MUD-Based Behavioral Profiling Security Framework for Software-Defined IoT Networks

Prabhakar Krishnan, *Member, IEEE*, Kurunandan Jain, Rajkumar Buyya, *Fellow, IEEE*, Pandi Vijayakumar, *Senior Member, IEEE*, Anand Nayyar, *Senior Member, IEEE*, Muhammad Bilal, *Senior Member, IEEE*, and Houbing Song, *Senior Member, IEEE*

***Abstract*—The rapid development and deployment of Internet of Things (IoT) devices in modern networks and Industry 4.0 have attracted substantial interest from cybersecurity researchers. In this study, we propose a software-defined framework that improves network intrusion detection systems by using manufacturer usage description (MUD) to enhance the behavioral monitoring in IoT networks. We aim to explore whether Industrial IoT (IIoT) devices typically serve a common role in cyber–physical systems, and their communications exhibit predictable patterns that can be defined in MUD profile(s) formally and succinctly. We design a framework that utilizes the concept of digital twins and software-defined networking to improve the security of IIoT environments. The MUD data are profiled, and the actions are evaluated on the network digital twin before they are used in the physical network. The behavioral profiling system is updated in real time, thereby improving the overall system security and compliance to policies in the IoT deployment. Evaluation results show that our solution outperforms existing approaches substantially in terms of attack detection accuracy, predicting security incidents, response time, and resource usage.**

***Index Terms*—Digital twin, manufacturer usage description (MUD), network security, software-defined networking (SDN).**

I. Introduction

With the proliferation of embedded and ubiquitous cyber–physical systems (CPSs) and applications, the case for fusing the physical and virtual worlds has become increasingly compelling. The rapid growth and proliferation of Internet of Things (IoT) in modern network environments, cloud-to-fog applications, smart cities, and Industry 4.0 have motivated substantial research interest in understanding applications, security, protection, risks, vulnerabilities, and ecosystems [1].

Recently, network device behavioral profiling, fingerprinting, and pattern recognition methods have attracted much attention from research communities in systems, networking, and defense. Accordingly, 5G (and beyond) is predicted to enhance the volume, real-time velocity, and diversity of data in basic CPSs. Investigating the viability of combining software-defined networking (SDN), intent-based network (IBN), and network function virtualization (NFV) with CPS deployments is necessary. Network programmability helps service and application delivery with higher agility and cost-effectiveness with better provisioning for network resources. The proliferation of IoT-based intelligent devices is projected at 50 billion by this decade and poses many problems and difficult issues concerning the safety and privacy of devices, users, and data consumed by applications [3]. SDN paradigm is increasingly used in enterprise networks to orchestrate and manage cyber-security [4]. Tampering with data in an IoT network can have dire consequences. For example, the largest Distributed Denial-of-Service (DDoS) assault in terms of volume was conducted because of malicious traffic streams generated by IoT devices infected with [5] the Mirai botnet virus. Behavioral fingerprinting [6] research and solutions offer encouraging results for IoT vendors to develop fingerprint profiles for anomaly detection. This trend accelerates the adoption of standard manufacturing practices in designing the hardware and corresponding compliant software, such as manufacturer usage description (MUD) standards and establishing concrete guidelines regarding device behavior and operations in the field. Cisco's MUD [7] is standardized in the device bootstrapping protocols, such as dynamic host configuration protocol (DHCP), link-layer discovery protocol (LLDP), and 802.x protocols. The researchers and manufacturer's community are developing guidelines detailing practical steps in implementing MUD in various use cases for IoT compliance and security. The open-source community has contributed to an implementation called *osMUD* [8]. The Internet Engineering Task Force (IETF) has defined an architecture and request for comment draft (RFC) for

Manuscript received January 12, 2021; revised June 30, 2021 and August 10, 2021; accepted September 4, 2021. Date of publication September 17, 2021; date of current version April 25, 2022. *(Corresponding author: Pandi Vijayakumar.)*

Prabhakar Krishnan and Kurunandan Jain are with the Center for Cybersecurity Systems and Networks, Amrita Vishwa Vidyapeetham, Amritapuri 690525, India (e-mail: kprabhakar@am.amrita.edu; kurunandanj@am.amrita.edu).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Pandi Vijayakumar is with the Department of Computer Science and Engineering, University College of Engineering Tindivanam, Tindivanam 604001, India (e-mail: vijibond2000@gmail.com; pvijayakumar@ieee.org).

Anand Nayyar is with the Graduate School, Faculty of Information Technology, Duy Tan University, Da Nang 550000, Vietnam (e-mail: anandnayyar@duytan.edu.vn).

Muhammad Bilal is with the Department of Computer Engineering, Hankuk University of Foreign Studies, Yongin 17035, Gyeonggi, South Korea (e-mail: m.bilal@ieee.org).

Houbing Song is with the Security and Optimization for Networked Globe Laboratory, Department of Electrical Engineering and Computer Science, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114 USA (e-mail: h.song@ieee.org).

Digital Object Identifier 10.1109/JIOT.2021.3113577

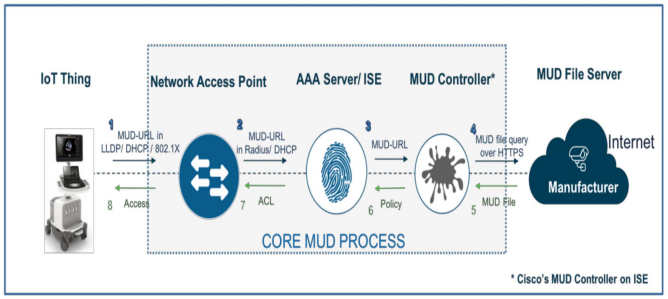


Fig. 1. MUD protocol workflow [7].

implementing MUD policies and for deploying the files [7]. The IETF consortium of IoT device manufacturers recommends the "Yet Another Next Generation (YANG)" policy model and "JavaScript Object Notation (JSON)" format for defining "Access Control Lists (ACLs)" and policies. The MUD adoption is still in the nascent stage and no consensus is reported among the manufacturers on profile formats, the granularity of the policy definitions, behavior specification, interoperability, compliance in heterogeneous vendor environments, and the enforcement scheme at run time.

However, "things" everywhere has opened new doors of vulnerabilities due to its ubiquitous connectivity and pervasive volume, and the attack surface has increased in modern networks. In Industrial IoT (IIoT) deployments, gateway switches are the crucial entry points into the network and are usually air-gapped through wireless connectivity. Considerable research in academia and commercial solutions has been conducted for securing the IIoT gateway. The MUD workflow is illustrated in Fig. 1. *MUD-File:* A manufacturer-created file that defines the system and its anticipated network behavior. This file is hosted on the MUD file server. The location of the file is embedded as a "uniform resource locator" (MUD-URL). This URL is accessed when a device powers on and joins an end-user network. The authentication server (AAA) is responsible for enforcing the rules and implementing access control policies. The MUD-Manager is the core of an MUD instantiation. It extracts the secure URL, retrieves MUD files corresponding to the device (based on serial number or model), and sends the resulting configurations/profiles to the AAA service. The network access device (NAD) or Gateway is primarily responsible for routing. However, an internal firewall is usually used by the MUD-Manager via the AAA server to control the traffic and enforce rules. The communication protocols between the devices and MUD-manager may be one of these protocols, such as DHCP, LLDP, or X.509. The MUD manager verifies device signatures and configures the routers with the device manufacturer's security configuration. Traffic and behavioral patterns are recorded and described clearly in the MUD profiles through fine-grained monitoring at the gateway switches. SDN OpenFlow rules are then generated and mapped to profiles, and the profiles then produce compliance decisions, which can be performed by programs or applications. Recently, studies [30], [31] used the SDN architecture to achieve MUD-based IoT bootstrapping, administration, security, and exhibited good results. Sivanathan *et al.* [14] found that the IoT devices in industrial environments reveal succinct traffic patterns and features, which might be utilized for machine learning (ML)-based anomaly detection systems. The main goal of our research was to study the feasibility of MUD data to detect the behavioral patterns of network attacks and rogue devices. On this basis, this solution learns the behavioral patterns of the devices defined by their MUD profiles and introduces a system for enforcement and validation at run time. We track the operations of every device, controller, and application in the entire deployment domain by using traffic analysis on various time scales.

The key contributions of our research as follows.

1) Feasibility analysis of MUD for IoT networks.
2) A translation scheme from MUD to SDN rules that are proactively/reactively installed in dataplane switches.
3) A system with the ability to learn the expected behavioral patterns of the devices in the IoT network, fingerprint, and enforce compliance to MUD during the operations.
4) MUD-aware Edge Gateway and Controller that can detect intrusions and security breaches to the gateway or IoT devices, deriving insight into traffic behavior.
5) A network digital twinning (DT) ML-model that learns the network behavior and validates the configurations, operations, and potential countermeasures in the simulated network before they are deployed in the real infrastructure.

The remainder of this article is organized as follows. Section II gives a discussion on related publications. Section III introduces the proposed solution architecture. Sections IV and V present the design and performance evaluation, respectively. Section VI discusses the effectiveness of our proposal, lists future research, and concludes this article.

II. Related Work

We discuss some related proposals that addressed the security and the integration of softwarized network paradigms (e.g., SDN and NFV) in IoT networks. Bhunia and Gurusamy [10] proposed an SDN framework *SoftThings* that uses an ML-based system (IDS) for the detection of intrusions and anomalies in IoT network traffic. *IoT Sentinel [19]* describes an ML-based system for IoT infrastructures and uses a behavioral-based fingerprinting technique in their IDS. *IoTScanner* [15] proposes a framework that monitors the traffic pattern at datalink layers and performs deep packet inspection on traffic frame metadata headers, with a specific capture time window. The downside of this solution is its excessive false positives and unreliable binary classification methods. *IoT-Keeper* [18] is a framework for safeguarding Edge networks by identifying and isolating harmful network traffic. It implements network access control dynamically on a per-device, per-destination basis via *ad hoc* overlay networks. Hamza *et al.* [12] first suggested the use of MUD-profile-based behavioral analysis and anomaly detection (IDS) in an SDN-enabled IoT environment. The IDS can accept/reject, whitelist/blacklist traffic from specified devices according to

TABLE I
COMPARISON WITH RELATED WORK

Parameter	Related Work						This Work
	[13]	[11]	[17]	[9]	[23]	[10]	
Framework for Customization	✗	✗	✗	✗	✗	✗	✓
Static Signature based IDS	✓	✓	✓	✗	✗	✗	✓
Dynamic Behavioral IDS	✓	✓	✗	✓	✗	✗	✓
Software based Solution	✗	✓	✗	✗	✗	✗	✓
Switching Platforms Portable?	✗	✗	✓	✗	✗	✗	✓
Diversity of Data sets	✗	✗	✗	✗	✗	✗	✓
Evaluated in Real IoT Testbed	✓	✓	✗	✓	✗	✗	✓

TABLE II
DISCUSSION OF CLOSELY RELATED WORK

Reference	General Description	Gaps Addressed in our proposal
Hamza [12]	Automated system to generate MUD profile for devices from captured traffic.	MUD profile can be generated only for those devices join the network.
Hamza [12]	A system with MUD profiles for detecting abnormal behaviors in IoT.	This does not address how the MUD is enforced at run time in real networks.
Hamza [11]	A method for enforcing MUD (ACEs) in SDN-IoT network.	It lacks translation mechanisms and consideration for external MUD data.
Garcia [13]	A SDN system to enforce IDS based on MUD ACEs to mitigate ARP attacks.	Non-MUD devices and run-time MUD profile variations are not considered.
SoftMUD [23]	Translating scheme for Access-Control-Entries in MUD to OpenFlow rules.	MUD can provide hints for ACEs so that the controller can dynamically program the enforcement strategy.
ClearMUD [17]	MUD behavior ML models use clustering detection algorithm to flag malicious devices.	It lacks training/testing against real-life datasets. Their clustering is limited to 2 dimensions (only two features).

the vendor's or system administrator's specifications. The National Institute of Standards and Technology (NIST) has recommended the MUD to the industry as a cost-effective technique [22] for standardizing and automating the threat or risk detection process in highly dynamic IoT networks. *MUDgee* is a tool developed by Hamza *et al.* [11] that can construct MUD profiles from packet capture (PCAP) trace files. A reference SDN/MUD proposal was implemented by the NIST. Ranganathan [23] implemented the *SoftMUD* MUD policy in the SDN OpenFlow pipeline. The enforcement of MUD profiles is accomplished by flow-rule translation. Meidan *et al.* [9] suggested the extraction of transmission control protocol (TCP) connection information from traffic flows. Their classifier can differentiate between traffic from IoT and non-IoT devices. Deep learning and deep belief network (DBN) approaches for MUD/ACL-based data sets have been experimented in the reference work [20]. Research is ongoing to assess their reliability for behavioral pattern detection. Detecting anomalies and outliers is a crucial problem in a wide number of application disciplines. The outliers that are detected using the hybrid technique, the mean of the Euclidean and Manhattan Distances, are accurate and highly efficient. Singh *et al.* [17] addressed some of the crucial issues of the pervasive IoT paradigm in industrial networks and recommended fog/edge technologies. A comparison checklist between our proposal and closely related work is given in Tables I and II.

III. PROPOSED SDN-ENABLED NETWORK IDS (NIDS) ARCHITECTURE

We propose a framework that comprises intelligent security monitoring mechanisms and multiple IDS schemes that collaboratively monitor/protect an MUD compatible IoT/multiaccess Edge network. The framework can detect network-centered attacks and unique streams of traffic that contribute to this. The network topology that interconnects equipment, devices, and control systems used in an industrial facility shares useful data and intelligence. Access controls and communication to the IoT network are protected with IDS by tracking its operation through a combination of "coarse-grained (per-device) and fine-grained (per-flow)" analytics schemes. The solution comprises a security orchestrator (controller, IDS, security applications, and MUD services), edge nodes (switches and gateways, equipped with lightweight monitoring mechanisms and intelligent attack detection services), and other services to enforce the MUD rules/policies.

A. System Architecture

A conceptual SDN-based MUD-aware Edge infrastructure is depicted in Fig. 2. This consists of two stages: 1) identifying anomalies in local/Intranet or 2) external/Internet traffic. In *Stage-1*, we use mechanisms in switches utilizing coarse-grained (device level) telemetry to detect an abnormal burst of any exceptional bindings. These mechanisms then trigger the corresponding recovery process and alarm the *Stage-2* system that utilizes fine-grained (flow level) telemetry. We detect malicious activity by comparing the MUD profiles and drop the packets in the gateway. These components communicate with each other to dynamically control the flow-table rules within the switch when monitoring the device's network operations, actions, and traffic patterns. Traffic monitoring and route manipulation modules are deployed in the gateway switches for automatic provisioning and remote configuration of services. The key components of the security gateway are 1) monitoring module and 2) lightweight filter. The control plane (Edge controller) hosts the heavyweight ML-based Classifier and Anomaly based IDS. The gateway switch is where the packets from all the devices are forwarded, and we leverage the computing resources on this platform for security monitoring and anomaly/intrusion detection. On this basis, lightweight software modules are embedded in the gateway switches to perform first-level traffic analysis. The edge gateway forwards only suspicious flows that are marked for further inspection at the controller.

Specification-Based MUD-IDS: In mapping the MUD profiles to the flow rules, the IDS detects the operations and behavior of the devices that are not compliant with the edge infrastructure. The framework is responsible for various tasks, such as generating the SDN flow rule/configuration file based on the MUD device profiles, discovering new devices joining the network, and publishing a *match-action* rule set based on DNS bindings. The gateway uses an MUD collector engine that periodically scans the IoT network to check if a new device has joined. The MUD database is continuously updated by fetching the records from the MUD manager hosted in the edge network. The system learns the (normal/abnormal) behavioral profiles and establishes a clustering-based classification model. The inspection of packets is done to validate the

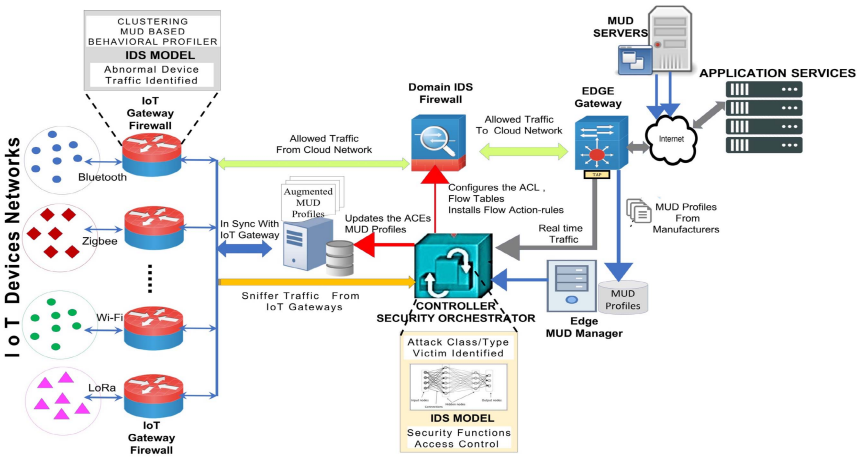


Fig. 2. SDN-based MUD-aware IoT network architecture.

flow rules inserted into the switch. The anomalous traffic is flagged for further analysis. The flows that are caught by the reactive flow rules are constructed to avoid subsequent packets from inspection, resulting in minimal system load. Our solution utilizes an IDS with traffic profiles (MUD) that recognizes the deviations from normal patterns and packet sequences, thereby detecting an attack to the network or specific device.

Anomaly Based ML-IDS: This system uses an ML technique to detect a malware/rogue device or a malfunctioning IoT operating in the network. The IDS is based on a neural network-based classifier, which captures and demodulates the traffic in the IoT network over time. The system extracts multiple fields from the packets based on the clustering and learns normal models of all the nodes and devices in the network. The learned models are compared with the deviations from the MUD profiles. The mitigation of the attacks is done through proactive ACLs/flow rules on the IDS, thereby dropping the malicious traffic and this ensures stringent security and access control are enforced over the simple MUD-profile-based IDS.

B. Adversary Model

The main adversaries are the attackers that target or launch attacks from compromised devices in the IoT infrastructure. We consider the malware to track, infect, and manipulate compromised devices in the network. Factory-default login credentials, open, unfiltered ports, and terminal access through *Secure Shell* or *TELNET* with blank passwords are all typical vulnerabilities. We assume the following threat conditions.

1) Manufacturers of the devices are not adversaries.
2) Devices can be vulnerable on arrival but have no back doors.
3) Gateway switches and controllers are not compromised.
4) Infrastructure facilitates secure/trusted execution platforms and integrity verifiable by remote attestation [21].
5) Automated identification and labeling of IoT devices.

C. Digital Twins in Industrial IoT

Digital twin is a new paradigm [2] for integrating physical-based models and data-driven models with virtual representation, performing predictive analytics and *What-if analysis* by using AI/ML and continuously updating a real system in the operation of the existing model. A network digital twin is a software simulation of an entire physical network, devices, and applications. The operating environment, deployed applications, and traffic pattern (including benign/normal and malign/attack) are emulated. Network dynamics are typically mimicked and generated through real-time interactions with the physical environment. In our framework, the digital twin network simulator can interoperate at any protocol layers with network management and monitoring tools, controllers, operator policy administrators, live applications, routers, firewalls, and other network devices. It includes the capabilities to collect, report, and visualize the result of the proposed flow rules and MUD profiles from the manufacturer under various operating conditions, including when exposed to cyberattacks.

IV. Design and Implementation

We describe the overall design and the core functionalities of the framework, including behavioral profiling, device fingerprinting, anomaly, and intrusion detection in a software-defined IoT network infrastructure. Our primary objective is to build a proactive and smart framework to construct typical IoT profiles by analyzing their communication patterns in the network, recognizing the anomalies, detecting attacks and abnormal devices, and taking remedial actions. The devices can communicate with other devices (device-to-device) and machines (machine-to-machine) within their local network via the IoT gateway and to the cloud network via the edge gateway through the firewalls. The major functions are as follows.

1) Secure configuration and bootstrapping of IoT devices by using MUD-profile-based compliance testing.
2) MUD specification-based security monitoring, behavioral profiling, and attack detection in the IoT gateway switches.
3) IDS using the ML deep-learning-based classification.

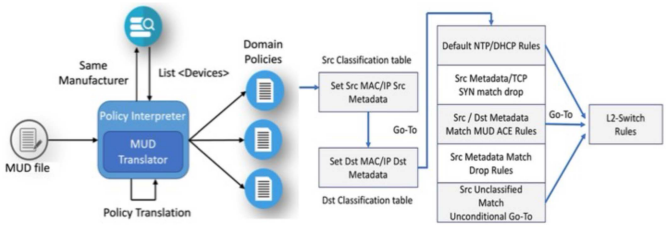


Fig. 3. MUD policy to OpenFlow rule translation.

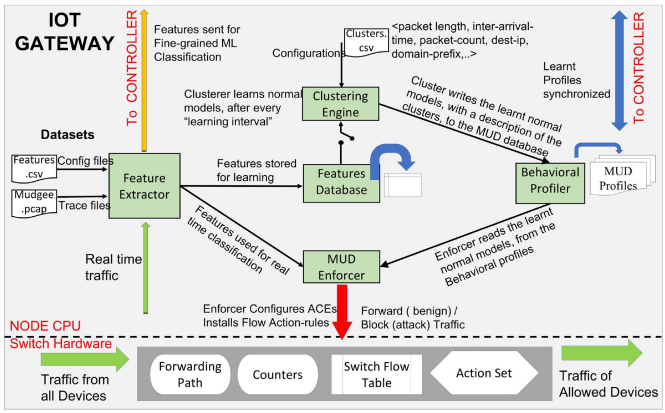


Fig. 4. IoT gateway architecture and operations.

A. MUD Compliance-Based Bootstrapping

IoT systems must be responsible for setting up and adjusting to the environment automatically without manual intervention and supervision. This condition can be achieved by deploying smart objects/things in the network that announce their behavioral requirements to the monitoring and control system. Connecting an IoT system to an existing infrastructure opens up many uncertainties without clear behavioral descriptions. Some research questions arise about the devices, services, behavior, features, limits, type of communication protocols, and possible conflicts with other devices in the network. A trusted private network service (e.g., MUD server), is responsible to read the uniform resource identifier (URI), pull up the MUD profile, and grant privileges. Although the MUD paradigm has brought in traction for device definitions and settings, it only offers an identification of a system and a list of the used addresses/ports. The standards do not define profiles to model dynamic behavior, and the functional intents of the devices. The workflow of translation in the OpenFlow pipeline is shown in Fig. 3.

B. Specification/Outlier-Based Anomaly Detection

The IoT Edge Gateway (Fig. 4) uses the resources of the switches and runs custom logic in addition to packet forwarding. The packet stream is obtained by the *Extractor* module from two sources, and the values/features are extracted from the packet header fields. The features are stored for a *learning interval* during the learning phase. The feature vectors (consisting of values extracted from each packet) are ingested into the *Clustering* engine to build the normal/benign behavioral model for every device. After each learning interval, the clustering engine continuously learns from the normal behavior at different periods/phases of the industrial process. This process ensures that the learned models are current and updated to be always applicable, although the normal behavior changes over time. Once the usual behavior models have been learned, the *Enforcer* module enforces the policies during the implementation. The Enforcer receives the characteristics of all IoT system traffic from the Extractor in real time. The Enforcer removes the misbehaving/rogue/victim devices from the network and installs new rules on the switches to block the malicious packets in two directions to the infected devices.

1) Feature Extraction: The initial composition of the raw statistics gathered from overall network traffic and individual device's activities must be preprocessed and is based on flow stats derived from all packet sequences carrying varying protocol data between devices. The flow features are derived from the packet headers to construct profiles, analyze, and model traffic behavior. These profiles define the behavior of flows, such as *data size*, *frame sizes,* and *packet-interarrival intervals* distribution. The *Feature Extractor* module (embedded in the switch) collects and processes packets from devices connected to its local IoT network and includes several subcomponents, as shown in Fig. 4. The packets can be fed in *pcap* format and translated to *json* format by the Extractor module. These JSON objects are consumed by the Extractor function and stored in a shared datastore. In a configuration file, the feature list is defined and can be adapted to various requirements and protocols. On the basis of some fields (e.g., MAC address) in the packets, the *Clusterer* differentiates between different devices. This process enables each device to hold its features independently so that the system can learn a typical behavior model for each device. The *Clusterer* uses this database collected over time as the training information models for the normal operation and behavioral profiles of all the devices in the domain.

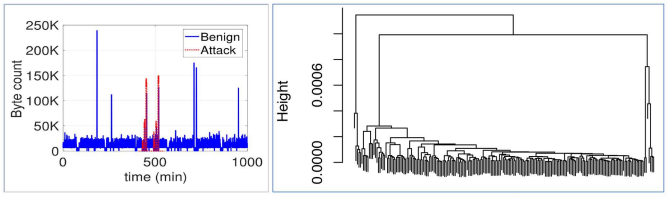
2) Compliance Testing and Behavioral Profiling: We define the components of our IoT behavioral model into two categories.

a) *Static Model:* Protocols used for communication.
b) *Dynamic Model:* Session interactions and communication patterns.

From every packet exchanged between the nodes in the network, we capture a *behavioral snapshot* that consists of protocol header fields. The compliance testing is done to determine whether the device behaves as specified in the MUD profile or not. Focusing on the vulnerable behavior of a device and ensuring the communications from a device are legitimate is required because the manufacturer of a device generates an MUD profile for its device over the cloud.

The profiling engine has two separate modules.

a) *Capture Intended Flows:* This module collects session details from TCP/IP traffic in the IoT network. It detects the services and captures bi-directional flows in the network.



(a) (b)

Fig. 5. (a) Traffic profile. (b) Dendrogram plot.

b) *Flow Rule to MUD:* MUD profiles for each device are generated. The runtime network activities are mapped to a known MUD profile. We verify its "likeness" to the established static MUD profiles given by manufacturers.

3) Clustering and Outlier Detection: The IoT gateway firewall (IoTGWF) collects network traffic and preprocesses it to create a data set. Service-based patterns are constructed across the data set by using a clustering method. We use a multistage clustering-based outlier detection technique to detect the anomalies and to differentiate between attacks and network exceptions (spikes/jitter). A *static threshold* may not differentiate busty attacks from benign traffic. Fig. 5 illustrates this phenomenon, and we can observe that the attack spikes (*red*) cannot be distinguished from the benign spikes (*blue*) using simple thresholds. Thus, we enhance the system by using a *dynamic model* to generate profiles in varying traffic scenarios to differentiate malicious flows from benign traffic. We take an example of a parameter *peak-request-rat*e used by this clustering algorithm. The peak request rate can be defined by the manufacturer in the MUD-File. For most of the low-end IoT devices that need to communicate with the manufacturer's services, the request rate can be calculated based on the characteristic of the device. For instance, getting 100 requests per second from a smart security camera is indeed unusual because 100 moving objects are unlikely to pass the smart security camera in 1 s.

Clustering refers to unsupervised learning methods that partition a data set of n units, and the sets in these partitions are called *clusters*. In *hierarchical clustering* [24] for each data point in the cluster, this technique calculates a metric called the *silhouette* score. The data points are standardized for outliers, without the usual *standard deviation* method. For the scope of this article, we consider that the data are represented by the 4D vector X in which we can choose up to *four* features from the traffic data. The administrator can configure the clustering engine with any feature set. For example, *<packet size, packets interarrival time, packet count, protocol>* can be one data-point plotting the feature vector X. These features are denoted as x_1, x_2, x_3, and x_4. Although we have a 4-D vector, we will only be looking at 3-D vectors where the x-axis is always fixed as the *interarrival time* (the time gap between the stream of packets within the sampling interval, as measured at the receiver end). Hierarchical clustering begins by building separate clusters for each data point utilizing *Euclidean* distances.

The formula is given as

$$d(i,j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}.$$

Algorithm 1 Hierarchical Clustering Algorithm

Input

Data Points: X{$x_1, x_2 \cdots x_n$ with n objects, where each x_i can be categorical attribute represented by a vector.

Output

Silhouette score

Definitions

pattern Similarity: Two data objects x_1 and x_1 are defined as similar iff (a) $d(x_1, x_1) \ll \tau$ and (b) d $(x_1, x_2)0$, if $x_1 = x_2$.
Mean Dissimilarty: of the point i to some cluster C_k as the mean of the distance from i to all points in C_k where $C_k \neq C_i$
Cluster: A cluster C_i is a subset of a dataset X, where for any pair of x_i's, say $(x_i, x_j) \in C$, $d(x_i, x_j) \ll \tau$.
Profile: A profile of a cluster C_i is a mean value, $\mu(x_{\mu,1}, x_{\mu,2} \cdots X_{\mu,d})$ of dataset X, where each $x_{\mu,j}$ is the mean of the jth column of the cluster C_i $d(i, j)$: the distance between i and j in that cluster C_i.

Method

For evrey i $\in C_i$, Mean distance,

$$a(i) = \frac{1}{|C_i - 1|} \sum_{j \in C_i i \neq j} d(i,j)$$

$a_i \leftarrow$ estimate of position in C_i.

For each data point, i $\in C_i$ the smallest mean distance of i to all points

$$b(i) = \min_{k \neq i} \frac{1}{|c_k|} \sum_{j \in C_k} d(i,j)$$

We calculate the silhouette score of a data point i,

$$s(i) = \frac{b(i) - a(i)}{\max[a(i), b(i)]} \text{ if } |C_i| > 1$$
$$s(i) = 0 \text{ if } |C_i| = 1$$

return s(i)

After an iterative merging process, we choose either one of the y-axis or z-axis or one of the features x_1, x_2, x_3, x_4 and plot the *dendrograms*. As shown in Fig. 5(b), the tree structure defines the hierarchical relationship between clusters (number of clusters on the x-axis, distances on the y-axis). Once the hierarchical clustering has been performed, an optimal axis is calculated to cut the dendrogram with a horizontal line. We use the popular *silhouette analysis* technique as the metric to generate the most optimal clusters. A *silhouette score* $s(i)$ is computed for every data point (using Algorithm 1) which denotes the best coordinate for that point within its cluster. Suppose the data were clustered into k *clusters*, we can apply the method to calculate $s(i)$ for the metadata from the packets and observe that $-1 \leq s(i) \leq 1$. The clusters are optimally formed when the *silhouette score is* high, and lower values imply that extremely many/few clusters are created.

The outlier score for each data point is calculated and compared to a user-defined *cutoff* value. The data points are sorted in accordance with their scores, and outliers O_{ij} (including the *minor outliers*) are identified in terms of *threshold* τ and the average of the (95th, 96th, and 99th) *percentiles*. Thus, we can determine the anomalous flows, and only the data/packets related to the *suspicious* flows are forwarded to the control plane ML-application for deep investigation.

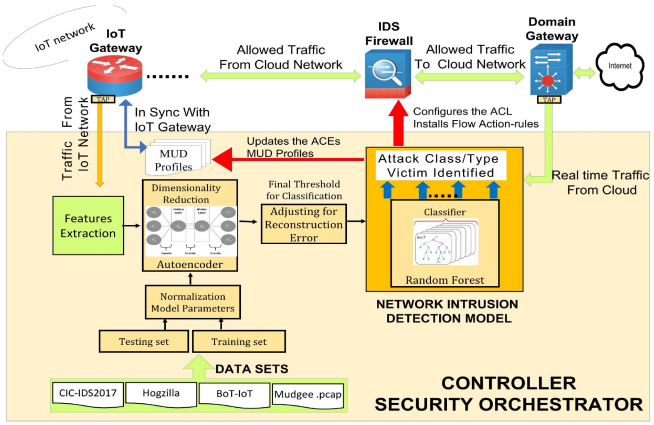


Fig. 6. Security controller architecture.

C. Hybrid ML-Based Anomaly Detection

The controller and security orchestrator node provides many computing resources, a global view, and integrated knowledge of the entire network. The main functions (monitoring, analysis, and classification) are built in the control plane (Fig. 6).

We adopted a hybrid pipeline (deep and shallow Learning, Fig. 7), nonsymmetric deep autoencoder (NDAE) model [29], and random forest (RF) as the output classifier. This unconventional design makes the NDAE system utilize less CPU and training time with higher accuracy. After careful analysis and experimentation with this new hybrid model, we improvised with the composition of the hidden layers, neurons, and feature set from the more recent data sets for training. The input feature sets (traffic samples) are reduced by the NDAE stack, and the resulting encoded representations (features) are fed to the output RF module, which learns and classifies the traffic into multiple attack types.

1) Data Sets: We generated MUD profiles from *MUDgee* [11] software, which takes a *pcap* input file from real traffic in the testbed. To ensure that no data sets were faulty, we validated MUD profiles and checked for policy consistencies and semantics. We used *MUDgee* data, which included 28 000 entries and approximately 32 PCAP traces. The CICIDS2017 [25], Bot-IoT [26], Hogzilla [27], and malicious IoT traffic [28] were replayed to simulate the representative traffic of real-world network [12]. We performed normalization and reorganized the samples from the data set. Network intrusion data set are imbalanced, especially in smart CPS infrastructures [28]. High *imbalanced* data sets have a predominant *bias* for most class data, leading to high *false positives* in detection. The data set goes through a preprocessing stage for: a) resampling and b) normalizing labels to tackle these shortcomings. Thus, the accuracy improves in detecting the class of intrusions and sustains the speed.

2) Feature Selection: At regular intervals, we collect values from flow counters, calculate the statistical features over a sliding 1 to 4-min window, and this process generates approximately 20 features per flow (see Table III), including the bidirectional data. To avoid the overhead of deep packet inspection, we perform flow-level analytics in the first stage.

TABLE III
DEFINITIONS OF FEATURE VECTORS

Structural Features	Temporal Features
Byte Count	Bytes In/Out Ratio
Packet Count	Packets In/Out Ratio
Flow Count	Flows In/Out Ratio
Unique Host count	Average Bytes/Packet
Unique Source Port count	Average Packets/flow
Unique Destination Port count	Average Bytes/flow
Unique Src/Dst IP Pair count	Average TCP/IP-Sessions length

Only upon the detection of suspicious flow(s) from *Stage-1*, packet-level attributes (*Destination IP addresses, ports or interarrival times*) and flow-rule history are considered. The behavioral profile can be constructed by collecting the data of its operations and how it functions. Its communication patterns are all consolidated as *fingerprints*. We generate a single feature vector by using the metadata from consecutive packets, $p^i \rightarrow p^{i+1} \rightarrow p^{i+2} \cdots \rightarrow p^n$, and the packet sequences serve as valid semantic data for the sessions. The set of all these characteristic vectors fits the system's observed behavioral profile and can be inputted into ML-based classifiers.

3) Deep Learning Neural Network (ML)-Based Classifier: Neural networks define the nonlinear relationships that can be fitted to data. A neuron takes one or more inputs and calculates a single output given by the equation

$$h_{w,b}(x) = f(x) = \left(w^T x \ + \ b\right)$$

where x denotes the input vectors for the unit, w is a vector of weights, b is an intercept term called the bias, and $f : \mathbb{R} \rightarrow \mathbb{R}$ is a so-called *activation function*, where each *chunk* represents all the observations from one *object*. In the current problem, the unlabeled data are the chunks $x_1, x_2, x_3, \ldots, x_n$, where each chunk is a K-dimensional vector of network *packets*. Effectively, the autoencoder is represented by the function $h_{w,b}(x) \approx x$ for any data point, $x \in \mathbb{R}^K$ where h is the output of the *hypothesis* from a network with weights w and bias b on an input x. Based on the principle of multiple-class classification, the system was trained with the data from its IoT network devices by using benign features. It can detect whether a traffic observation belongs to a specific trained class.

D. Digital Twin With MUD and Behavioral Monitoring

We present a design for integrating the *network twin* with the NIDS, as shown in Fig. 8. The deployment domain contains two zones: 1) physical IoT edge network and security gateway (Physical Twin) and 2) digital twin Manager replicates the entire infrastructure in a virtual network, runs simulations of IoT network, tests the policies, and resolves any behavioral conflicts with the vendor-MUD profiles or with operator's policies associated with the domain. The Twin Manager runs a battery of compliance test suites, simulates attack traffic, and by applying the MUD-rule/SDN action in the virtual network. This trial run is conducted to validate all the planned run-time actions before propagating to the real physical network (via SDN controller and switches). The decisions derived from the MUD profile and SDN flow rules during the TEST phase can result in a state inconsistent with the security posture or network QoS policies of the deployment domain.

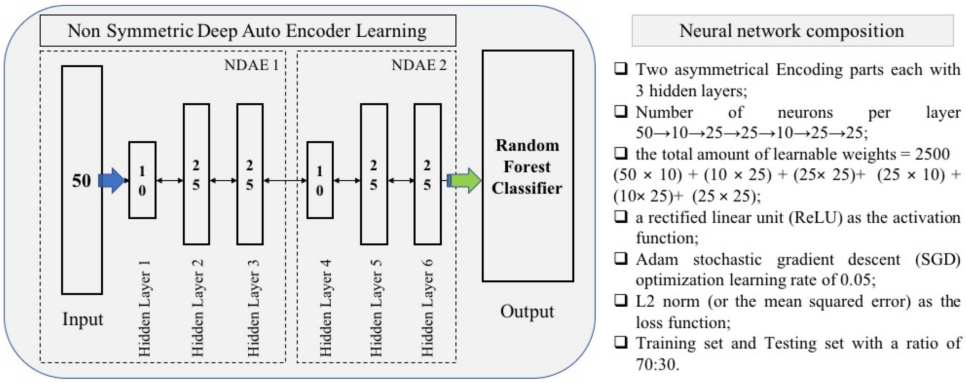


Fig. 7. Hybrid deep learning pipeline composition.

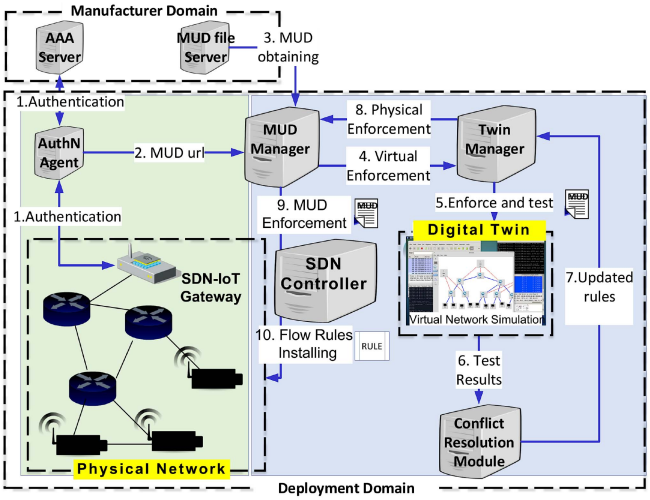


Fig. 8. Integrating MUD-enhanced NIDS with digital twins.

The *Conflict Resolution* component suggests any alternative remediation plan. The MUD rules from manufacturers can be enforced to suit the deployment domain because they are suggestive (not directive).

V. Performance Evaluation

We deployed a variety of multiaccess devices that constitute the larger connectivity to the public Internet/Cloud at the Edge.

A. Bootstrapping Performance

1) *Bootstrapping/Authenticating Latency [Fig. 9(a)]:* The overhead increases with an attack ratio of 0.1–0.9. Our solution bootstraps nine times faster than a legacy SDN stack.
2) *Packet Loss Ratio:* During network congestion, our IDS shows a gradual loss due to faster processing in the data plane. The legacy IDS drops more packets over time.
3) *Packet Transmission Rate [Fig. 9(b)]:* Our IDS sustains a wire speed throughput of 1000 pkts/timeslot (20 Mb/s), an average rate of 720 pkts/timeslot during attacks. With the legacy IDS, it drops down to 200 pkts/timeslot.

Thus, in terms of the key performance metrics for IoT networks, the applications can achieve the guaranteed quality-of-service, ultralow latencies with security by using our IDS.

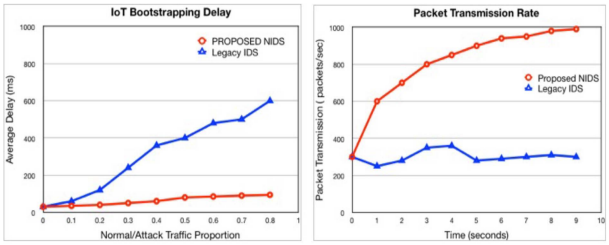


Fig. 9. (a) Smart authentication. (b) Throughput performance.

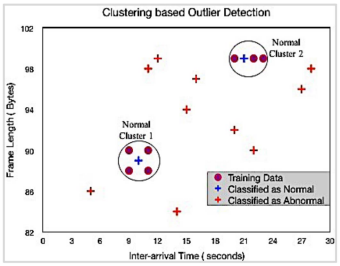


Fig. 10. Clustering model.

B. Clustering and Outlier Detection

To evaluate the *specification-based* IDS, we set up a network of IoT devices to emulate a typical smart CPS infrastructure. The value range of the parameters (*interarrival time, size*) is defined: normal (10 and 11 s, 90 and 91 bytes) and abnormal (11 s, 98 bytes). The experiment begins with normal operation, and the NIDS (Gateway and the Controller) learns and establishes normal models for two devices. The 2D vector—*interarrival time, frame length* is configured in the IDS, and the training data set is created by running traffic for some hours. Some anomalies are injected at various devices. The devices are configured for the transmission of packets with shorter inter-arrival and longer inter-arrival intervals to establish mixed behaviors. The data points are plotted as they arrive at the IDS, as shown in Fig. 10.

The IDS learns the normal models that resulted in two "normal clusters." The packets (data points) from the network pass through the IDS classifier (in real time) and are plotted. The IDS model marks the normal clusters (in circles) and abnormal packets that are outliers as "+." The *Enforcer* in the IoT gateway installs the flow rules in the forwarding table corresponding to ACL entries in the MUD profiles (generated from the *Clusterer*

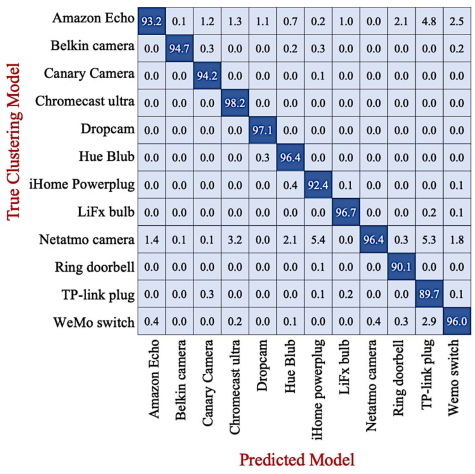
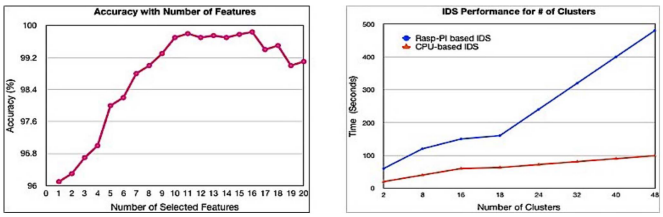


Fig. 11. Confusion matrix.



(a) (b)

Fig. 12. (a) Accuracy variation. (b) Scaling with cluster diversity.

output). The FLOW RULE constitutes the *Forward* Action for normal packets and the *Drop* Action for these "+" matching abnormal packets. Fig. 11 shows the confusion matrix plotted for the devices from the test data and proves the performance of the classifier model with high confidence. Based on the functionality gathered in the dataplane, we verified the efficiency of our statistical detection. We aimed to collect the least data for optimal resource usage because these features are captured from the switch counters/table at wire speed.

Fig. 12(a) confirms the results that the top 10–12 top features exhibit the maximum accuracy, and the features beyond 16 tapers down due to an increase in *bias*. Fig. 12(b) shows the clustering and classification performance of the gateway platform compared with *Raspberry-Pi/ARM-based* single-board system. The performance (detection time on the *y*-axis) improves linearly with the increase in data points and the formation of more clusters (includes *normal* and *benign*). From the graph, for clustering count = 18, all the data points from 1 to 5-min sliding windows are processed by the Clustering-IDS under 160 s at the gateway node with single-board *ARM* processor. The same IDS processed the data points and detected the attack type under 60 s on a *CPU/Intel-Core-i5* based node. Thus, we infer that our scheme is portable and can perform close to the wire speed even on low-cost single-board IoT gateway and on any switching platforms with moderate CPU power.

C. Mirai IoT Botnet Experiment

We simulated the operations of the Mirai botnet [5]—*scanning, infiltration, victim finding, infection*, and *controlling* the infected devices and hosts. In each phase of the botnet life cycle (Fig. 13), the botnet behavior and communication

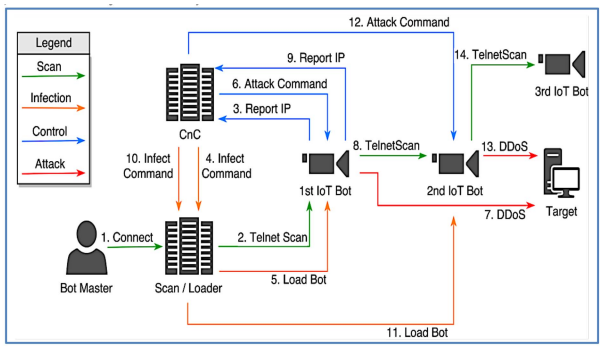


Fig. 13. Mirai Botnet operation life cycle [5].

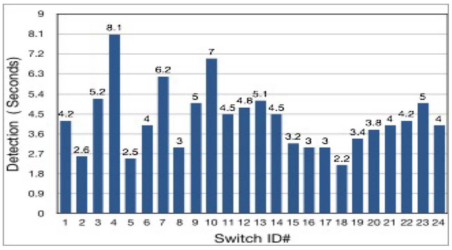


Fig. 14. Botnet detection speed.

TABLE IV
MIRAI DETECTION IN PHASES

Phase	Packets/s	Detection (ms)	TPR(%)
Standby	0.06	4,081,554	33
Pre-Infection	459.99	558	100.00
Infection	729.18	279	94.50
Scanning	754.20	189	100.00
DoS	1,449.90	99	90.63
Average	900.81	279	96.03

patterns leave *fingerprints* that deviate from the MUD profiles, which can be detected (seen as *outliers*) by using the Clusterer model. The Mirai botnet manifests in approximately ten different variants. We randomly picked a device from one of the devices to infect it with Mirai malware and activate the botnet in the network (e.g., Scanning of ports 23 and 2323 for new devices to infect). The timer starts when the infection begins and stops when one of the gateway switches (NIDS) detects it. The time taken to detect the botnet is measured. For every iteration, a victim IoT device for Mirai infection is chosen randomly. Fig. 14 plots the time taken to detect in each iteration. The framework takes on average approximately 4.5 s to detect Mirai patterns and identify the victim node in the network. The bar chart shows the shortest detection time of 2.5 s by switch#5 and longest time of 8.1 s by switch#4 under different iterations.

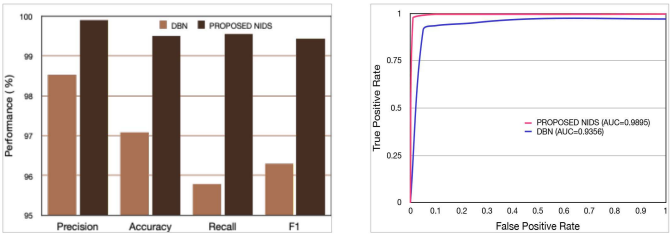
Table IV shows the detection performance (speed and accuracy) of the IDS in different phases of an active botnet session. The average delay is approximately 279 ms after approximately 200 packets (out of 900 packets during *pre-infection*). Behavioral profiling is so sensitive that attacks are detected during the *stealth scanning* phase itself (ahead of the *Infection* phase).

D. Performance of Attack Detection and Classifier

In pattern recognition and traffic classification, binary class and multiclass are two major methods. Table V shows that

TABLE V
IDS PERFORMANCE FOR DIFFERENT ATTACKS

Attack Type	Recall	Precision	FP Rate	TP Rate
Benign	1.000	1.000	0.000	1.000
FTP-Patator	1.000	1.000	0.000	1.000
SSH-Patator	1.000	1.000	0.000	1.000
DDoS	1.000	1.000	0.000	1.000
Heartbleed	1.000	1.000	0.000	1.000
PortScan	1.000	0.999	0.000	1.000
DoSHulk	0.999	1.000	0.000	0.999
DoSGoldenEye	1.000	1.000	0.000	1.000
WebAttack: BruteForce	0.968	0.901	0.007	0.968
WebAttack: XSS	0.904	0.946	0.003	0.901
Infiltration	1.000	1.000	0.000	1.000
WebAttack: SQL	1.000	0.999	0.000	1.000
Botnet	1.000	1.000	0.000	1.000
DoS Slow HTTP Test	0.990	0.990	0.000	0.990
DoS Slow Loris	0.990	0.990	0.000	0.990
Weighted Average	**0.990**	**0.988**	**0.001**	**0.990**



(a) (b)

Fig. 15. (a) ML key metrics. (b) Receiver operating characteristic.

the proposed hybrid NIDS can detect most network-centric attacks, achieving the highest sensitivity (0.998), and lowest false positive rate (0.02). The scan attack (e.g., *Mirai reconnaissance*) detection achieves 0.998 accuracy and 0.986 F1-score. The proposed solution is faster in detecting *scanning* attacks and their variants, proving the efficacy of our monitoring and detection mechanisms at multiple stages of the attack cycle. We evaluated our NIDS with standard methods and compared the key metrics with another closely relevant deep learning algorithm from a *DBN model* [20], as shown in Fig. 15(a). The ROC curves in Fig. 15(b) show that our NIDS is superior in terms of area under the curve (AUC, 0.998) compared with the other *DBN model* (0.935). The data set is composed of heuristics, which are approximately 100 000 samples: with 60% benign/40% malign ratio samples. In BotIoT [26] (5 attack classes, 8000 malign, and 60 000 benign) and CICIDS [25] (8 types of attacks, 5000 malign samples, and 60 000 benign).

E. Result Discussion

The results of the performance evaluation are promising. All attacks (from internal and external vectors) on any compromised IoT device or gateway in the Edge infrastructure were identified by our monitoring and detection mechanisms (achieving approximately 100% *true positive* rate). The attack detection scheme showed an extremely low *false alarm* rate, with a mean false positive rate of 0.007 $\pm$ 0.01. In various evaluated scenarios from the launch of attacks, we observed an average detection time of approximately 279 ms $\pm$ 99 ms to detect botnet attacks and approximately 4 to 8 s for sophisticated malware attacks. The systems and the framework were created with security by design, utilizing proven and agile/reliable monitoring components. We focused on the primary vulnerabilities affecting IoT-based infrastructures, such as insufficient security configurations, insecure network connections, behavioral anomalies, malicious devices, and noncompliance with the manufacturer's operational standards. The key performance indicators in these critical axes are given in the following.

Authentication: The lightweight cryptographic verification and MUD-based bootstrapping ensure resistance to spoofing, forging, and adversary-in-the-middle snooping attacks.

Access Control: The combined trust in access and credentials is achieved by using the MUD *manifest* and OpenFlow *rules*.

Network Security: Combining device-level (ACL+MUD), network-level IDS, and cryptographic mechanisms we can resist most modern attacks, such as cloning, DDoS, Botnet, and malwares that are known in most recent data sets.

Behavior Evolution: The behavior of devices in the IoT ecosystem evolves because new features/functionalities are added and with software/firmware upgrades. The IETF standard is increasing the scope of MUD to explicitly describe the predicted actions and intents of IoT devices in the network.

VI. CONCLUSION AND FUTURE WORK

We proposed an SDN-based framework that comprises intelligent security monitoring mechanisms and multiple IDS that collaboratively monitor/protect an MUD compatible IoT/multi-access edge network. This framework aims to deal with common attacks in the edge network by monitoring the MUD-compliant IoT devices. We posit that the MUD standard expands the possibility for explicitly describing the expected activities and intents of IoT devices connected to the network. We designed mechanisms for detecting behavioral anomalies from the network traffic from/to the IoT ecosystem and enhanced the network digital twin with MUD profiles for simulation/conflict resolution in operator policies/solutions. We tracked the operations of every device, gateway, switch, controller, and application in the entire deployment domain by conducting traffic analysis (flow level/device level) on various time scales. In future work, we plan to design a hybrid IoT security gateway for integrating IIoT, a smart behavioral filter, and data sets for diverse applications of IoT, including wireless sensory networks and traditional Industrial control systems (e.g., MUD profiles for SCADA-based PLC policies). Diving more into the realm of deep learning with graphs is an emerging field, and we plan to perform network topology graph validations on more robust feature vectors encoded on the graph. The network digital twin model in collaboration with behavioral analytics and software-defined IDS improves the cyber resiliency, security, and compliance in Industry 4.0 environments. Our study of merging a variety of cutting-edge technologies into a novel framework lays the groundwork for future research in this exciting new field.

REFERENCES

[1] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity Internet of Things devices," in *Proc. ACM Asia Conf. Comput. Commun. Security*, 2016, pp. 461–472.

[2] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019.
[3] F. Loi, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivaraman, "Systematically evaluating security and privacy for consumer IoT devices," in *Proc. Workshop Internet Things Security Privacy*, 2017, pp. 1–6.
[4] A. M. Zarca *et al.*, "Security management architecture for NFV/SDN-aware IoT systems," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8005–8020, Oct. 2019.
[5] R. Hallman, J. Bryan, G. Palavicini, Jr., J. Divita, Joseph, and J. Romero-Mariona, "IoDDoS—The Internet of distributed denial of service attacks—A case study of the mirai malware and IoT-based botnets," in *Proc. 2nd Int. Conf. Internet Things Big Data Security (IoTBDS)*, 2017, pp. 47–58, doi: 10.5220/0006246600470058.
[6] B. Bezawada, I. Ray, and I. Ray, "Behavioral fingerprinting of Internet-Things devices," *Wiley Interdiscipl. Rev. Data Min. Knowl. Discov.*, vol. 11, p. e1337, Aug. 2019.
[7] "Manufacturer usage description specification Internet-Draft draft-ietf-opsawg-mud-18," Internet-Draft, IETF, 2018.
[8] *osMUD-Open-Source MUD Manager*. Accessed: Sep. 23, 2021. [Online]. Available: https://osmud.org
[9] Y. Meidan *et al.*, "ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis," in *Proc. Symp. Appl. Comput.*, 2017, pp. 506–509. [Online]. Available: https://doi.org/10.1145/3019612.3019878
[10] S. S. Bhunia and M. Gurusamy, "Dynamic attack detection and mitigation in IoT using SDN," in *Proc. ITNAC*, Melbourne, VIC, Australia, 2017, pp. 1–6, doi: 10.1109/ATNAC.2017.8215418.
[11] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, and V. Sivaraman, "Clear as MUD: Generating, validating and applying IoT behavioral profiles," in *Proc. Workshop IoT Security Privacy*, 2018, pp. 8–14.
[12] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on lot devices via SDN-based monitoring of MUD activity," in *Proc. ACM Symp. SDN Res.*, 2019, pp. 36–48.
[13] S. N. M. García, A. M. Zarca, J. L. Hernández-Ramos, J. B. Bernabé, and A. S. Gómez, "Enforcing behavioral profiles through software-defined networks in the Industrial Internet of Things," *Appl. Sci.*, vol. 9, p. 4576, Oct. 2019. [Online]. Available: https://doi.org/10.3390/app9214576
[14] A. Sivanathan *et al.*, "Characterizing and classifying IoT traffic in smart cities and campuses," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2017, pp. 559–564, doi: 10.1109/INFCOMW.2017.8116438.
[15] S. Siby, R. R. Maiti, and N. O. Tippenhauer, "IoTScanner: Detecting privacy threats in IoT neighborhoods," in *Proc. 3rd ACM Int. Workshop IoT Privacy Trust Security*, 2017, pp. 23–30, doi: 10.1145/3055245.3055253.
[16] G. S. S. Chalapathi, V. Chamola, A. Vaish, and R. Buyya, "Industrial Internet of Things (IIoT) applications of edge and fog computing: A review and future directions," 2019. [Online]. Available: arXiv:1912.00595.
[17] S. Singh, A. Atrey, M. L. Sichitiu, and Y. Viniotis, "Clearer than MUD: Extending manufacturer usage description (MUD) for securing IoT systems," in *Proc. Int. Conf. Internet Things*, Jun. 2019, pp. 43–57.
[18] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 45–59, Mar. 2020.
[19] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, 2017, pp. 2177–2184.
[20] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Security Appl.*, vol. 50, Feb. 2020, Art. no. 102419.
[21] G. Dessouky *et al.*, "LO-FAT: Low-overhead control flow attestation in hardware," in *Proc. ACM/EDAC/IEEE Design Autom. Conf.*, 2017, pp. 1–6.
[22] V. Jeffrey, K. Rick, L. Phillip, and A. Sophia, "NISTIR 8222: Internet of Things (IoT) trust concerns," NIST, Gaithersburg, MD, USA, White Paper, 2018.
[23] M. Ranganathan, D. Montgomery, and O. E. Mimouni, "Soft MUD: Implementing manufacturer usage descriptions on OpenFlow SDN switches," in *Proc. Int. Conf. Netw. (ICN)*, 2019, pp. 49–54.
[24] J. F. Nieves and Y. C. Jiao, "Data clustering for anomaly detection in network intrusion detection," in *Proc. Int. Conf. Res. Alliance Math Sci.*, 2009, pp. 1–12.
[25] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.
[26] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.
[27] P. A. A. Resende and A. C. Drummond. (2018). *The Hogzilla Dataset*. [Online]. Available: http://ids-hogzilla.org/dataset
[28] *Kitnet Dataset*. (2018). [Online]. Available: https://goo.gl/iShM7E
[29] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb 2018.
[30] P. Krishnan, J. S. Najeem, and K. Achuthan, "SDN framework for securing IoT networks," in *Ubiquitous Communications and Network Computing* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 218. Cham, Switzerland: Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-73423-1_11
[31] P. Krishnan, K. Jain, K. Achuthan, and R. Buyya, "Software-defined security-by-contract for blockchain-enabled MUD-aware Industrial IoT edge networks," *IEEE Trans. Ind. Informat.*, early access, Jun. 2, 2021, doi: 10.1109/TII.2021.3084341.

Prabhakar Krishnan (Member, IEEE) received master's degree in computer science and engineering from the National Institute of Technology Tiruchirappalli, Tiruchirappalli, India, in 1995, and the Ph.D. degree in cybersecurity from Amrita University, Amritapuri, India, in 2020.

He is a Research Scientist with the Department of Cyber Security Systems and Networks, Amrita Vishwa Vidyapeetham, Amritapuri. He has over two decades of Industry experience in the USA. He was a Visiting Adjoint Professor with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA. His current research interests are primarily in cybersecurity, with special focus in network security, quantum safe Internet and cryptography, network softwarization and architecture, SDN/NFV, 6G communications, cyber forensics and IoT standardization.

Dr. Krishnan is a working member of the core-development group of OpenAirInterface Software Alliance 5G Wireless community at Eurecom in Europe. He is a member of the IEEE Computer Society.

Kurunandan Jain received the bachelor's degree in mathematics, the M.Sc. degree in applied mathematics, and the Ph.D. degree in mathematical physics from Imperial College London, London, U.K., in 2011, 2012, and 2017, respectively.

He is currently an Assistant Professor with the Department of Cyber Security Systems and Networks, Amrita Vishwa Vidyapeetham, Amritapuri, India. His research interests include cryptography, authentication and privacy design, cryptanalysis, digital signature, quantum cryptography, and applications for future networks.

Rajkumar Buyya (Fellow, IEEE) received the Master of Engineering degree in computer science and engineering from Bangalore University, Bengaluru, India, in 1995, and the Doctor of Philosophy degree in computer science and software engineering from Monash University, Melbourne, VIC, Australia, in 2002.

He is a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems Laboratory with the University of Melbourne, Melbourne. He is also serving as the founding CEO of Manjrasoft, Melbourne, a spin-off company of the University, commercializing its innovations in Cloud Computing. He has authored over 725 publications and seven textbooks, including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets, respectively. He is one of the highly cited authors in computer science and software engineering worldwide (H-index = 137, G-index = 304, and more 100000 citations).

Pandi Vijayakumar (Senior Member, IEEE) received the B.E. degree in computer science and engineering from Madurai Kamaraj University, Madurai, India, in 2002, the M.E. degree in computer science and engineering from the Karunya Institute of Technology, Coimbatore, India, in 2005, and the Ph.D. degree in computer science and engineering from Anna University, Chennai, India, in 2013.

He is the former Dean and currently an Assistant Professor with the Department of Computer Science and Engineering, University College of Engineering Tindivanam, Melpakkam, India, which is a constituent college of Anna University, Chennai, India. He has completed four Ph.D. candidates successfully. He has also authored or coauthored many quality papers in various IEEE transactions/journals, ACM transactions, Elsevier, IET, and Springer journals. Untill now, he has authored four books for various subjects that belong to the department of computer science and engineering. His current research interests include key management in network security, VANET security, and multicasting in computer networks.

Dr. Vijayakumar is an Associate Editor of *International Journal of Communication Systems* and *Security and Privacy* (Wiley). He is also serving as an Associate Editor of *International Journal of Organizational and Collective Intelligence*, *International Journal on Semantic Web and Information Systems*, *International Journal of Software Science and Computational Intelligence*, *International Journal of Cloud Applications and Computing*, *International Journal of Digital Strategy, Governance, and Business Transformation* (IGI Global), and *PLOS One*. He is serving as a Technical Committee Member in Computer Communications (Elsevier), and an Academic Editor of *Security and Communication Networks* (Wiley | Hindawi).

Anand Nayyar (Senior Member, IEEE) received the Ph.D. degree in computer science from Desh Bhagat University, Amloh, India, in 2017, in the area of wireless sensor networks and swarm intelligence.

He is currently working with the School of Computer Science, Duy Tan University, Da Nang, Vietnam. Published more than 125 research papers in various high-quality ISI-SCI/SCIE/SSCI impact factor journals cum Scopus journals, over 50 papers in international conferences indexed with Springer, IEEE Xplore, and ACM Digital Library, over 40 book chapters in various SCOPUS, Web of Science Indexed Books with Springer, CRC Press, Elsevier, and many more with citations: Over 4100, H-Index: 35, and I-Index: 111. He has authored/coauthored cum edited over 30 books of computer science. He has 11 Australian patents, two Indian patents, one Indian copyright to his credit in the area of wireless communications, artificial intelligence, cloud computing, IoT, and image processing. He is currently researching in the area of wireless sensor networks, IoT, swarm intelligence, cloud computing, artificial intelligence, drones, blockchain, cyber security, network simulation, and wireless communications.

Dr. Nayyar awarded over 32 Awards for Teaching and Research-Young Scientist, the Best Scientist, the Young Researcher Award, the Outstanding Researcher Award, and the Excellence in Teaching. A certified professional with over 75 professional certificates from CISCO, Microsoft, Oracle, Google, Beingcert, EXIN, GAQM, and Cyberoam. He is associated with more than 500 international conferences as a programme committee/chair/advisory board/review board member. He is acting as an Associate Editor for *Wireless Networks* (Springer), *Computer Communications* (Elsevier), *International Journal of Sensor Networks* (Inderscience), *Frontiers in Computer Science*, *Human Centric Computing and Information Sciences*, *IET Quantum Communications*, *IET Wireless Sensor Systems*, *IET Networks*, *International Journal of Distributed Systems and Technologies*, *International Journal of Information Security and Privacy*, *International Journal of Cognitive Informatics and Natural Intelligence*, and *International Journal of Gynecological Cancer*. He is acting as the Editor-in-Chief of IGI-Global, USA Journal titled *International Journal of Smart Vehicles and Smart Transportation*. He has reviewed more than 1400 Articles for various Web of Science indexed journals. He is a Life Member of ACM and a member of more than 50 associations.

Muhammad Bilal (Senior Member, IEEE) received the B.Sc. degree in computer systems engineering from the University of Engineering and Technology Peshawar, Peshawar, Pakistan, in 2008, the M.S. degree in computer engineering from Chosun University, Gwangju, South Korea, in 2012, and the Ph.D. degree in information and communication network engineering from the School of Electronics and Telecommunications Research Institute (ETRI), Korea University of Science and Technology, Seoul, South Korea, in 2017.

From 2017 to 2018, he was with Korea University, where he was a Postdoctoral Research Fellow with the Smart Quantum Communication Center. In 2018, he joined the Hankuk University of Foreign Studies, Seoul, where he is currently working as an Associate Professor with the Division of Computer and Electronic Systems Engineering. His research interests include design and analysis of network protocols, network architecture, network security, the IoT, named data networking, blockchain, cryptology, and future Internet.

Dr. Bilal was a Technical Program Committee Member on many international conferences, including the IEEE VTC, the IEEE ICC, and the IEEE CCNC. He serves as an Editor for the IEEE Future Directions Ethics and Policy in Technology Newsletter and the IEEE Internet Policy Newsletter.

Houbing Song (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in August 2012, and the M.S. degree in civil engineering from the University of Texas, El Paso, TX, USA, in December 2006.

In August 2017, he joined the Department of Electrical Engineering & Computer Science, Embry-Riddle Aeronautical University, Daytona Beach, FL, USA, where he is currently an Assistant Professor and the Director of the Security and Optimization for Networked Globe Laboratory. He served on the faculty of West Virginia University, Morgantown, WV, USA, from August 2012 to August 2017. In 2007, he was an Engineering Research Associate with the Texas A&M Transportation Institute, College Station, TX, USA. He has authored more than 100 articles. His research interests include cyber–physical systems, cybersecurity and privacy, Internet of Things, edge computing, AI/machine learning, big data analytics, unmanned aircraft systems, connected vehicle, smart and connected health, and wireless communications and networking.

Dr. Song was a recipient of the Best Paper Award from the 12th IEEE International Conference on Cyber, Physical and Social Computing in 2019, the Best Paper Award from the 2nd IEEE International Conference on Industrial Internet in 2019, the Best Paper Award from the 19th Integrated Communication, Navigation and Surveillance Technologies Conference in 2019, the Best Paper Award from the 6th IEEE International Conference on Cloud and Big Data Computing in 2020, and the Best Paper Award from the 15th International Conference on Wireless Algorithms, Systems, and Applications in 2020. His research has been featured by popular news media outlets, including IEEE GlobalSpec's Engineering360, Association for Unmanned Vehicle Systems International, USA Today, U.S. News & World Report, Fox News, Forbes, WFTV, and New Atlas. He has served as an Associate Technical Editor for *IEEE Communications Magazine* (since 2017), an Associate Editor for IEEE INTERNET OF THINGS JOURNAL (since 2020), IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS (since 2021), and IEEE JOURNAL ON MINIATURIZATION FOR AIR AND SPACE SYSTEMS (since 2020). He is a Senior Member of ACM and an ACM Distinguished Speaker.