000000000000000000000000000000000000000000
Device: 1010100001101110100001000011011111111100010010011
Connected to MySQL database
Retrieved data: 1010100001101110100001000011011111111100010010011
MAC a8:6e:84:37:f8:93 already exists in DB with Device ID: 10101000011011101000010000110
1111111100010010011
Fingerprint logged for MAC a8:6e:84:37:f8:93: 1010100001101110100001000011011111111100010
0100110010000100000000 (Count: 13)
```

Figure 14: Depicts an unsuccessful fingerprint retrieval



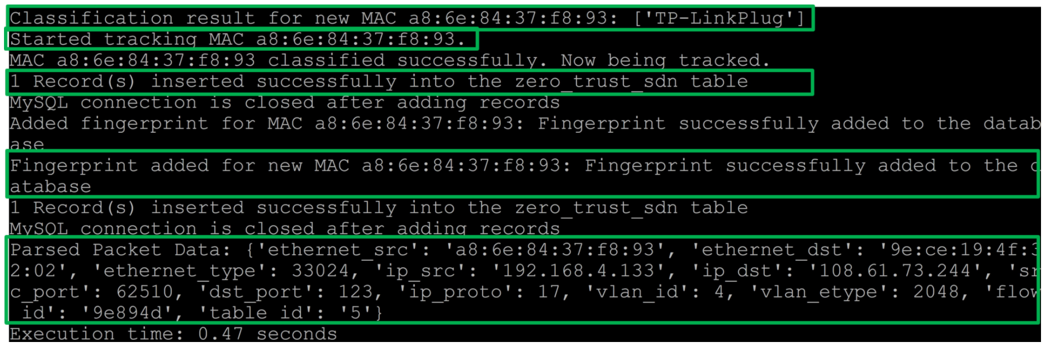
```
                | 27 - packet-forwarder - 1.0.0.SNAPSHOT | Flow table update failed; pa
cket not sent out.
2025-02-23T17:35:56,787 | INFO  | DOMNotificationRouter-listeners-2 | PacketProcessor
                | 27 - packet-forwarder - 1.0.0.SNAPSHOT | Ethernet type is not IPv4, I
Pv6, or VLAN-tagged. Skipping reverse flow entry update.
2025-02-23T17:35:56,789 | INFO  | DOMNotificationRouter-listeners-2 | PacketProcessor
                | 27 - packet-forwarder - 1.0.0.SNAPSHOT | Total time to complete opera
tion: 58 ms
```

Figure 15: Depicts an unsuccessful fingerprint output message from ODL



```
fingerprint: 0111000000000011100111110001100101111010111101110010000100000100
Connected to MySQL database
No data found for the given FingerprintID
MySQL connection closed
Retrieved fingerprint: 0000000000000000000000000000000000000000000000000000000000000000
Device: 011100000000001110011111000110010111101011110111
Connected to MySQL database
No data found for the given DeviceID

   ARP  LLC  EAPOL  IP  IPV6  ICMP  ICMP6  IGMP  TLS  TCP  UDP  TCP_w_size  \
0    0    0      0   1     0     0      0     0    0    0    1           0

   HTTP  HTTPS  DHCP  BOOTP  SSDP  DNS  MDNS  NTP  IP_padding  IP_add_count  \
0     0      0     0      0     0    1     0    0           0            26

   Port_count  DNS_count  IP_ralert  Portcl_src  Portcl_dst  Pck_size  \
0           8          2          0           3           1        77

   Pck_size_z  Pck_size_6  Ttl_value  Hlim_value  Pck_rawdata  payload_l  \
0    0.322214           0          4           0            0         31

   Payload_z  Entropy    Label
0   0.363104        0  Unknown
Classification result for new MAC 70:03:9f:19:7a:f7: ['Other']
New MAC 70:03:9f:19:7a:f7 classified as 'Other'. Not being tracked.
Notification to administrator: Prediction classified as 'Other' for fingerprint: New MAC 70:03:9f:19:7a:f7 classified as 'Other'.
Updated fingerprint log for MAC 70:03:9f:19:7a:f7: 0111000000000011100111110001100101111010111101110010000100000100, Count: 1
```

Figure 16: Show a newly discovered device classified as ”Other”



```
Classification result for new MAC a8:6e:84:37:f8:93: ['TP-LinkPlug']
Started tracking MAC a8:6e:84:37:f8:93.
MAC a8:6e:84:37:f8:93 classified successfully. Now being tracked.
1 Record(s) inserted successfully into the zero_trust_sdn table
MySQL connection is closed after adding records
Added fingerprint for MAC a8:6e:84:37:f8:93: Fingerprint successfully added to the datab
ase
Fingerprint added for new MAC a8:6e:84:37:f8:93: Fingerprint successfully added to the d
atabase
1 Record(s) inserted successfully into the zero_trust_sdn table
MySQL connection is closed after adding records
Parsed Packet Data: {'ethernet_src': 'a8:6e:84:37:f8:93', 'ethernet_dst': '9e:ce:19:4f:3
2:02', 'ethernet_type': 33024, 'ip_src': '192.168.4.133', 'ip_dst': '108.61.73.244', 'sr
c_port': 62510, 'dst_port': 123, 'ip_proto': 17, 'vlan_id': 4, 'vlan_etype': 2048, 'flow
_id': '9e894d', 'table_id': '5'}
Execution time: 0.47 seconds
```

Figure 17: Show a newly discovered device being successfully classified

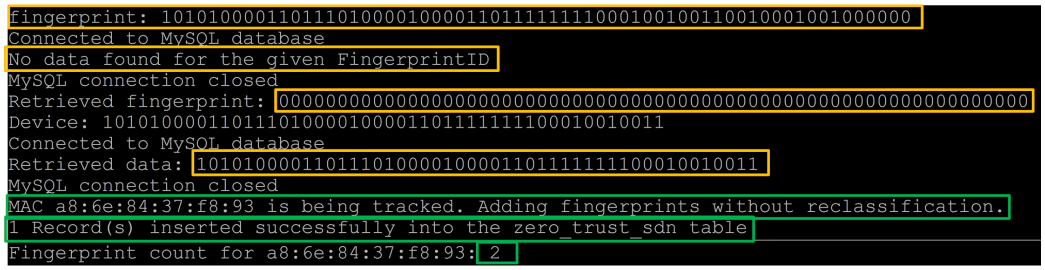
Classification is performed using a pre-trained model $\mathcal{M}$ and a condition list C:

$$\text{Classify}(M_s) = \begin{cases} \mathcal{M}(\mathbf{x}) \wedge C(\mathbf{x}), & \text{if predict\_proba}(\mathcal{M}, \mathbf{x}) \geq 0.75 \\ \text{reject}, & \text{otherwise} \end{cases} \tag{23}$$

where $\mathbf{x}$ is the feature vector extracted from P (e.g., header-based features):

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(M_s, \mathbf{f}_0)\} \tag{24}$$

In this particular example, we are adding a TP-Link Smart Plug model HS103. This device matches the device type (make and model) of a device already classified and operating within the ZTA. Device type classification will attempt to match the newly identified device type to an existing one in the database. A model was developed and stored as a pickle file during the data analysis phase.



```
fingerprint: 10101000011011101000010000110111111111000100100110010001001000000
Connected to MySQL database
No data found for the given FingerprintID
MySQL connection closed
Retrieved fingerprint: 0000000000000000000000000000000000000000000000000000000000000000
Device: 101010000110111010000100001101111111100010010011
Connected to MySQL database
Retrieved data: 101010000110111010000100001101111111100010010011
MySQL connection closed
MAC a8:6e:84:37:f8:93 is being tracked. Adding fingerprints without reclassification.
1 Record(s) inserted successfully into the zero_trust_sdn table
Fingerprint count for a8:6e:84:37:f8:93: 2
```

Figure 18: Show fingerprint added to the database for a newly discovered device where classification is bypassed

This model will be used during classification. Additionally, we used the Random Forest (RF) predict_proba function with a probability_threshold of 0.75 to identify newly discovered devices. If a newly discovered device does not fall within this threshold, it is assigned a classification of "Other" during classification.

$$\text{Classify}(M_s) = \text{"Other"} \tag{25}$$

A classification label of "Other" indicates that it was not able to successfully match the device against one that already exists via the model. In this scenario, a flow is not installed in the SDN switch's flow table representing the PEP. Figure 16 presents the results of a failed classification. This results from the absence of the device type in the classification model. Devices labeled "Other" are either illegitimate or exhibit unrecognized behavioral patterns. Thus:

$$\text{if Classify}(M_s) = \text{"Other"} \Rightarrow \text{RequiresManualReview}(M_s) = 1$$

The system then notifies the administrator:

$$\text{NotifyAdministrator}(M_s) \tag{26}$$

However, if the newly discovered device matches an existing device via the classification model, its fingerprint is added to the database for future use. Figure 17 presents the results of a successful classification. Additionally, the application will only allow one fingerprint to be added to the database. Subsequent unique fingerprints generated from the same MAC Address will be denied access due to the absence of matching fingerprints, but the MAC Address will still exist as outlined in the previous section. To overcome this limitation in the application, once a new device is successfully classified, the MAC address is tracked, and the first six fingerprints generated from the newly discovered device are automatically added to the database, eliminating the classification process as long as they are received during the first five minutes of the device being discovered and correctly classified.

$$\begin{aligned} \text{If } t_i \in [t_0, t_0 + \Delta t] \text{ and } \mathbf{f}_i \notin \mathcal{D}(M_s), \text{ then } \mathcal{D}(M_s) \\ \leftarrow \mathcal{D}(M_s) \cup \{\mathbf{f}_i\}, \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{27}$$

We chose six fingerprints as most devices in our environment show six fingerprints or fewer in the database. Figure 18 presents the results of adding a second fingerprint following this process and bypassing device classification. Additionally, two log files are maintained that store fingerprints generated during unsuccessful fingerprint retrievals and device classifications.

4.3. Flow Table Size

Flow table overflow is always a pressing concern in any SDN architecture. TIMeFoRCE is discriminatory and restrictive, where each flow is required to match multiple criteria. For example, a flow that matches on source MAC address would only need two flows for all destinations. A flow that matches the source and destination MAC addresses will need two flows for each MAC address combination. Extrapolate that to the source and destination MAC addresses, the source and destination IP addresses, and the source and destination ports, and the exponential growth in the flow table size becomes apparent. An SDN switch flow table grows exponentially as more fields are added to the match criteria, leading to more flow table entries.
We derive a formula to illustrate this growth.

Let:

- N_s be the number of source MAC addresses,
- N_d be the number of destination MAC addresses,
- N_{IP_s} be the number of source IP addresses,
- N_{IP_d} be the number of destination IP addresses,
- N_{port_s} be the number of source ports,
- N_{port_d} be the number of destination ports.

If a flow matches only the source MAC address, then a single flow rule can handle all destinations:

$$F = N_s \tag{28}$$

When matching on both source and destination MAC addresses, the required flow table size increases to:

$$F = N_s \times N_d \tag{29}$$

When matching on both MAC and IP addresses, the number of flows further increases:

$$F = N_s \times N_d \times N_{IP_s} \times N_{IP_d} \tag{30}$$

If the flow table matches not only MAC and IP addresses but also source and destination ports, the total number of required flows is:

$$F = N_s \times N_d \times N_{IP_s} \times N_{IP_d} \times N_{port_s} \times N_{port_d} \tag{31}$$

The number of required flow entries grows exponentially with the number of fields included in the match criteria. This presents scalability challenges in SDN, particularly in large-scale deployments.

As a result, timeout values play a critical role in regulating flow table growth by helping to maintain a manageable table size. Open vSwitch (OVS) in its default state supports approximately 200,000 flows [58]. However, this can be adjusted and is highly dependent on the hardware used. More resources equal more capabilities. We performed flow table lookup for six idle and hard timeout combinations: 1 and 2 minutes, 2 and 4 minutes, 3 and 6 minutes, 4 and 8 minutes, 5 and 10 minutes, and 10 and 20 minutes, respectively. Figure 19 provides the metrics for the various timeout values. The higher the timeout threshold, the greater the accumulation of flows. If λ is the average flow arrival rate, and T be the timeout duration applied to each flow (in seconds). The expected number of active flow entries in the PEP flow table, denoted by N, is given by:

$$N = \lambda \cdot T \tag{32}$$

This indicates a linear relationship: As the timeout T increases, the size of the flow table N increases proportionally.

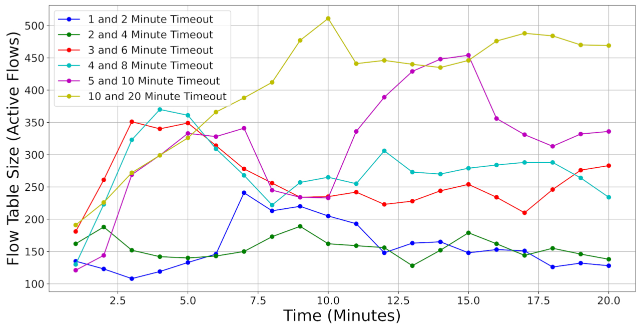


Figure 19: Size of the SDN switch flow table at 1-minute time intervals over a 20-minute period.

4.4. Packets Matched

In addition to the flow table size for idle and hard timeouts, we also examine the matched packets at each 60-second interval. Network traffic is highly unpredictable. As described in Table 5 of Section 4.1, devices can generate frequent communication requests even in standby mode or when not in use. To compensate for variability in packet transmission, we collected five sets of metrics for each idle/hard timeout combination and averaged the results. $M_k^{(j)}$ represent the number of packets matched in minute $k \in \{1, \ldots, 20\}$ for period $j \in \{1, \ldots, 5\}$. The average number of matched packets in minute k, denoted $\bar{M}_k$, is calculated as:

$$\bar{M}_k = \frac{1}{5} \sum_{j=1}^{5} M_k^{(j)} \tag{33}$$

Figure 20 presents the packets matched every 60 seconds. Irrespective of the timeout values set, packets sent across the network should not change much, as devices are abstracted away from the timeout values. Based on the observed results, the packets that matched appeared relatively uniform across the various idle/timeout combinations. The most noticeable distinction lies in the flow table size of the policy enforcement point (PEP). This is expected behavior, as flows are removed after the idle/hard timeout, limiting the size of the flow table.

$$\text{Remove}(\phi_i) = \begin{cases} \text{True} & \text{if } t - t_{\text{last}}^{(i)} \geq \tau_{\text{idle}} \\ \text{True} & \text{if } t - t_{\text{created}}^{(i)} \geq \tau_{\text{hard}} \\ \text{False} & \text{otherwise} \end{cases} \tag{34}$$

This led us to another question. Do the idle/hard timeout values significantly impact the number of authentication requests sent to the policy decision point (PDP)?

4.5. Result of Authentication Request

An authentication request is sent to the PDP when there are no matching policies; in this case, no flow rule exists at the PEP, creating a ”miss” condition. Assuming entries are removed based on the minimum of the idle and hard timeout, we can model:

$$P_{\text{miss}} \propto \frac{1}{\min(T_{\text{idle}}, T_{\text{hard}})} \tag{35}$$

The packet-in rate is approximately:

$$\text{Packet-In Rate} = \lambda \cdot P_{\text{miss}} \approx \frac{\lambda}{\min(T_{\text{idle}}, T_{\text{hard}})} \tag{36}$$

As mentioned in the previous section, there is a direct correlation between flow table size and authentication requests, especially for chatty devices. Lower timeout values lead to more frequent flow expirations, increasing the number of packets sent to the PDP as misses and requiring authentication via TIMeFoRCE. We utilized the putty logging mechanism to capture log files from the policy engine (PE), in this case, FWS, for the 20-minute period. We then filter the logs to obtain the number of authentication requests sent for the 20-minute period. Figure 21 provides the number of authentication requests sent for each idle/hard timeout combination. Based on the graph, the 1-2 minute idle/hard timeout has the highest number of authentication requests, while the 10-20 minute idle/hard timeout has the lowest. This is expected behavior, as the flows will persist for a longer period at the PEP before a device needs to re-authenticate.

4.6. Authentication Success Rate

Any effective authentication solution that provides identity and access management must perform accurately. This means it correctly identifies and authenticates legitimate subjects and rejects illegitimate ones. Authentication succeeds if either the fingerprint matches the database or the device's behavior is successfully classified (for new devices).
Let:

- M: MAC address of the device
- $\mathbf{x} \in \{0, 1\}^{16}$: binary feature vector extracted from packet headers
- $\mathcal{F}$: fingerprint database containing known $(M, \mathbf{f})$ pairs
- $F \in \{0, 1\}$: fingerprint match indicator

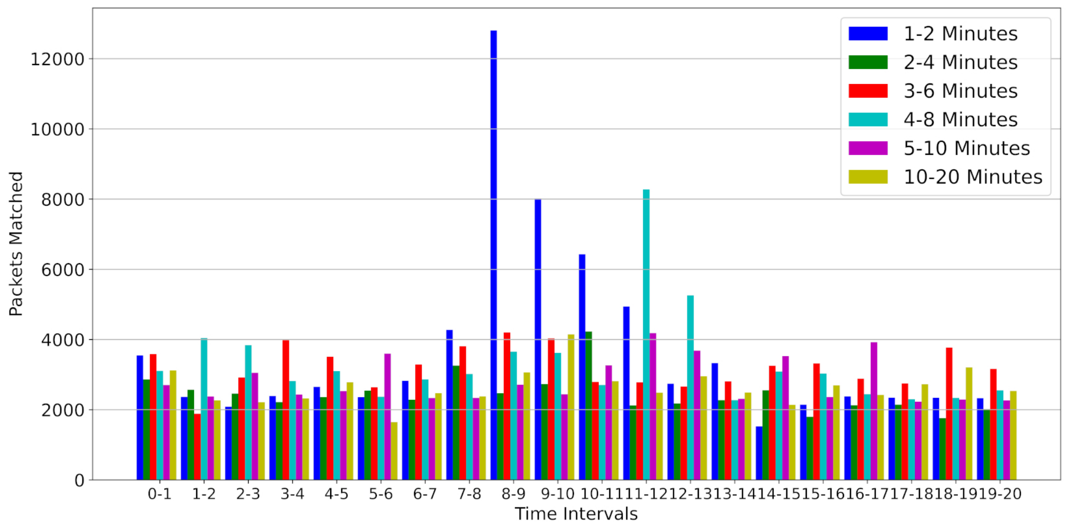


Figure 20: Packets matched every 60 seconds over 20 minutes

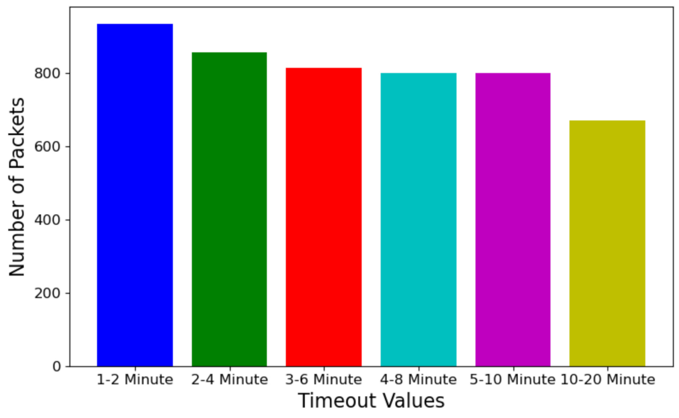


Figure 21: Authentication request sent to the PDP PE for each idle/hard timeout value.

- $\hat{p}_\gamma \in [0, 1]$: predicted classification confidence
- $\theta = 0.75$: minimum confidence threshold
- $\gamma \in \{0, 1\}$: classification acceptance outcome
- $A \in \{0, 1\}$: overall authentication decision

Define the fingerprint match:

$$F = \begin{cases} 1, & \text{if } (M, \mathbf{f}) \in \mathcal{F} \\ 0, & \text{otherwise} \end{cases} \tag{37}$$

Define classification acceptance based on confidence threshold:

$$\gamma = \begin{cases} 1, & \text{if } \hat{p}_\gamma \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{38}$$

Authentication is successful if either fingerprint or classification is accepted:

$$A = F \vee \gamma \tag{39}$$

Assuming independence:

$$P(A = 1) = 1 - (1 - p_F)(1 - p_\gamma) \tag{40}$$

To demonstrate the effectiveness of our solution for IoT device authentication in a ZTA, we present metrics for 1,000 access requests. We sent 1000 access requests to the PDP (TIMeFoRCE) and measured the percentage of successful authentication requests. Figure 22 shows that of the 1000 requests sent to the PDP, a successful authentication was returned for 961 of the requests, which is a 96.1% success rate. A successful authentication is one in which a fingerprint is retrieved from the database, and the PDP PA updates the PEP with bidirectional flows. Thirty-nine of the requests were unsuccessful, which is 3.9%. It is, however, important to note that the 39 unsuccessful requests are from only four devices, each sending the same repeated authentication request. If we treat those as only four requests, the resulting authentication accuracy is over 99%. Of the four (4) devices, two (2), representing 31 unsuccessful authentications, were not authorized to communicate based on the observed traffic match criteria and were successfully denied. A denial results in no flow being added to the PEP's flow table, enforcing Zero Trust. This occurred because of the removal of specific fingerprints from the database. Two (2) devices, representing eight requests, did not successfully authenticate, resulting in denied traffic. This is an incorrect assessment, as those represent legitimate device communications, which can be attributed to human error, as those fingerprints were not added to the database.

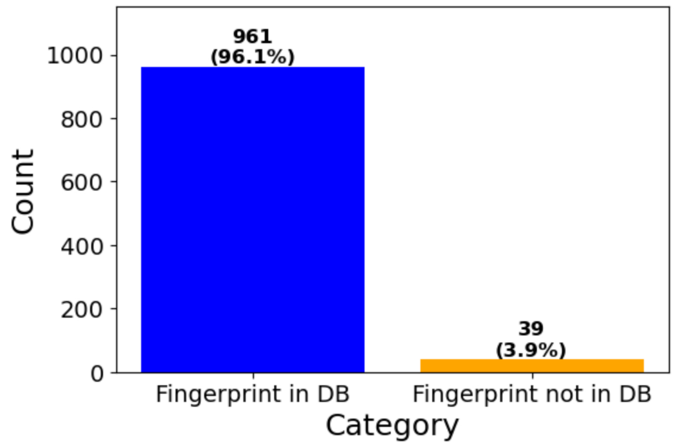


Figure 22: Authentication request sent to the PDP PE, resulting in successful authentication.

4.7. Introduced Latency For First Packets

In any network architecture, latency is key and can significantly impact performance. Latency was observed while using our Packet-Forwarder application to implement authentication. Figure 23 illustrates the additional latency introduced, including a trend line showing a decrease in the time it takes to perform the overall operation. This latency is incurred exclusively by the initial packet, as subsequent packets are not redirected to the PDP until the designated timeout has elapsed. T_n, is the delay between the PEP and PDP, T_p, PDP processing time, and T_f, is the delay to install the flow rule back to the PEP.

$$L_{\text{fwd}} = 2T_n + T_p + \max_{1\le i\le N} T_f^{(i)} \tag{41}$$

Since the PDP installs a reverse flow along with the forward flow, the reverse packet does not trigger a miss requiring authentication. Thus the reverse flow incurs no latency for authentication beyond L_{normal}, the normal network latency:

$$L_{\text{rev}} = L_{\text{normal}} \tag{42}$$

Thus the total setup latency for the bi-directional flow is:

$$L_{\text{total}} = L_{\text{fwd}} + L_{\text{rev}} = 2T_n + T_p + \max_{1\le i\le N} T_f^{(i)} + L_{\text{normal}} \tag{43}$$

It is important to note that this latency is in addition to the normal network operational latency. We elected to present the additional latency rather than the overall latency for two primary reasons: (1) each network environment is unique, and the latency observed in one setting may not accurately represent that in another, and (2) the additional latency specifically characterizes the delay introduced by the first transmitted packet, rather than applying uniformly to every packet sent by a device.

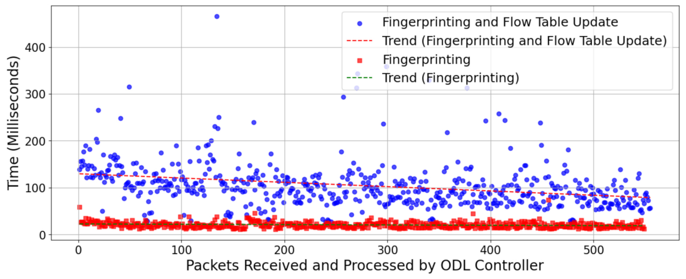


Figure 23: The additional latency incurred to perform Zero ZTA authentication measured for the first packet in each communication request, with results illustrated alongside a corresponding trend line.

4.8. Network Throughput

Network throughput is an important consideration when implementing any architecture. We present the throughput achieved with an SDN switch using static flow rules and our Packet-Forwarder solution. We obtained performance metrics using the iperf3 [59] tool and conducted the test over 60 seconds at 1-second intervals. We used a Raspberry Pi 4B with 4GB of RAM running Kali Linux 2024.4 as the iperf3 client, and a UP2 development board with 8GB of RAM running Ubuntu 22.04 as the iperf3 server. The results presented in Figure 24 show that using our Packet-Forwarder Zero Trust application, we obtained a throughput of approximately 895 Mbits/sec compared to 920 Mbits/sec using a default configuration SDN switch on a 1000 Mbits/sec interface.

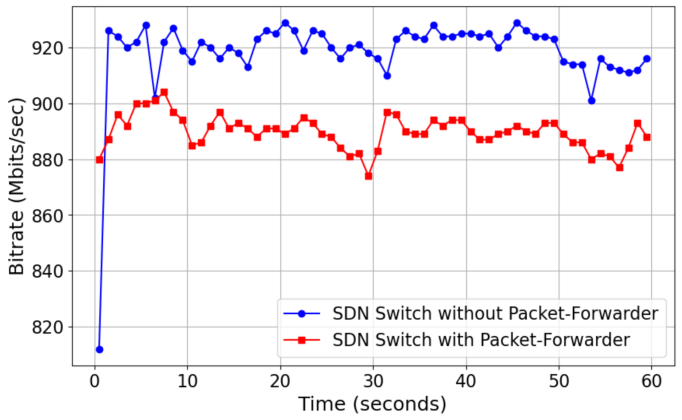


Figure 24: Network throughput showing the throughput for TiMEFoRCE and the SDN switch operating in default mode

4.9. Adversarial Model

In any identity and access management solution, several threats are ever-present. As it relates to our TIMeFoRCE, common threats include identity spoofing, evasion, replay, misclassifications or false positives, man-in-the-middle (MitM), and denial of identity. The identity management system TIMeFoRCE is designed to operate in a Zero Trust environment where devices are authenticated via behavioral fingerprinting and SDN-based flow enforcement. This section describes the adversarial assumptions, capabilities, objectives, and attack strategies targeting the system's core components. The adversary seeks to:

- Bypass identity verification by mimicking legitimate devices.
- Evade classification and enforcement by exploiting ambiguous or unclassifiable behavior.
- Trigger classification errors (false negatives) to obtain unauthorized access.
- Interfere with SDN flow rule installation or behavior.
- Overload the system with excessive identity attempts or fingerprint variations.

Given the following capabilities, the adversary may possess one or more of the following:

- Passive packet sniffing to observe MAC addresses, ports, and QNAMEs.
- Active injection of spoofed packets and manipulated fingerprints.
- Replay of legitimate traffic traces and behavioral fingerprints.
- Partial knowledge of the ML model's feature set and threshold logic.
- Ability to manipulate or observe SDN flow control messages (e.g., man-in-the-middle).
- Capability to launch resource exhaustion attacks using high volumes of malformed or ambiguous identity traffic.

The primary concern in IoT device security as it relates to TIMeFoRCE is identity spoofing, where an illegitimate device impersonates a legitimate one. We consider adversaries with varying levels of access and capabilities who seek to spoof the IoT fingerprinting system. Let the fingerprint for device i be defined as:

$$f_i = \text{MAC}_i \parallel \vec{x}_i, \quad \vec{x}_i \in \{0,1\}^{16} \tag{44}$$

where MAC_i is the device's 48-bit MAC address and $\vec{x}_i$ is the 16-bit binary feature vector extracted from packet headers.

To address this concern, we combined the device's MAC address with 16 binary features from the packet header. The 16 binary features are used to mitigate the possibility of a cloned MAC Address. A malicious device would need to clone the MAC address of an existing device and generate comparable feature variables. The probability that the attacker successfully spoofs a known MAC address (P_{mac}), and the probability that the attacker successfully replicates the correct 16-bit binary feature vector (P_{feat}), resulting in the probability that the attacker spoofs the entire fingerprint $(M^*, \mathbf{f}^*)$ ($P_{\text{fingerprint}}$) is defined by:

$$P_{\text{fingerprint}} = P_{\text{mac}} \cdot P_{\text{feat}} \tag{45}$$

If each bit in the feature vector is guessed randomly:

$$P_{\text{feat}} = \frac{1}{2^{16}} = 1.5259 \times 10^{-5} \tag{46}$$

Then, if $P_{\text{mac}} = 0.9$ (A very high probability of being spoofed):

$$P_{\text{fingerprint}} = 0.9 \cdot \frac{1}{2^{16}} \approx 1.3733 \times 10^{-5} \tag{47}$$

resulting in extremely low probability. The addition of the timeout values ensures the device is re-authenticated for each subsequent communication request after the timeout expires. This will aid in determining if a device is potentially compromised, as its behavior will change. This will also guard against replay attacks.

Additionally, we make the following system assumptions:

- The adversary does not possess direct access to the fingerprint database.
- The classifier operates on 16 binary features extracted from packet headers.
- The classification threshold is fixed at $\theta = 0.75$.
- The SDN control plane is assumed to be protected, but may be targeted via MitM attacks.

We developed an identity management solution that can satisfy the tenets of Zero Trust, and while we present an adversarial model for one of the most common threats, spoofing of identities, other threats include:
The crafting feature vector $\mathbf{x}'$ such that:

$$\mathcal{M}(\mathbf{x}') \in \text{Other} \quad \text{and} \quad \text{predict\_proba}(\mathcal{M}, \mathbf{x}') < \tau \tag{48}$$

or applying perturbation δ to feature vector:

$$\mathbf{x}' = \mathbf{x} + \delta \quad \text{where} \quad ||\delta|| < \epsilon \quad \text{and} \quad \mathcal{M}(\mathbf{x}') \neq \mathcal{M}(\mathbf{x}) \tag{49}$$

These types of attacks are designed to weaken the model performance to create false negatives, where legitimate devices are classified as 'Other'. If the device being introduced into the system is not one that is approved, the classification of 'Other' is appropriate, as classification only occurs during the addition of new devices. The administrator is notified whenever a new device is given the 'Other' designation, prompting manual actions. Additionally, an adversary may seek to intercept or modify SDN control messages $m \in \mathcal{F}$ to create:

$$m' = f(m) \quad \text{where} \quad f \text{ alters flow behavior} \tag{50}$$

The controller operates on an out-of-band (OOB) channel that is separate from the data path. The attacker may also generate sequence $\{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$ with:

$$\forall i, \mathbf{f}_i \notin \mathcal{D} \quad \text{and} \quad k \gg |\mathcal{D}| \tag{51}$$

We rate-limit new fingerprint additions by enforcing threshold-based rejection for unknown devices. If a device sends 10 identical packets that fail to generate a successful fingerprint, a flow is written to block that device from communicating for the specified communication type.

5. Conclusion

Security is a critical network element and the primary driver of a ZTA. Current ZTA solutions remain insufficient in delivering comprehensive Identity and Access Management for IoT devices. We demonstrated that authentication for IoT devices can be provided for a ZTA using device behavioral traits. TIMeFoRCE observed an overall success rate of 96.1% across all traffic and a 99% success rate for all devices that attempted authentication after removing duplicate packets resulting from retransmissions. Additionally, we achieved classification accuracy above 99% using the dictionary, even on validation data not previously encountered. The results are derived from a small testbed; however, we contend that, with sufficient resources and processing power, these findings can be extrapolated to accommodate significantly larger commercial network environments. Organizations can mitigate the blast radius of compromised or misbehaving devices by authenticating IoT devices and constraining their communication. The Packet-Forwarder application is modular, meaning the back-end classification algorithm used to identify devices can be updated as technology evolves. This research focuses solely on providing a solution to address identity management and access control. Further research is needed to address the other areas not discussed in this paper. Although the solution demonstrates strong performance, there remains a potential risk that traffic from a legitimate device may be denied if a corresponding fingerprint is not identified during the database query. To account for this, TIMeFoRCE maintains log files that store the fingerprints generated during unsuccessful retrievals and device classifications. A record of the device fingerprint and the number of times it was observed is maintained. These log files allow a network programmer to scrutinize devices add fingerprints manually to the database.

6. Future Work

TIMeFoRCE is implemented with a single SDN controller. The flows are written to the SDN switch flow table with hard-timeout values, causing the flow to expire and be removed. During implementation and testing, it was discovered that whenever the SDN switch (the PEP) loses access to the SDN controller (the PDP), the flows remain active on the SDN switch until the SDN controller is reconnected. Existing connections will remain active; however, new connections will not be possible. Exploration into a distributed SDN controller architecture or a fail-safe that installs a default flow in the SDN switch flow table until controller functionality returns is necessary. Machine learning models, large language models (LLMs), and generative AI, such as ChatGPT [60], are constantly evolving and have become integral to many security solutions. It is important to explore their capability to provide improved identification and classification. To examine the effectiveness of the proposed solution, IoT devices deployed in environments such as industrial, medical, and smart cities should be investigated. TIMeFoRCE used a probability threshold of 0.75, which is significantly higher than the standard random forest threshold of 0.50. While correct classification was achieved with a higher threshold, this could lead to legitimate devices being incorrectly classified as "Other," denying authentication and the successful addition of fingerprints to the database. Finally, the fingerprints are stored in plain text in the database; a more secure approach is to hash the values during generation.

Conflict of Interest The authors declare no conflict of interest.

Acknowledgment The authors would like to thank the Cybersecurity Assurance Policy (CAP) Center for providing the resources necessary to perform this research.

References

[1] V. Morris, K. Kornegay, "Flow Table Modification Using Behavioral-based Fingerprinting Technique to Facilitate Zero Trust Identity Management and Access Control," in 2025 16th Annual IEEE Computing and Communication Workshop and Conference (CCWC), 325–332, IEEE, 2025, doi:DOI: 10.1109/CCWC62904.2025.10903861.

[2] B. Bezawada, I. Ray, I. Ray, "Behavioral fingerprinting of Internet-of-Things devices," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, **11**(1), 41–50, 2021, doi:DOI: 10.1002/widm.1337.

[3] H. Noguchi, M. Kataoka, Y. Yamato, "Device identification based on communication analysis for the internet of things," IEEE Access, **7**, 52903–52912, 2019, doi:DOI: 10.1109/ACCESS.2019.2910848.

[4] R. Perdisci, T. Papastergiou, O. Alrawi, M. Antonakakis, "IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis," Available at `www.google.com`, accessed: [Please insert the access date].

[5] C. Buck, C. Olenberger, A. Schweizer, F. Völter, T. Eymann, "Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust," Computers and Security, **110**, 102436, 2021, doi:DOI: 10.1016/j.cose.2021.102436.

[6] O. C. Edo, T. Tenebe, E. Etu, A. Ayuwu, J. Emakhu, S. Adebiyi, "Zero Trust Architecture: Trend and Impact on Information Security," International Journal of Emerging Technology and Advanced Engineering, **12**(7), 140–147, 2022, doi:DOI: 10.46338/ijetae0722_15.

[7] Y. G. Wu, W. H. Yan, J. Z. Wang, "Real identity based access control technology under zero trust architecture," Proceedings - 2021 International Conference on Wireless Communications and Smart Grid, ICWCSG 2021, 18–22, 2021, doi:DOI: 10.1109/ICWCSG53609.2021.00011.

[8] S. Rose, O. Borchert, S. Mitchell, S. Connelly, "Zero Trust Architecture," NIST Special Publication 800-207, National Institute of Standards and Technology, 2020, doi:DOI: 10.6028/NIST.SP.800-207.

[9] N. F. Syed, S. W. Shah, A. Shaghaghi, A. Anwar, Z. Baig, R. Doss, "Zero Trust Architecture (ZTA): A Comprehensive Survey," IEEE Access, **10**, 57143–57179, 2022, doi:DOI: 10.1109/ACCESS.2022.3174679.

[10] U. Mattsson, "Zero Trust Architecture," Controlling Privacy and the Use of Data Assets, 127–134, 2022, doi:DOI: 10.1201/9781003189664-11.

[11] P. Assunção, "A Zero Trust Approach to Network Security," in Proceedings of the Digital Privacy and Security Conference 2019, 2019.

[12] J. Kindervag, "No More Chewy Centers: Introducing the Zero Trust Model of Information Security," Technical report, Forrester Research, 2010, accessed: January 24, 2024.

[13] M. Campbell, "Beyond Zero Trust: Trust Is a Vulnerability," Computer, **53**(10), 110–113, 2020, doi:DOI: 10.1109/MC.2020.3011081.

[14] Defense Information Systems, Agency, National Security Agency : Zero Trust Engineering Team, "Department of Defense (DOD) Zero Trust Reference Architecture CLEARED For Open Publication Department of Defense OFFICE OF PREPUBLICATION AND SECURITY REVIEW," 170, 2021.

[15] Cisco Systems, "Cisco Identity Services Engine (ISE)," `https://www.cisco.com/site/us/en/products/security/identity-services-engine/index.html`, 2024, accessed: 2025-06-13.

[16] Microsoft, "Introduction to Azure IoT," https://learn.microsoft.com/en-us/azure/iot/iot-introduction, accessed: January 21, 2025.

[17] Palo Alto Networks, "IoT Security for Zero Trust Enterprise," https://www.paloaltonetworks.com/network-security/iot-security, 2024, accessed: 2025-06-13.

[18] Armis Inc., "Armis Platform Overview," https://www.armis.com/platform/, 2024, accessed: 2025-06-13.

[19] Zscaler Inc., "Zscaler for IoT and OT Security," https://www.zscaler.com/solutions/iot-security, 2024, accessed: 2025-06-13.

[20] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. R. Sadeghi, S. Tarkoma, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," Proceedings - International Conference on Distributed Computing Systems, 2177–2184, 2017, doi:DOI: 10.1109/ICDCS.2017.283.

[21] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, "IoTSense: Behavioral Fingerprinting of IoT Devices," arXiv:, 2018.

[22] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, V. Sivaraman, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," IEEE Transactions on Mobile Computing, **18**(8), 1745–1759, 2019, doi:DOI: 10.1109/TMC.2018.2866249.

[23] R. R. Chowdhury, S. Aneja, N. Aneja, E. Abas, "Network Traffic Analysis based IoT Device Identification," ACM International Conference Proceeding Series, 79–89, 2020, doi:DOI: 10.1145/3421537.3421545.

[24] S. Aneja, N. Aneja, M. S. Islam, "IoT Device Fingerprint Using Deep Learning," in Proceedings - 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS), 174–179, 2019, doi:DOI: 10.1109/IOTAIS.2018.8600824.

[25] P. Oser, F. Kargl, S. Lüders, "Identifying devices of the internet of things using machine learning on clock characteristics," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), **11342 LNCS**, 417–427, 2018, doi:DOI: 10.1007/978-3-030-05345-1_36.

[26] A. Hameed, "IoT Traffic Multi-Classification Using Network and Statistical Features in a Smart Environment; IoT Traffic Multi-Classification Using Network and Statistical Features in a Smart Environment," Technical report, 2020.

[27] S. V. Radhakrishnan, A. S. Uluagac, R. Beyah, "GTID: A Technique for Physical Device and Device Type Fingerprinting," IEEE Transactions on Dependable and Secure Computing, forthcoming or In Press; update fields once details are known.

[28] K. Kostas, M. Just, M. A. Lones, "IoTDevID: A Behavior-Based Device Identification Method for the IoT," 2021, doi:DOI: 10.1109/JIOT.2022.3191951.

[29] N. Basta, M. Ikram, M. Kaafar, A. Walker, "Towards a Zero-Trust Micro-segmentation Network Security Strategy: An Evaluation Framework," 1–7, 2022, doi:DOI: 10.1109/noms54207.2022.9789888.

[30] D. Comer, A. Rastegarnia, "Externalization of Packet Processing in Software Defined Networking," IEEE Networking Letters, **1**(3), 124–127, 2019, doi:DOI: 10.1109/lnet.2019.2918155.

[31] C. Liu, R. Tan, Y. Wu, Y. Feng, F. Z. Z. Jin, Y. Liu, Q. Liu, "Dissecting zero trust: research landscape and its implementation in IoT," Cybersecurity, **7**, 20, 2024, doi:10.1186/s42400-024-00212-0.

[32] C. Bast, K. Yeh, "Emerging Authentication Technologies for Zero Trust on the Internet of Things," Symmetry, **16**(8), 993, 2024, doi:10.3390/sym16080993.

[33] H. Kang, G. Liu, Q. Wang, L. Meng, Lei, , J. Liu, "Theory and Application of Zero Trust Security: A Brief Survey," Entropy, **25**(12), 1–26, 2023, doi:10.3390/e25121595.

[34] D. Eidle, S. Y. Ni, C. Decusatis, A. Sager, "Autonomic Security for Zero Trust Networks," in 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 288–293, 2017, doi:DOI: 10.1109/UEMCON.2017.8249053.

[35] S. Ameer, L. Praharaj, R. Sandhu, S. Bhatt, M. Gupta, "ZTA-IoT: A Novel Architecture for Zero-Trust in IoT Systems and an Ensuing Usage Control Model," ACM Transactions on Privacy and Security, **27**(3), 2024, doi:10.1145/3671147.

[36] R. Chandramouli, Z. Butcher, "NIST Special Publication NIST SP 800-207A ipd 2 A Zero Trust Architecture Model 3 for Access Control in Cloud-Native Applications in Multi-Location," .

[37] Amazon Web Services (AWS), "AWS IoT Core," https://aws.amazon.com/iot-core/, accessed: 21 January 2025.

[38] Siemens, "Industrial Internet of Things (IIoT)," https://www.sw.siemens.com/en-US/solutions/industrial-internet-of-things-iiot/#6Xl1D3sPnBHuo88QKMZ1EV, accessed: January 21, 2025.

[39] "IoT Cloud Platform Market," https://www.marketsandmarkets.com/Market-Reports/iot-cloud-platform-market-195182.html, accessed: 2021-03-20.

[40] U. Guide, "WRT 3200ACM Gigabit Wi-Fi Router," .

[41] A. Lunn, V. Didelot, F. Fainelli, "Distributed Switch Architecture, A.K.A DSA," Netdev 2.1, 2017.

[42] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, "Behavioral fingerprinting of IoT devices," in Proceedings of the ACM Conference on Computer and Communications Security, 41–50, Association for Computing Machinery, 2018, doi:DOI: 10.1145/3266444.3266452.

[43] M. R. Shahid, G. Blanc, Z. Zhang, H. Debar, "IoT Devices Recognition Through Network Traffic Analysis," in Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018, 5187–5192, Institute of Electrical and Electronics Engineers Inc., 2019, doi:DOI: 10.1109/BigData.2018.8622243.

[44] Y. Guo, Y. Wang, F. Khan, A. A. Al-Atawi, A. A. Abdulwahid, Y. Lee, B. Marapelli, "MAB with SDN Orchestration," 1–18, 2023.

[45] L. Fan, S. Zhang, Y. Wu, Z. Wang, C. Duan, J. Li, J. Yang, "An IoT Device Identification Method based on Semi-supervised Learning," in 2020 16th International Conference on Network and Service Management (CNSM), 1–7, IEEE, 2020, doi:DOI: 10.23919/CNSM50824.2020.9269044.

[46] "OpenDaylight project," https://en.wikipedia.org/wiki/OpenDayligh_Project., accessed: Nov. 2, 2024.

[47] "Flask Documentation (v3.0.x)," https://flask.palletsprojects.com/en/3.0.x/, accessed: January 24, 2024.

[48] "Python," https://www.python.org/, accessed: January 24, 2024.

[49] "MySQL," https://www.mysql.com/, accessed: January 24, 2024.

[50] "Apache Maven," https://maven.apache.org/, accessed: January 24, 2024.

[51] "MVNRepository," https://mvnrepository.com/, accessed: January 24, 2024.

[52] P. K. Sharma, J. H. Park, Y. S. Jeong, J. H. Park, "SHSec: SDN-based Secure Smart Home Network Architecture for the Internet of Things," Mobile Networks and Applications, **24**(3), 913–924, 2019, doi:DOI: 10.1007/s11036-018-1147-3.

[53] P. Kumar, A. Gurtov, J. Iinatti, M. Ylianttila, M. Sain, "Lightweight and Secure Session-Key Establishment Scheme in Smart Home Environments," IEEE Sensors Journal, **16**(1), 254–264, 2016.

[54] P. Krishnan, K. Jain, K. Achuthan, R. Buyya, "Software-Defined Security-by-Contract for Blockchain-enabled MUD-aware Industrial IoT Edge Networks," IEEE Transactions on Industrial Informatics, 2021, doi:DOI: 10.1109/TII.2021.3084341.

[55] S. Bera, S. Misra, A. Jamalipour, "FlowStat: Adaptive Flow-Rule Placement for Per-Flow Statistics in SDN," IEEE Journal on Selected Areas in Communications, **37**(3), 530–539, 2019, doi:DOI: 10.1109/JSAC.2019.2894239.

[56] O. Corporation, “VirtualBox 7.0.14 for Windows,” https://www.virtualbox.org/wiki/Download_Old_Builds_7_0, accessed: March 16, 2024.

[57] OpenDaylight, “Karaf Integration 0.20.3,” https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/karaf/0.20.1, accessed: March 16, 2024.

[58] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. G. J. Rajahalme, A. Wang, J. Stringer, P. Shelar, K. Amidon, M. Casado, “The Design and Implementation of Open vSwitch,” Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2015, 117–130, 2015.

[59] T. iPerf Group, “iPerf,” https://iperf.fr/, 2025, accessed: February 6, 2025.

[60] OpenAI, “ChatGPT,” https://chat.openai.com/chat, 2023, accessed: April 17, 2024.

Copyright: This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-SA) license (https://creativecommons.org/licenses/by-sa/4.0/).