Received January 16, 2021, accepted March 5, 2021, date of publication March 10, 2021, date of current version March 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3065123

Role of Device Identification and Manufacturer Usage Description in IoT Security: A Survey

NOMAN MAZHAR[1,2], ROSLI SALLEH[1], MUHAMMAD ZEESHAN[3], (Senior Member, IEEE), AND M. MUZAFFAR HAMEED[1]

[1]Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia
[2]Center for Research in Industry 4.0, Faculty of Engineering, University of Malaya, Kuala Lumpur 50603, Malaysia
[3]Department of Computing, School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan

Corresponding authors: Noman Mazhar (wva180032@siswa.um.edu.my) and Rosli Salleh (rosli_salleh@um.edu.my)

This work was supported in part by the RU Grants (under Faculties) of the Malaysia Research University Network (MRUN), University of Malaya, Malaysia, under Grant GPF017D-2019, in part by the Faculty of Computer Science and Information Technology, University of Malaya, and in part by the Center for Research in Industry 4.0, University of Malaya.

ABSTRACT This paper presents an overview of device identification techniques and the Manufacturer Usage Description (MUD) standard used for the Internet of things to reduce the IoT attack surface. The ongoing diversity and the sheer increase in the number of connected IoT devices have crumpled security efforts. There is a need to reconsider and redesign the underlying concept of developing security systems to resolve IoT security challenges. In this backdrop, device profiling and identification have emerged as an exciting technique that helps to reduce IoT device attack surface. One of the known approaches for device identification is to fingerprint a device. There are many ways to fingerprint the device, mostly using device network flows or device local attributes. The device identification ensures the authenticity of the device attached to the network, like user authentication. Since IoT devices mostly work using machine-to-machine (M2M) communication, this requires identifying each device properly. But there is no unified approach for device identification for the ever-growing world of IoT devices and applications. One of the major steps forward in this direction is the development of the Manufacturer Usage Description (MUD) standard that defines the role of a device within the network. It limits the device to execute the primary task only, which will help to reduce the attack surface. Since the inception of MUD, many security frameworks use this standard for IoT security. However, there is a need to scrutinize the security frameworks based on the MUD, to find out the claimed effectiveness of the standard in IoT security. This paper initially identifies and classifies the potential vulnerabilities in IoT devices. Then, the study provides an overview of the research that focuses on device identification techniques and analyzes their role in IoT security. Finally, the research presents an overview of MUD technology, its implementation scenarios, the limitation of the latest MUD standard, and its applications in the industry. The prime aim of this work is to examine the MUD benefits in IoT security along with the weaknesses and challenges while implementing this standard along with future directions.

INDEX TERMS Manufacturer usage description (MUD), Internet of Things (IoT), device identification (DI), software defined network (SDN), machine learning (ML), deep learning (DL).

I. INTRODUCTION

Thereis a sharp hike in security attacks due to the increase of Internet of Things (IoT) applications in our daily lives. Therefore, IoT has been overspreading in the manufacturing and household sectors in recent years. The number of IoT devices is expected to hit nearly 500 devices for each household in 2022 [1]. Within the common infrastructure, these heterogeneous smart devices need to be incorporated into a single secure network. These devices also carry security issues, despite the number of advantages in terms of usability and flexibility [2]. Some of the stats show that there is a huge economic loss in recent years because of the botnets (e.g., Mirai [3]) in IoT devices when deployed in the critical infrastructure. This concern is quite valid for certain scenarios like healthcare (e.g., involving eHealth devices), transportation

The associate editor coordinating the review of this manuscript and approving it for publication was Victor S. Sheng.

and industry as it can compromise users' safety. This incites the need for security in IoT devices for the upcoming IoT era to reduce the attack surface.

Traditionally, identification and authentication of devices rely only on the techniques of cryptographic authentication. IoT systems, however, are typically equipped with limited capabilities. These devices, therefore, use tiny cryptographic keys to be used for cryptographic activities. The use of tiny keys is what makes IoT vulnerable and more easily hacked. IoT devices are specialized equipment manufactured to perform a certain task. Therefore, traditional access control and cryptographic techniques are not enough to secure the device properly specifically due to their limited capabilities.

Device Identification methods using machine learning are used to supplement and operate in tandem with conventional cryptographic authentication techniques to compensate for the problem of limited key size and ensure greater protection of identification and authentication for such devices [4]. One of the ways to reduce the attack surface is by permitting the device access according to its intended use. For this, the device behaviour or profile should be known. There is a lot of work on detecting the IoT device behaviour such as Homonit [5] and SmartAuth [6] that use NLP (Natural Language Processing) and code analysis to analyze the device behaviour if it is according to the design goal or is executing some unauthorized activity. One of the study FlowFence [7] use information flow tracking to ensure legal data access by the users. However, such a solution mainly focus on IoT applications security.

One of the approaches is to keep a set of pre-define behaviours of IoT devices. It is very challenging to define the specifications and enforce this in an environment having a huge number of heterogeneous IoT devices that can communicate in peer to peer mode. To address this, the Internet Engineering Task Force (IETF) develops a new standard known as Manufacturer Usage Description (MUD) [8], a standard that defines the device profile. The profile restricts the device communication to and from other devices by defining a device intended behaviour using the Access Control Lists (ACLs).

MUD become internet standard in March 2019 [6]. The MUD architecture is based on MUD files known as device profiles (provided by the manufacturer or third party) and its communication to the device from the MUD server. The policies defined in the MUD file use JavaScript Object Notation (JSON) [9] for serialization and Yet Another Next Generation (YANG) [10] for modelling device profile. MUD recently received worldwide attention from research communities, industries and standardizing bodies, such as CISCO [11] and National Institute Standards and Technology (NIST) [12]. MUD only define the network access policies but to translate and enforce them in the network is still an open challenge. Software-Defined Networking (SDN) [13] has been considered to address this issue. This technique gave centralized control of the network by decoupling the data and control plane enable the MUD developers to implement it in the network [14], [15].

Cisco has been working on a solution to arm IoT protection using Manufacturer Use Description (MUD). MUD's main goal is to make system visibility and segmentation easier for your network administrators by allowing them to quickly recognise the type of IoT device and define the required behaviours for that device. To do so correctly, the project includes a new participant in the discussion: the manufacturer. IoT system manufacturers will tell us what their products are and what network policies they need for them to operate properly. Customers may use this whitelist assertion to deploy access policies in their networks without having to guess [11].

Device vulnerabilities are the keyless doors for the attackers to intrude in the system. In this context, the study analyzed the IoT device attack vectors, with possible cyber-attacks, as investigated by the Open Web Application Security Project (OWASP) [16]. This research takes an overview of the device identification approach, analyze the security systems using this technique and the methodology adopted to create device profiles. Further, the study enumerates the performance gains of these systems using the MUD approach. Then, the paper, investigate the role of MUD to reduce the device attack surface and to limit the vulnerable devices from compromising the whole network. For this, the research presents an overview of MUD building blocks and its implementations scenario. Further, the study reviews the MUD role in IoT security with a special focus on MUD applications in different domains. Finally, the paper discusses MUD weaknesses in the context of providing end-to-end IoT security along with MUD limitations and extensions proposed by different researchers. The final part summarizes the paper with some discussion on alternative standards and techniques in parallel to MUD along with MUD future research challenges.

A. CONTRIBUTION OF THIS SURVEY ARTICLE

In our previous work [17], the researchers analyse the MUD with intrusion detection systems in terms of effectiveness and performance but in this effort, the study analyzes the MUD implementation details and its challenges. In this research, the study digs deep into MUD architecture to get a thorough understanding of the MUD process and implementation activities. This paper investigates up to date implementations and applications of MUD in the industry as well as its research challenges as explored in the related academic studies. To the best of our knowledge, this is a comprehensive work on MUD applications in the IoT security frameworks and related device identification technologies. The contribution of this paper is as follow:

1) The study presents a summary of IoT vulnerabilities by Identifying and classifying them based on their scope. This will help the researchers to quickly get an overview of the IoT security challenges in general without getting dive down on each IoT architectural layer issue.

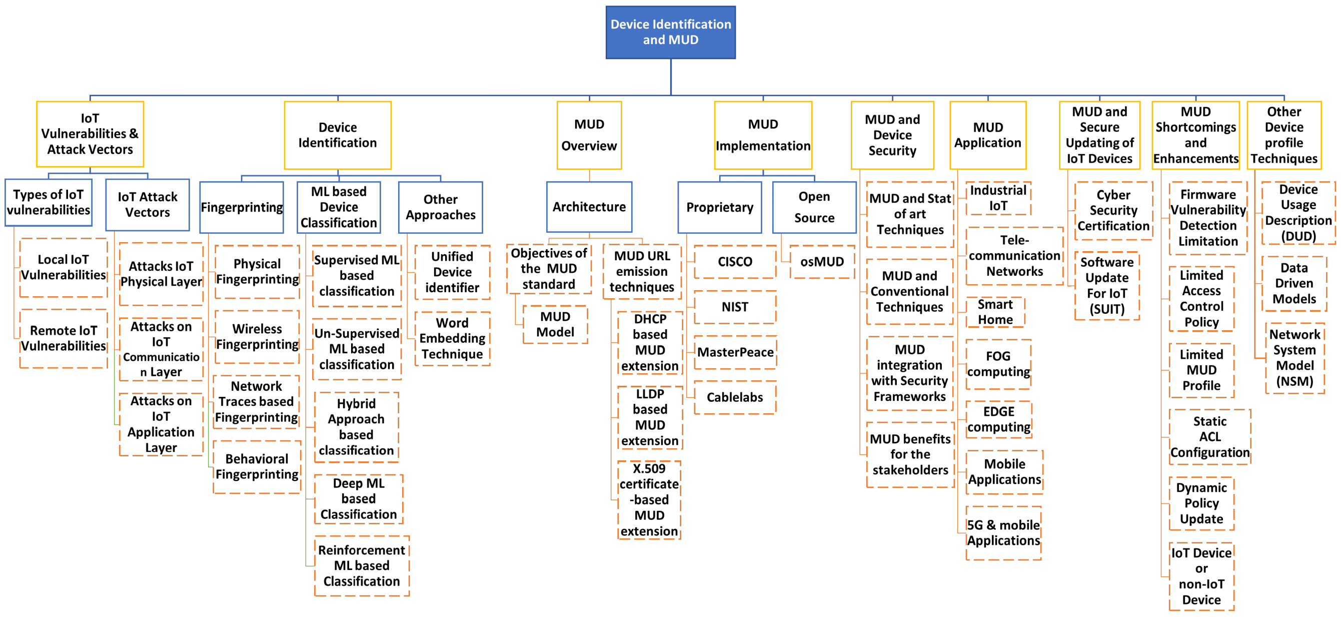
Device Identification and MUD
IoT Vulnerabilities & Attack Vectors
Types of IoT vulnerabilities
Local IoT Vulnerabilities
Remote IoT Vulnerabilities
IoT Attack Vectors
Attacks IoT Physical Layer
Attacks on IoT Communication Layer
Attacks on IoT Application Layer
Device Identification
Fingerprinting
Physical Fingerprinting
Wireless Fingerprinting
Network Traces based Fingerprinting
Behavioral Fingerprinting
ML based Device Classification
Supervised ML based classification
Un-Supervised ML based classification
Hybrid Approach based classification
Deep ML based Classification
Reinforcement ML based Classification
Other Approaches
Unified Device identifier
Word Embedding Technique
MUD Overview
Architecture
Objectives of the MUD standard
MUD Model
MUD URL emission techniques
DHCP based MUD extension
LLDP based MUD extension
X.509 certificate-based MUD extension
MUD Implementation
Proprietary
CISCO
NIST
MasterPeace
Cablelabs
Open Source
osMUD
MUD and Device Security
MUD and State of art Techniques
MUD and Conventional Techniques
MUD integration with Security Frameworks
MUD benefits for the stakeholders
MUD Application
Industrial IoT
Tele-communication Networks
Smart Home
FOG computing
EDGE computing
Mobile Applications
5G & mobile Applications
MUD and Secure Updating of IoT Devices
Cyber Security Certification
Software Update For IoT (SUIT)
MUD Shortcomings and Enhancements
Firmware Vulnerability Detection Limitation
Limited Access Control Policy
Limited MUD Profile
Static ACL Configuration
Dynamic Policy Update
IoT Device or non-IoT Device
Other Device profile Techniques
Device Usage Description (DUD)
Data Driven Models
Network System Model (NSM)

FIGURE 1. **Overall device identification and MUD classification.**

2) The research provide an overview of the Device identification approaches, especially in IoT Security. The paper also discusses the pros and cons of each approach to help the researchers to figure out future challenges in current approaches.
3) This work describes different MUD implementations for industrial applications, this includes proprietary and open-source approaches. This work also provides the pros and cons of each implementation scenario.
4) This survey thoroughly describe the MUD applications in device security along with their pros and cons in the context of each application.
5) The research further provides a detailed discussion on the MUD limitations and proposed extensions for industrial applications and future work in this domain. This helps to light up future research directions and research gaps in this domain.

The complete structure of the paper is explained in Figure 1, showing the taxonomy of IoT vulnerabilities, Device identification and MUD. The remainder of this paper is organized as follows: Section II provides an overview of IoT vulnerabilities. Section III discuss device identification and its role in device security. Section IV shed light on different implementations efforts done in the industry along with their pros and cons. Section V put forward a detail discussion about MUD role in device security. Section VI and VII show the MUD applications in industry and other security frameworks. Section VIII analyzes the MUD limitations and proposed extensions. Section IX highlight other techniques using device profile for device security. Section X outlines some open issues, challenges, and future research directions, and Section XI concludes the paper.

II. IoT VULNERABILITIES AND ATTACK VECTORS

The global IoT market is expected to hit $1.1 trillion by 2026 [18], and it is estimated that there will be 41 billion IoT devices by 2027 [19]. This poses a big challenge for several vulnerabilities already present in these devices. One of the report [20] shows that more than half of IoT devices are vulnerable to cyberattacks. The following section discusses the IoT vulnerabilities in detail.

Open Web Application Security Project (OWASP) [16] describe some of the IoT vulnerabilities to enlighten the manufacturer, developer and consumer, to understand the security risks regarding the IoT devices. As shown in Table 1. The local vulnerabilities are those based on local exploits, concerned with the device internal services and processes. While the remote vulnerabilities are defined as a remote exploit, the one that works on the network services and exploit the security vulnerabilities without getting into the device and seize devices to communicate and accomplish its primary tasks.

A. LOCAL VULNERABILITIES

This is a type of vulnerability, the paper defines, based on the local exploits [21]. The local exploits require access to the device before exploiting any of the vulnerabilities in the compromised device. Also, this may require an alleviated administrated level to execute some vulnerabilities. Some of such vulnerabilities are discussed as follow:

1) WEAK, GUESSABLE, OR HARDCODED PASSWORDS

Most IoT devices are not reconfigured to allow users to change default passwords, especially those that come with web interfaces, and that leaves them vulnerable to a host of password attacks. When the password can be quickly

TABLE 1. IoT vulnerabilities classification [16].

S#	Vulnerabilities	Description	Vulnerability Classification
1	Weak, Guessable, or Hardcoded Passwords	• Use of bruteforced • Backdoors in firmware • Unauthorized access to deployed systems	Local
2	Insecure Network Services	• Insecure network services • Unauthorized remote control	Remote
3	Insecure Ecosystem Interfaces	• Insecure web, backend API, cloud, or mobile interfaces • Lack of authentication or authorization • Lacking or weak encryption • Lack of input and output filtering	Remote
4	Lack of Secure Update Mechanism	• Lack of ability to securely update the device. • Lack of firmware validation • Lack of secure delivery • Lack of anti-rollback mechanisms • Lack of notifications of security changes due to updates	Remote
5	Use of Insecure or Outdated Components	• Use of deprecated or insecure software components and libraries • Insecure customization of operating system platforms • Use of third-party software or hardware components	Local
6	Insufficient Privacy Protection	• User's personal information stored on the device • No access policy and rights	Local
7	Insecure Data Transfer and Storage	• Lack of encryption • Lack of access control of sensitive data • Lack of secure transactions and processing of data	Remote
8	Lack of Device Management	• Lack of asset management • Lack of update management • Lack of secure decommissioning • Lack of systems monitoring and response capabilities	Remote
9	Insecure Default Settings	• Insecure default settings • Limited operators from modifying configurations	Local
10	Lack of Physical Hardening	• Lack of physical hardening measures • Disabling ports • Insecure firmware • Insecure Booting	Local

guessed or brute-forced, why would an attacker waste time in an attempt to circumvent other security controls? Another problem is that some devices don't have passwords, which is a serious weakness when it comes to IoT protection. Although it can make life simpler for remote technicians to embed fixed passwords into smart devices, it also does it for hackers attempting to gain access to your devices or your network. Also, several IoT devices are published with unstable firmware that includes backdoors for debugging purposes to gain access.

2) USE OF INSECURE OR OUTDATED COMPONENTS

The use of outdated software or the referencing of unsafe libraries in code will compromise the overall safety of the product. IoT vulnerabilities include anything that can be used as an entry point or leveraged to perpetuate an attack by introducing weaknesses into the device, from unreliable operating system customizations to the use of insecure third-party hardware or software modules. Furthermore, risks associated with a compromised supply chain could affect the manufacturing process. Besides, risks associated with a compromised supply chain can tamper with the production process early on and remain undetected, and have a significant effect on the device's safety. Supply chain attacks continue to be a major part of the threat environment, with a rise in attacks by 78% in 2018, according to an Internet security threat study by Symantec.

3) INSUFFICIENT PRIVACY PROTECTION

This concerns the unsafe storage, processing, or disclosure of personal data without permission from the user. Cornell University's 2017 study looks at the data that passive observers (such as ISPs) can collect only by monitoring IoT network traffic, even though that traffic is encrypted. Privacy of data, especially when it comes to IoT, is beginning to be addressed through legislative acts. In addition to the above-mentioned issues, the processing of customer data without express consent has been a problem all along. Through collecting and keeping such data, particularly now that IoT is such a huge part of our daily lives, can also lead to a compromise in our physical world security.

4) INSECURE DEFAULT SETTINGS

On a smartphone, the default passwords or system settings are also vulnerable. Although it is often just stupidity on the client part that users do not change default settings, device settings such as hardcoded passwords, exposed services running with root permissions cannot be altered at other times, etc. Fortunately, these insecure activities are being fought by some lawmakers. California, for instance, has a law requiring manufacturers of IoT devices to set specific pre-programmed passwords or enable users to update their passwords before using the devices.

5) LACK OF PHYSICAL HARDENING:

Hardening the device against physical attacks protects it from attempts to steal confidential information by malicious users that can later be leveraged to initiate a remote hack or gain control of the device. For example:

a) Debug ports that are usually not removed or disabled leave your devices vulnerable to access by hackers.
b) Simply removing a memory card to read its content can reveal passwords or other sensitive data.
c) Using secure boot helps validate firmware and ensures that only trusted software can run on the device.

B. REMOTE VULNERABILITIES

Remote exploits become the basis for the classification of remote vulnerabilities [22]. In remote exploits, the attacker does not require access to the device, rather only required to exploit the security vulnerabilities over the communication medium or at the remote hosts or servers. Some of the vulnerabilities under this class are as follow:

1) INSECURE NETWORK SERVICES

Insecure network services are also a serious vulnerability. Even as IoT devices come into action, network protection tools such as firewalls, intrusion detection system/intrusion prevention systems (IDS/IPS), unified threat management solutions (UTMs), etc., remain important. Due to unauthorized access (due to default passwords, open ports, etc.), IoT protection has also been breached and can potentially lead to these devices being used as part of a larger botnet. To execute threats such as a distributed denial of service (DDoS) attacks on targeted websites or network resources, botnets are also used.

2) INSECURE ECOSYSTEM INTERFACES

On the OWASP Top 10 2014 chart, it was previously split into three categories: vulnerable network, cloud, and mobile app. Interfaces such as the web, cloud, smartphone, or back-end APIs that allow you to communicate with the smart device may have (or worse, a total lack of) vulnerabilities in authentication/authorization implementation, encryption weaknesses, data filtering, etc. These security vulnerabilities could eventually lead to the system or some of its related components being compromised.

3) LACK OF SECURE UPDATE MECHANISMS

The problem here is that the ability to safely update many IoT devices is missing. This is an environment where producers of electronics should step up their game. For example, in the United Kingdom, a recently proposed law will make it possible for IoT device manufacturers to have a minimum amount of time within which security updates would be received by their devices. Update processes, however, are not all about fixes deployed and vulnerabilities closed.

4) INSECURE DATA TRANSFER AND STORAGE

Nowadays, it may seem obvious to preserve data protection with experts constantly reminding us about encryption, data classification, and careful handling of confidential information, but it's no wonder that this paper still talking about it given all the data breaches this research mention still see in the headlines daily. In addition to limiting access to confidential data in general, ensuring data is encrypted at rest, in transit, or during the processing stage is essential. When encryption is not strictly enforced, if it is absent from your smart devices, it leaves data vulnerable and becomes an IoT security issue.

5) LACK OF DEVICE MANAGEMENT

Just as it is vital to know what assets are on your network, it's equally important to handle them efficiently. If they communicate with the network and have access to it, regardless of the size of the devices or their costs, then methodically handling them should be one of the primary concerns. An integral part of the process should be to participate in network security best practices to upgrade management to safe decommissioning, device monitoring, etc. The entire network can be compromised by failing to handle your IoT devices effectively (such as relying on old methods such as asset monitoring using Excel spreadsheets).

C. IoT ATTACK VECTORS

The OWASP has published a detailed draft regarding the attack surfaces of IoT, these are the areas in IoT systems and applications that are vulnerable and prone to threats. Figure 2 shows a summary of these attack surface areas [23].

1) ATTACKS ON PHYSICAL LAYER

The compromised IoT devices are the primary sources from where the attack can be initiated. The main parts of the device like memory, firmware, physical interface, web interface, and network surfaces are vulnerable. The attackers can further exploit the default setting, old components, and insecure update mechanism.

An attacker could eavesdrop the communication by using various tools to catch network traffic between components of the IoT, in which case the attacker could analyze network traffic and identify the type, status, unique identifiers, and operating system of the user devices being used [24], this type of attack is known as *Eavesdropping*. It is very straightforward for the attacker to sniff out the sensitive information such as passwords or any other data flowing from [25], [26] tag-to-reader or reader-to-tag due to the wireless characteristics of the RFID, which renders it vulnerable because

TABLE 2. IoT cyber attacks classification on each layer.

Layer	Attacks	Reference
Physical Layer	Eavesdropping	[24] [25] [26] [27]
	RF Jamming	[27]
	Spoofing	[28] [25]
	Unauthorized Access to tags	[29]
	Tag Clone	[30]
	Natural disasters and environmental threats	[31]
	Human-caused physical threats	[31]
Communication Layer	Jamming attack	[32] [33]
	Selective forwarding attacks	[34] [35] [36] [37]
	Sinkhole Attack	[34] [36] [36]
	Sybil attack	[38] [39]
	Denial of sleep attack	[40]
	Wormhole attacks	[41] [34]
	Man-in-the-middle attack	[42] [43] [44] [45]
	Denial of Service attack	[46] [47]
	Malicious Code Injection	[48]
Application Layer	Social engineering	[49] [50] [51]
	Buffer overflow	[52] [53]
	Sniffing attack	[54] [25]
	Spear phishing attack	[55] [56]
	Backdoor attacks	[51]

the attacker can make it accessible in despicable ways [27]. Another technique is *Spoofing*, when an intruder can listen to network traffic and recognize the MAC address of a network privileged device and transmits false information for example like in the RFID systems and mistakes its originality, making it appear from the source [28]. The attacker gets complete access to the device in this way, rendering it vulnerable [25].

Another type of attack is known as *RF Jamming*, RFID tags may also be compromised by a kind of DoS attack in which an excess of noise signals interrupts communication through RF signals [27]. This is quite common and can easily be executed using simple devices. It can cause serious situations where RFID is used for touch-and-go scenarios. Also, the Tags can be accessed by anyone without authorization due to the lack of proper authentication mechanisms in a large number of RFID systems categorized as *Unauthorized Access to tags attacks*. The attacker is unable to only read the data, but it is also possible to alter or even delete the data [29]. Similarly, Since tags are deployed on various items that are visible and their details can be read and changed with certain hacking techniques, any cybercriminal who can build a copy of the tag can easily catch them and thus compromise them in a way that the reader can not differentiate between the original and the compromised tag [30], termed as *Tag Cloning*.

The physical infrastructure of the IoT networks may be disrupted by natural disasters such as tornadoes, hurricanes, earthquakes, ice storms, lightning, and floods, known as *Natural disasters and environmental threats*. Environmental risks such as excessive temperature and humidity values, water accidents (e.g. electrical short circuits), fires, chemical accidents, and living organism infestations (e.g. insects, rodents) may also cause severe IoT network damage. Consequently, this sort of hazard results in the destruction of resources, rendering their availability unlikely. It is possible to describe the effect of these threats as ''Doomsday''; however, their likelihood is ''rare'' since such phenomena are very rare and there are current security measures that can recognize and mitigate them [31]. On the same lines, compared to the aforementioned natural disasters and environmental risks, *human-caused physical threats* are more difficult to handle because they are deliberately designed to circumvent security measures and, at the same time, threaten the most vulnerable physical infrastructure. This category covers eavesdropping, vandalism, computer tampering, and misuse. All the above security criteria can be affected by this form of threat. [31].

2) ATTACK IoT COMMUNICATION CHANNEL

Communication channels are another area of security concern. The channel connects the IoT devices and the outside world. The protocol used for communication in the IoT networks has security issues and can affect the entire system. IoT systems get vulnerable to attacks like denial of the service (DoS) and spoofing.

A *jamming attack* prevents the nodes that occupy the communication channel from communicating with each other. This form of attack can be divided more precisely into four categories: persistent jamming, deceptive jamming, random jamming, and reactive jamming [32], [33]. In general, the MAC protocol allows approved nodes to send packets only if they do not use the required communication channel. In the case of a constant jamming attack, however, the attacker tries to use the communication channel continuously by transmitting a radio signal. Therefore, the channel should not be used by legal nodes. On the other side, the intruder continuously sends packets to the communication channel without any delay in the deceptive jamming attack. Therefore, a valid node is forced to stay in receiving mode during the attack because it assumes there are remaining packets to receive. The random jamming model aims to consider the conservation of energy. The at-tacker specifically can function either in a non-active state or in a jamming state. One of the previous two models is based on the function of the jamming state. The reactive jamming model uses an alternative technique, in contrast to the previous ones, in which it acts in a quiet mode when the communication channels are not used.

Some of the attacks are based on network traffic like, some malicious nodes refuse to send any packets in *Selective forwarding attacks* to break the routing paths of the network [34], [35]. There are different forms of these attacks. A typical case is the black hole attack, in which each packet is rejected by the malicious node and does not forward any of them [36]. The Neglect and Greed attack, in which the attacker drops some packets or segments of them, is another type [37]. Such attacks are aimed at destroying the availability of data and services. The purpose of the malicious nodes in *Sinkhole Attack* is to direct the network traffic to a particular node. Typically, they promote a specific network path and attract other nodes to use this route [34], [36]. This form of attack does not cause serious harm to the functionality of the network, but when combined with other attacks, it may be destructive [34]. As in the previous case, the availability of systems is also the object of these assaults. The compromised nodes forge or produce multiple identities in the *Sybil attack* to deceive other nodes [38], [39]. In this case, the attacker aims to take control of various areas of the network, without any physical node being used. This attack can be categorized in more detail into three types: SA-1, SA-2, and SA-3 [38]. To undermine the authenticity and availability of the systems and facilities is the aim of these attacks. The malicious nodes generate a direct communication link in *Wormhole attacks*, which is used to forward network traffic data ignoring intermediate nodes [41]. The name of this communication channel is worm-hole and is distinguished by excellent network metrics such as high throughput. To create a wormhole, two collaborative nodes are typically needed. It should be noted that such a link without malicious intentions may be used for specific important reasons [34]. However, it poses a significant threat if paired with other network attacks, such as a sinkhole attack or a Sybil attack.

The IoT nodes normally use batteries with a limited lifetime. They use sleep cycles to expend their lifetime. The attacker manipulates their sleep function result in the continuous working mode this shortens their battery lifetime and the nodes start to shut down [40], the attack is known as *Denial of sleep attack*. This attack-type *Man-in-the-middle attack* is described as a form of eavesdropping in which correspondence messages exchanged between two parties can be illegally monitored by the intruder. Neighbor Discovery Protocol (ND or NDP) poisoning [42], [43], Address Resolution Protocol (ARP) poisoning [57], [58], replay assaults [44], [45], session hijacking [59], [60] and malicious proxy servers [61], [62] are examples of these attacks. These attacks challenge the security and authenticity of the systems at the same time. The *Denial of Service attack* attacks [46] are aimed at rendering computing systems usable and can be carried out in all layers of the proposed IoT stack. In particular, by undermining the computational infrastructure that sustains them, they aim to impede legitimate organizations from using applications or services. For example, they try to reduce the Central Processing Unit (CPU) output or flood the memory size [47]. Flooding attacks, distributed DoS attacks (DDoS), reflection attacks, amplification attacks, and jamming attacks that have been addressed before are some examples of this type of attack. Another serious attack is *Malicious Code Injection*, the attacker injects the malicious code in one of the nodes which causes the complete system shutdown, or the attacker gets full control of the system [48].

3) ATTACK ON APPLICATION SOFTWARE

In this type, the vulnerabilities of web applications and software used in IoT systems can become a great cause of a compromised system. The web application can steal user data and insert malicious updates into the system. To defend these challenges, many techniques have been developed and are being used in the industry, intrusion detection, and prevention system is one of them.

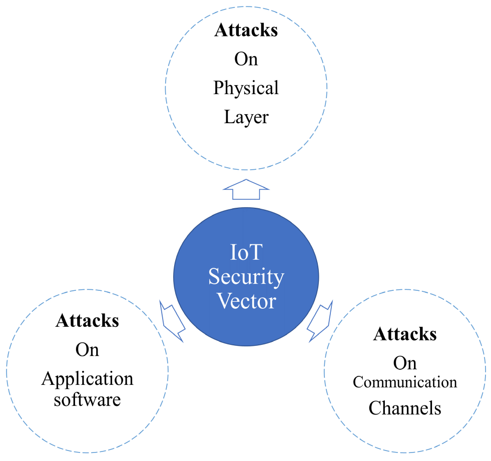
Attacks On Physical Layer
IoT Security Vector
Attacks On Application software
Attacks On Communication Channels

FIGURE 2. Taxonomy of IoT security.

A psychological assault aimed at misleading users to reveal sensitive information or conduct specific malicious activities unintentionally is classified as *social engineering* [49], [50]. The powerful social engineering technique, instant messages, or domain name system (DNS) spoofing processes is the phishing assault in which the attacker attempts to gain the attention of the user by using spoofed emails. Usually, users are led to a bogus website that urges confidential data to be inserted. Spear phishing is a more dangerous variant of this form of attack. The attacker has extensively analyzed the recipients in this case, and each fake message is carefully designed to fit the recipient profile [51]. These attacks concentrate primarily on knowledge confidentiality, honesty, and authenticity.

A *buffer overflow or buffer overflow or buffer overrun* is a form of attack that enables the attacker to inject more data into a buffer than the capacity limit allows, according to NIST. To inject malicious code that will make it possible to manipulate the overall system [52], [53], the attacker attempts to overwrite existing information in the buffer. Stack overflow, global data area overflow, overflow of format strings, heap overflow, and integer overflow are some examples of these assaults. The attacker usually uses the assembly code to execute such an attack. These attacks are aimed at undermining systems' credibility and authenticity.

When an attacker implements sniffing into the system in the form of a sniffing application that, in turn, gains network information resulting in system corruption is known as *sniffing attack*. Sniffing can be found in [54] ARP poisoning, DHCP attack, MAC flooding, and password sniffing are classified into DNS poisoning. On the data link layer, sniffers start their sniffing work. If the sniffer is the data link layer, the other upper-layer is involved in the sniffing process. In this attack, the attacker sometimes installs some sniffer application in the system to get the network information which results in system corruption [25].

Phishing is a threat to online security that has existed for over two decades. PhishTank describes phishing as a fraudulent attempt to steal personal information, primarily via email. Phishing is also associated with the theft of personal information related to financial systems, such as passwords for online banking or email passwords [55]. In *Spear phishing attack* the attacker traps the user by sending him an alluring email. When the user opens the email, the attacker gets access to some credential of the user and can get sensitive information [56].

Similarly, A *Backdoor attacks* or otherwise trapdoor is a software code section that allows an attacker to bypass particular processes that may have security controls [51]. More specifically, when the user em-ploys specific credentials or a specific sequence of events is executed, a backdoor is triggered. Interestingly, a backdoor is not inherently a security threat, since it can be used by a system administrator to bypass time-consuming procedures and easily monitor the software's features. However, if an attacker is aware of the particular code block, extremely negative effects may be induced. Typically, malicious backdoors serve as a network service that allows the attacker to connect to an unusual port on the network and perform malicious activities. Confidentiality, confidentiality, and authenticity of data are typically the objective of backdoors.

III. DEVICE IDENTIFICATION (DI)

The increase in the number of heterogeneous IoT devices proportionates the sharp rise in the complexity and size of the network. This causes more unknown vulnerabilities and security loops in the networks. One of the ways to reduce the vulnerabilities and attack surface of the devices is by using well-designed identification and authentification mechanism. This design pattern will help the IoT devices to locate the malicious IoT devices in the network [63]. IoT identification is not simple rather more challenging due to the heterogeneous type of IoT devices, communication protocols, manufacturers, and control interfaces. Traditionally device identification was done using cryptographic techniques.

As the IoT devices are resource constraint, conventional cryptography is not suitable, however, a lightweight cryptographic technique has been used. This technique uses a small key size that is more vulnerable and can be easily compromised. Machine learning is used in the identification techniques in IoT devices along with cryptographic techniques to minimize address the issues relating to the key size problem. Device identification involves two methods *fingerprinting* and *classification*.

A. DEVICE FINGERPRINTING

The device should have a unique fingerprint to be used by the ML algorithms for the identification process. There are many ways to get these fingerprints as discussed below:

1) PHYSICAL FINGERPRINTING

One of the technique is Physical Unclonable Functions (PUFs) for authentication of IoT devices at the hardware level. By exploiting the inherent random variations of the physical (sub-)microscopic structure of an integrated circuit [64], PUFs provide IoT devices with a specific hardware fingerprint. Another approach is, Remote fingerprinting of a physical device, as opposed to an operating system or device type, without the known cooperation of the fingerprinted device is an important technique used in computer forensics [65]. By manipulating microscopic variations in system hardware, the technique achieves this objective to varying degrees of precision: clock skews. There is no change to the fingerprinted devices needed by this technique. It records accurate measurements when the meter is thousands of miles away from the fingerprinted device, several hops, and tens of milliseconds away, and when the fingerprinted device is connected to the Internet from various locations and access technologies. Various types of this technique may be applied, such as passive and semi-passive techniques, when the fingerprinted device is behind a NAT or firewall and the machine time of the device is preserved through NTP [66] or SNTP [67].

2) WIRELESS FINGERPRINTING

This approach utilizes the characteristics of the wireless channel between two entities to produce ‘‘wireless fingerprints’’ which are then used to provide data provenance. To define the wireless link between two individuals, it uses the Link Quality Indicator (LQI) values. The concept behind developing wireless fingerprints is that the wireless channel is intrinsically symmetric between two interacting entities. However, if one of two transmitting entities travels more than half of a wavelength, according to Jake's fading model [68], then the wireless channel decorrelates rapidly and becomes independent for a distance exceeding one wavelength. To extract protection primitives from wireless channel characteristics, this fact and the reciprocity property of electromagnetic wave propagation are used [69]. This Fingerprint uses the protocols and interfaces used in the wireless domain. Based on certain wireless attributes the fingerprinting is established [70], [71].

3) NETWORK TRACES BASED FINGERPRINTING

This method belongs to another class of strategies, those that seek to identify network traffic based solely on the flows' statistical characteristics. The main concept behind this technique is that it should be sufficient to determine the size of the IP packets, their inter-arrival time, and the order in which they are seen at the classifier to determine the device that created the traffic. The systems based on this technique will dynamically classify flows as packets move through the classifier, determining whether a flow belongs to a given device, or created by an ‘‘unknown’’ (i.e., non-fingerprinted) device. [63], [72] [73].

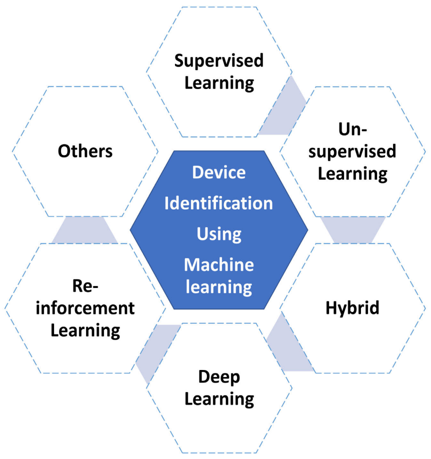

FIGURE 3. **Machine learning techniques for device identification.**

4) BEHAVIOURIAL FINGERPRINTING

To recognize devices, a passive behavioral fingerprinting technique is used. Extracting features from a series, a collection of packets ordered by time-stamps produces the fingerprint. To create a unique fingerprint for each device, these features are extracted from the header and payload of the network packets. Such behavioral and flow-based fingerprints can be used as a basis to monitor the actions of the device continuously when connected to the network [74].

The next section discussed some of the Machine learning approaches used for the identification of the IoT devices

B. DEVICE CLASSIFICATION USING MACHINE LEARNING (ML)

Device identification is a multiclass classification problem [75]. As the process involves the identification of a different class of devices and their instances. The classification process is required to identify the class of the device. Many classifiers can be used to accomplish this task using Supervised, unsupervised, hybrid, reinforcement, and deep learning techniques as shown in Figure3.

1) DI AND SUPERVISED ML

Supervised learning has been the most common technique used for the identification of IoT devices. Some of the work [76] used this technique to distinguish IoT and non-IoT devices. The algorithm works on the network traffic and analyzes incoming and outgoing packets ratio along with the time to live attributes. Other studies [77] detect the IoT devices not mentioned in the whitelist. For this purpose, the research uses the supervised machine learning algorithm known as random forest to analyze the network traffic from the devices included in the whitelist to train the algorithm. Also, the algorithm can detect the other IoT devices automatically not given in whitelist.

One of the interesting concepts [63] is a brownfield approach, this concept based on the concept of coexisting with the compromised IoT devices in a network. The approach ensures that the compromised legacy devices do not endanger other devices in the network. The system achieves this by automatically detecting the IoT devices and by enforcing communication policies to such devices to restrict their role in the network. The system uses SDN to isolate and filtering of network traffic. For device identification, the system uses the device fingerprints during its initial communication to determine the type and the model of the device.

Some approach [78] uses a genetic algorithm (GA) to extract the features from the packet and apply multiple machine learning algorithms like decision tree, decision table, oneR, and PART for the classification of the device giving the set of features to them. The passive behavior fingerprinting technique [74] extracts the features from the sequence of packets as compared to a single packet to fingerprint a device. The system uses machine learning algorithms to automatically identify the whitelisted device type. Also, the study proposes a security model to enforce the policies for the IoT device in the network, detect the malicious device, and restricts their communication in the network. Random forest [75], [79] used for the classification of the device that is further compared to other machine learning algorithms. The random forest is compared to some of the deep learning algorithms, still, it comes out to be the better choice. Even for the encrypted traffic the identification rate is 86-99% [80].

The concept of a centralized machine learning algorithm is not a scalable solution neither effective in the detection of new devices. Distributed fingerprinting (DEFT) use multiple classifiers for fingerprinting the devices also the gateways near the devices help to classify them. The controllers and gateways enable the system to detect new devices in the network. The system uses SDN and NFV for the detection of new devices and shows a very low false-positive rate and high accuracy [81].

Open Issues: Device identification and whitelisting in large-scale enterprises require more traffic data to train models, identify multiple vendors per device type, and multiclass classification, as opposed to binary classification citeRN3683. There is a need to undertake fine-grained characterization and classification of IoT devices in a smart environment such as a home, city, campus, or enterprise [73]. Encrypted traffic is a big challenge for IoT identification. The system identification technique gets compromised when encrypted traffic is used for device authentication further the device firmware updates also produces different fingerprints causing issues to the system. (ML supervised algorithm random forest). Genetic algorithms have limitation for dealing with a complex feature set and machine learning algorithms used for classification increases the processing time and complexity of the system in the context of resource constraint devices. Privacy is an again big challenge especially when the cloud services are used for the classification of device identification [82]. Network fingerprinting does not work below the network layers in the TCP/IP stack. While there are solutions that perform fingerprinting using the radio signals [83].

2) DI AND UNSUPERVISED ML

Training classification models take a lot of time and human effort to train the model from labeled data, due to the enormous number of IoT devices from different vendors. In this context unsupervised machine learning can be useful for the A model that works autonomously to detect the device identity by analyzing the network traffic. In this context, a system autonomous device identification (AuDI) [84] has been proposed. The system uses passive fingerprinting using the network traffic and does not require labeled training data nor prior knowledge of the device. The system is not limited to the training data like the prior solutions, it can scale up to the new already available IoT devices present in the market. Further, the system can detect the device in any mode, standby mode or operator interaction mode, or normal activity mode. The feature extraction is done by the unsupervised machine learning algorithm. The fingerprints obtained from the features are sent to the cloud service for the classification of the device.

Open Issues: For classification, the unsupervised machine learning approach uses unlabeled data. The most frequently used unsupervised machine learning technique is clustering [85], in which instances in the unlabeled dataset are clustered into separate clusters based on their similarity. Instances in the same cluster are known as having similar characteristics or properties and are graded in the same class. Because of the manual assignments of cluster number, it provides lower accuracy results for classification as the downside of unsupervised learning. Nevertheless, in terms of detecting new attack instances, it has better performance than supervised machine learning, and it is also considered to be more efficient for IoT attack detection [86]. As claimed the system is autonomous, but still requires offline training time. This reduces the effectiveness of the system for real-time applications. Also, the use of cloud services further imposes delay and complexity for resource-constrained devices. Device privacy and security become a challenge for the system as it uses cloud services.

3) DI AND SEMI-SUPERVISED APPROACH

The approach to semi-supervised machine learning blends supervised and unsupervised approaches to machine learning. For preparation, it uses both labeled and unlabeled data and extracts the intra-structure of the dataset [87]. The semi-supervised approach uses supervised and unsupervised machine learning for the detection of seen and unseen devices to reduce overhead. As one of the work [88] can detect the unseen devices connected to the network, as compared to the other research, where the only supervised approach has been dominated for the device identification. The system uses the Random Forest algorithm for training and classifying the data, and Ordering points to identify the clustering structure (OPTICS) is used for unsupervised learning. Also, semi-supervised learning is used to distinguish various devices, the model takes input features including time interval features, traffic volume features, protocol features, and related TLS (Transport Layer Security) features. The framework allows the converted features to form a single, compact cluster per class, which can be segregated directly, to solve the problem of feature fluctuation better. Finally, the model differentiates IoT and non-IoT to minimize the effect of non-IoT devices and distinguishes particular IoT devices based on multi-task learning [89].

Open Issues: Many practitioners indeed make assumptions about the relationship between labels and the unlabeled distribution of data when using semi-supervised learning in practice (e.g. cluster assumption). Yet, this can lead to undesirable circumstances. How to reduce the risks remains to be explored for future studies. Relaxing deep assumptions in semi-supervised learning, thus maintaining substantially improved sample complexity [90]. If the algorithm is transductive then it assigns labels to the unlabeled data in the training dataset. If the algorithm is inductive then it assigns labels to the unlabeled data in the training and test set [91].

4) DI AND DEEP ML

As the device identification, involves feature extraction for fingerprinting and then classify them to identify the device, but with deep learning, no such traffic engineering is required. Also no need to label data and the process of device identification become straightforward and autonomous. The passive behavior techniques like the inter-arrival time (IAT) between packets used in fingerprinting to determine the device type, when used with a deep learning approach become more efficient. Conventional work uses statistical analysis to analyze IAT to identify the device type, but [92] plot graphs for the IAT packets, as IAT is unique for each device, thus graphs become a more efficient way to determine the type of device. Sometimes probabilistic models [93] are used for the device class identification like IoT or non-IoT class. The stacked encoders help to find the features from the network flow without the need for the labeled data. This model also helps in determining IoT or non-IoT classes. However multi-class identification is also possible in this [94] the system proposes a multiclass classifier for the device identification.. the classifier is trained using the public set of data to detect known devices, then the classifier is trained with the unauthorized devices to detect and identify the unseen devices. The system does not follow the rule-based or device behavior features for the detection of the device, rather it uses a novel representation or feature learning approach.

Open Issues: The well-known issue with deep networks is that they are difficult to train, can collapse into local extremes, take a long time, and require powerful computational resources, e.g. GPU, and the network is not flexible/adaptive to various new data once trained [95]. Since DL models require a large portion of resources, such as processors, battery power, and memory, DL methods can hardly be used in IoT and resource-constrained devices for training purposes [96]. The deep learning approach for feature extraction poses some limitations over the system regarding encrypted network traffic. Some deep learning models like

Bayesian Model cannot predict the unknown category that is not present in the training data, this limits the ability of the model to predict unknown devices [97]. Most of the models and systems work on the TCP/IP network stack only. The other layers in the stack are not much discovered for device identification purposes.

5) DI AND REINFORCEMNET ML

Reinforcement learning is a continuous process of having reward and punishment, best suited for complex and unforeseen scenarios like in the case of device identification for IoT. Device identification is a prerequisite for IoT security. The brand and manufacturer can be known from the protocol banner, but for fine-grained device information multiple protocols need to be analyzed, this can cause an overhead for resource constraint devices. As using multi-protocol probe scheme is quite efficient in the context of time and resources [4]. The model uses reinforcement learning and the Markov decision process for the banner-based device identification process shows high accuracy. Another model uses a Q-learning approach to dynamically learn the neighboring devices and select the best system in the D2D communication network for immediate and improved connectivity. The algorithm runs on each network device, collects data from its neighboring devices, and adjusts the latency-based communication range, and selects the best device to be connected to. The model also identifies the devices in the D2D communication [98].

Open Issues: The biggest challenge with reinforcement learning is its data-hungry nature thus require a lot of data. To get output from this data requires more computing power and resources and hence not suitable for resource constraint IoT devices. Further, the output of the model is not always correct, which possess its limitation to be used in critical applications like health and transport where safety is a priority [99]. The performance of security systems when used with device fingerprinting along with the machine learning approach against the attacks is shown in the following Table 3.

C. OTHER APPROACHES

Some of the approaches consider unifying the identity of devices among different platforms.

1) UNIFIED DEVICE IDENTIFIER

One of the ways is to unify all the device identification schemes by different vendors in one standard. One of the efforts in this regard [100] shows that it is difficult to identify the IoT device in different IoT platforms, as each platform uses a different IoT identification technique. So it is important to work on the interoperability of device identification among different IoT platforms. The paper proposes a device name system (DNS) that translate different device ID formats (i.e., oneM2M, GS1 'Oliot', IBM 'Watson IoT', and OCF 'IoTivity') to the oneM2M device IoT format. The [101] Implements this system on a microcomputer. And the results show how successfully the oneM2M requests the resources from the Watson IoT and FIWARE sensors.

2) WORD EMBEDDING TECHNIQUE

Another technique [102] similar to word2Vec, the word embedding technique, known as IoT2Vec. This model gets the embeddings from a dataset and then apply it on the device usage data to identify similar device type. The technique determines the IoT devices efficiently by identifying the usage of pattern similarities.

Open Issues: There is a need for a common standardized body for issuing the device ID globally and a standard for manufacturers to build the device for the standard task accordingly [100]. The authentication and security issues become serious when the users want to use the services from other platforms that require a remote connection. This requires the remote connections and devices should be authenticated to ensure the privacy of the network.

IV. MUD OVERVIEW

Internet Engineering Task Force (IETF) started to work on the MUD standard in Jan.2016 and develop the internet standard on Mar. 2019 [8]. The goal of this standard is to enable the devices to signal the network regarding the kind of access and network functionality for the device to work properly. The main intent of MUD is [8]:

1) To reduce the attack surface area on the device by defining the intended device behavior.
2) Address the scalability issue for the network policies for heterogeneous devices in the network.
3) To create minimum deterrence for the devices against the attacks till the system updates.
4) Provide a cost-effective solution.
5) Develop an extendable solution so that the manufacturer can add on other device capabilities.

In MUD manufacturers are responsible for defining device behavioral profile instead of the administrator. However, MUD architecture allows to automate the network access policies defined by the manufacturer but it is important to note that the initiation of these profiles is network domain-dependent containing the devices. YANG [10] is used for defining the MUD profile model while the JSON [9] is used for the serialization of the model in the MUD file. The next section discusses MUD architecture and the components involved in it.

A. MUD ARCHITECTURE

The key components of the MUD architecture are MUD file, MUD file server, MUD manager, MUD URL, Thing, and Manufacturer as shown and explained in Table 4.
The MUD process starts when the things send the MUD URL to the switch or router, which further sends the URL to the MUD manager. The standard defines multiple approaches for URL processing like DHCP, LLDP, and X.509. The role of the MUD manager is to receive the MUD URL and get the

TABLE 3. Performance analysis of machine learning and fingerprinting techniques.

Ref#	Machine Learning Techniques					Fingerprinting technique					Accuracy
	Supervised Learning	unSupervised Learning	Hybrid Learning	Deep Learning	Reinforcement Learning	Physical	Wireless	Network	Behaviour	Other	
[76]	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	99.281%
[77]	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	99%
[63]	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	81.5%
[78]	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	95%
[74]	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	90.3%
[79]	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	94.5%
[75]	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	97%
[80]	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	99%
[81]	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	70.55%
[84]	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	98.3%
[88]	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	91.2%
[92]	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	86.7%
[93]	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	70%
[94]	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	97.87%
[4]	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	92.3%

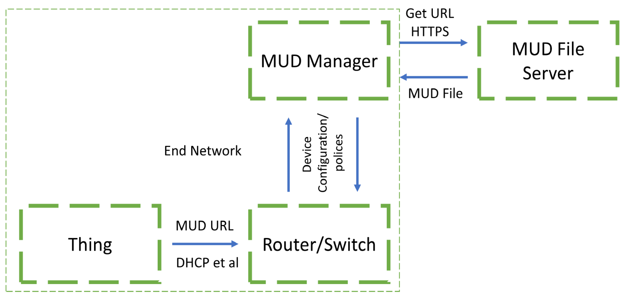

FIGURE 4. MUD architecture [8].

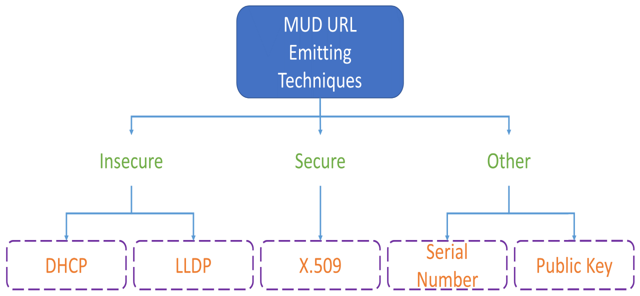

FIGURE 5. MUD URL emitting techniques.

MUD file from the MUD server. Also, interpret the MUD file and convert it to the network configuration. Also, update the configuration on the specific network device. The information returned by the MUD server is only valid till the time the thing is connected to the network. In case if the MUD manager detected that the file for the thing has been changed then the MUD manager is responsible for the change update in the network [8].

B. MUD MODEL

MUD files are consist of network policies like Access Control Lists (ACLs). It defines the restrictions on the device to whom it can communicate like talking to the device from the same manufacturer, also communicating to a specific service like DNS, or allowing and denying to access a specific port. The MUD model uses YANG [10] standard while for serialization and translation in the file it uses JSON [9].

C. MUD URL EMITTING TECHNIQUES

MUD uses a universal resource locator (URL) for a couple of purposes. First, it helps to classify the type of device. Secondly, it allows the network to locate the device description for the device. To emit and discover the MUD URL there are several ways as shown in Figure 5. Secure X.509 certificates are one of the ways, for secure URL transmission. MUD uses X.509 non-critical certificate extension to transport MUD URL. Again there is a need to communicate this certificate among the devices. Tunnel Extensible Authentication Approach (TEAP) [103] a secure protocol is used in this approach.

Other insecure techniques are Dynamic Host Configuration Protocol (DHCP) [104] and Link Layer Discovery Protocol (LLDP) [105]. In the context of DHCP, the DHCP client sends the request to the DHCP server. The DHCP server is responsible for giving an appropriate response. It depends if the MUD manager also embeds a DHCP server in one entity. In this case, the DHCP server will also act as a MUD manager and process the MUD URL. This obviates the need to contact the controller. Another option is LLDP. This option works on the data link layer bring down the communication complexity to layer 2 as compare to DHCP that works on the application layer of the TCP/IP stack.

V. MUD IMPLEMENTATIONS

A. MUD BUILDING BLOCKS

There are three building blocks for the MUD as shown in Figure 6. The first one is the device description universal

TABLE 4. Main MUD architecture components [8].

Component	Description
MUD file	A file defines the network behavior for the device
MUD file server	MUD file hosted by a web serve
MUD manager	Responsible for requesting and receiving the MUD file along with further sending instructions to enforce the policy to the respective network devices.
MUD URL	A resource locator string that is used by the MUD manager to acquire the MUD file from the server
Thing	Any device able to send MUD URL
Manufacturer	The entity responsible for the configuration of the device to send the MUD URL and also develop a MUD file for each device. The manufacturer can be an original equipment manufacturer (OEM) or any other third party.

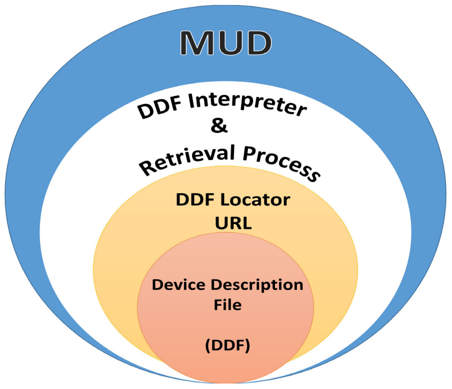
MUD
DDF Interpreter & Retrieval Process
DDF Locator URL
Device Description File (DDF)

FIGURE 6. MUD building blocks.

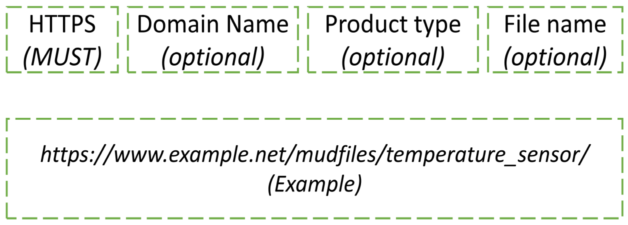

FIGURE 7. MUD URL scheme.

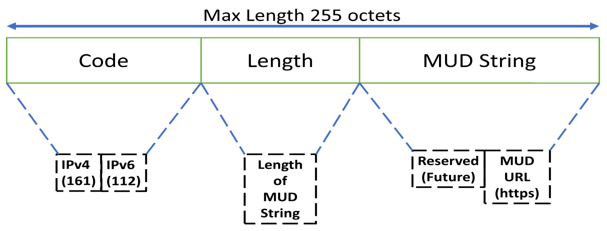

FIGURE 8. MUD DHCP extension option.

resource locator (URL); locates the device description artifact over the network. The second is the device description or MUD file itself, this includes file model and serialization methods. In this case, the MUD file uses YANG and JSON respectively. And the third part is the process of getting the MUD file for each device from the web server and file interpretation and processing mechanism. There are multiple MUD URL emitting techniques that are described in the next section in detail, further MUD implementation proof of concepts efforts by different vendors are discussed in the following sections.

B. MUD UNIVERSAL RESOURCE LOCATOR (URL)

The first part of the MUD architecture is the resource locator for the device description. The URL can be used to locate the description location within the local network or remote server. However, MUD URLs are required to use an "HTTPS" scheme. The format of the MUD URL is shown in Figure 7. The format of the MUD URL starts with the hypertext transfer protocol (HTTP) over transport layer security (TLS) known as HTTPS [106]. After that the domain name hosting the resource. After that shows device identity followed by the exact name of the description file. The sample is also presented here for a complete description.

1) MUD DHCP EXTENSION

Dynamic host configuration protocol (DHCP) is an automatic configuration protocol [107], used primarily for the automatic assignment of IP addresses to the network devices. Whenever a new device is connected to the network DHCP server assigns a unique IP address to it so that it can communicate with other devices. To make it compatible with the MUD, DHCP options have been used. Options like OPTION_MUD_URL_V4 and V6 for IPv4 and IPv6 respectively have been devised for the transportation of MUD URL.

The format of the option is shown in Figure 8. The main intent of this option is to identify the device to the network. Also, it can send certain configuration codes to the router. The length of the entire option packet does not exceed 255 octets. The first part is the code that identifies the option type either

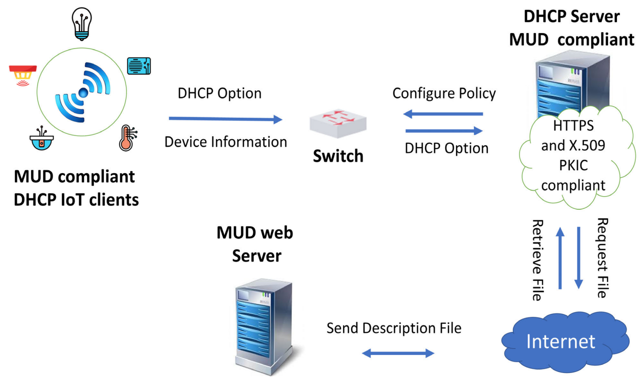

FIGURE 9. DHCP server with MUD processing capability.

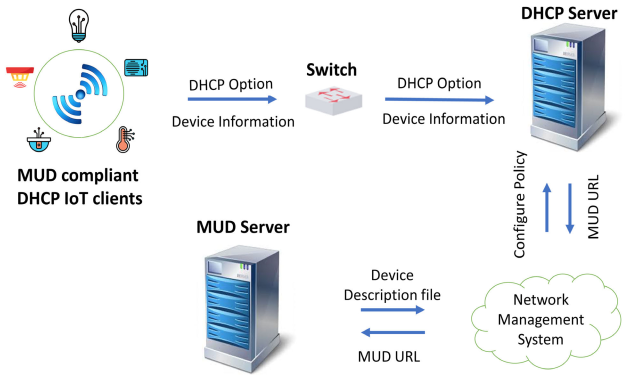

FIGURE 10. DHCP server without MUD processing capability.

IPv4 or IPv6. Then is the length that defines the total length of the following MUD string. MUD string is composed of two parts. MUD URL and some reserved part, the reserve part can be used for additional configuration or future use.

There are a couple of scenarios, one in which the DHCP server act additionally as a MUD manager as shown in Figure 9. In this case, the DHCP server first resolves the MUD URL. Then it requests the MUD web server for the device description. The MUD server should be compliant with the HTTPS and X.509 public key infrastructure certificate (PKIC) this is required to validate the Transport Layer Security (TLS) [108] certificates hosted on the webserver. The DHCP server receives the file then processes it to configure the network switch for the particular device according to the description.

In the second scenario, if the DHCP server does not act as a MUD manager as shown in Figure 10. The IoT devices embedded with the DHCP client send the MUD URL to the DHCP server. The DHCP server simply forwards the request along with the device information to the respective controller or network management system. The request contains the MUD URL along with the device information (optional) send either to the network management system or to the MUD manager. The management system resolves the URL and retrieves the MUD file from the MUD web server located on the Internet. Also, the network management system is responsible to process the MUD file and configure the network switches according to the policy mentioned in the file. Further, in either scenario, the DHCP server is responsible to update the MUD manager regarding the change in the state of the devices. This will help the network management system to update the MUD file for the device and configure the network accordingly.

2) MUD LLDP EXTENSION

Link layer discovery protocol is a layer two protocol [109]. It can only communicate to one hop, which means only directly connected devices can communicate over this interface. The link discovery service within controllers takes advantage of the data link layer LLDP to detect links between

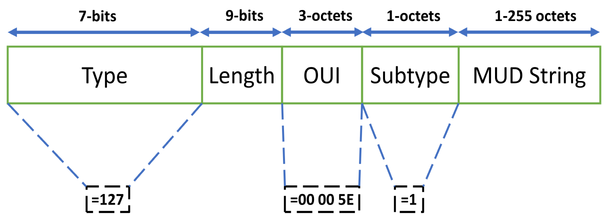

FIGURE 11. LLDP MUD extension TVL.

switches to discover the topology in the network dynamically. The LLDP information, in the form of an Ethernet frame, is sent by controllers from each of their interfaces at a fixed interval. The multicast destination MAC address and source MAC address respectively reflect destination address (DA) and source address (SA). The field with the ether type is set to 0x88CC frame contains one data unit from the LLDP (LLDPDU). A sequence of type-length-value (TLV) structures[10] is each LLDPDU that carries the payload of an LLDP frame. The following integral mandatory TLVs are always included with each LLDP frame: chassis ID, port ID, and time-to-live. Any number of optional TLVs accompany the compulsory TLVs. The frame ends with a special TLV, called the LLDPDU end, in which both the fields of form and length are zero [110].

This extension aims to provide both the network with a new Thing classifier and some suggested configurations for the routers enforcing the policy. It is, however, solely the purview of the network system to decide what to do with this information as handled by the network administrator. The main purpose of this extension is simply to structurally define the type of Thing to the network in such a way that existing toolsets will easily locate the policy.

The frame of LLDP uses specific TLVs for MUD extension. This TLV consists of a type, length, and device information, like MUD URL as shown in Figure 11. In type it uses code 127 to identify as a vendor-specific TLV, further length shows the TLV string length. The organizational unique identifier (OUI) for very vendor [111]. The rest of the frame contains the MUD string containing the MUD URL and device-specific information.

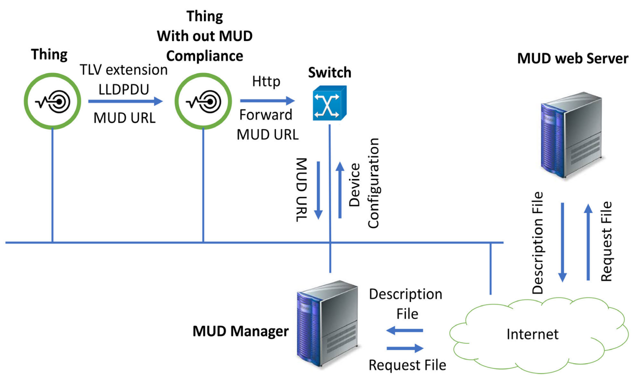

FIGURE 12. LLDP extension without MUD processing capability.

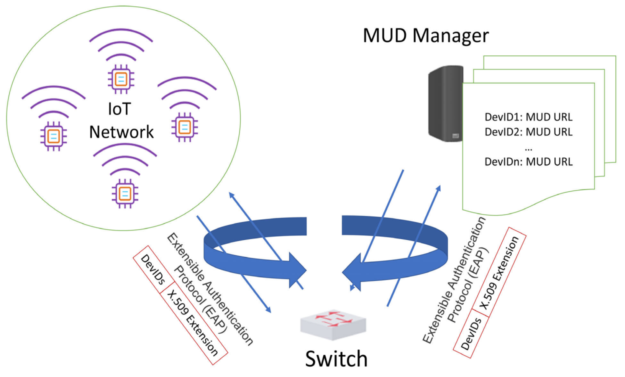

FIGURE 14. MUD extension based on X.509 Extension.

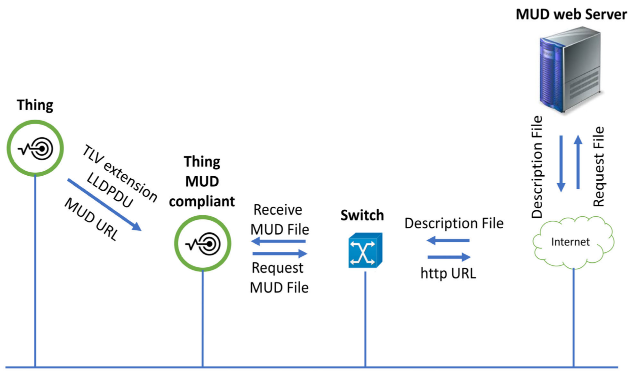

FIGURE 13. LLDP extension with MUD processing capability.

LLDP based MUD implementation has two scenarios. First is shown in Figure 12, this case discusses the scenario in which the receiver does not possess the capability to process the MUD URL on its own. Things emit the LLDP data unit (LLDPDU) containing the MUD URL to the receiver entity. When the receiver entity receives the request, since the entity does not have the MUD processing capability, so it simply forward the request to the MUD manager or network management system using the HTTP scheme. The network management system is then responsible to process the request and retrieve the MUF profile from the webserver. Also, the network management system is responsible to process the file and transform the device profile to network policies to configure the network accordingly.

In the second case, if the receiving device is MUD compliant and can process the MUD URL as shown in Figure 13, then the device after receiving the MUD URL process it without forwarding it to the MUD manager or network management system. Then the device sends the request for the device description or MUD file to the MUD web server. The server sends the device description using the Hypertext Transfer Protocol (HTTP) [112] scheme, the host then translates the file and executes the configuration task.

3) MUD X.509 EXTENSION

The MUD uses the X.509 non-critical certificate extension. It contains the MUD URL pointing to the MUD description located online. The private extension certificates are normally of two types: authority information access and subject information access [113]. MUD uses the second type. This extension contains the access information and services regarding the subject in the certificate. In this extension, MUD uses a couple of extensions: one for the MUD URL and the other for the MUD file a signing certificate. The format of the certificate is defined in [114], [115].

Normally things send the secure device IDs known as an initial device identifier (IDevID) [116] along with the X.509 certificate to the MUD manager as shown in Figure14. But some devices can use the local device identifier (LDevID) as given by the administrator of the network. This may lead to complexity at the server end. To handle the issue the standard [117] suggests maintaining a mapping scheme within the MUD manager for the device IDs and the MUD URL. This will identify the exact resource for each device in the network. The communication between the MUD manager and the devices uses the extended authentication protocol (EAP) [118] for end-to-end secure communication. EAP (Extensible Authentication Protocol) is used to transfer authentication information between the device (Wi-Fi workstation) and the authentication server (Microsoft IAS or other). Currently, the EAP form manages and describes authentication. The access point that serves as an authenticator is just a proxy that enables the requestor to connect with the authentication server. EAP-MD-5, EAP-TLS, EAP-PEAP, EAP-TTLS, EAP-Fast, and Cisco LEAP are some of the most commonly deployed EAP authentication forms [119]. The main goal now was to implement the standard to analyze the efficacy of MUD against cyber attacks. There are some implementation efforts discussed below:

VI. MUD IMPLEMENTATION

The MUD specification has provided ample flexibility and scope to choose various implementations to integrate different deployment scenario requirements. This survey will discuss various implementations of MUD deployments in this section: implementation of Cisco proof-the-concept (PoC), National Institute of Standards and Technology (NIST), CableLabs, MasterPeace Solutions [120] and open source

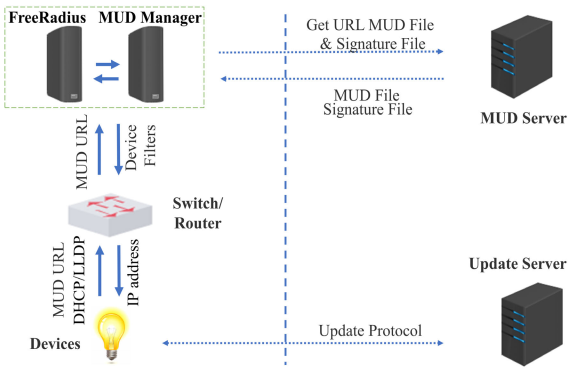

FIGURE 15. Cisco implementation of MUD [120].

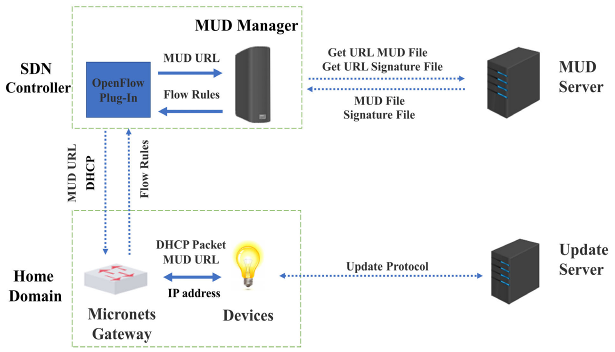

FIGURE 16. NIST implementation of MUD [120].

[121] implementation based on Software Defined Network and OpenSource MUD [122] built by a consortium of system manufacturing and network security companies (Cable Labs, Cisco, CTIA, Digicert, ForeScout, Global Cyber Alliance, Patton, and Symantec).

A. CISCO MUD

Cisco proposed a proof of concept (POC) of MUD implementation. This POC will help the engineers and the industrialist to understand the standard and its implementation requirements. As shown in Figure15. The main devices here are the MUD manager, the FreeRADIUS server (act as AAA server), and the catalyst switch 3850-S. The catalyst 3850-S switch hosted the DHCP server. Also, the switch is configured to enable the MUD on it. It enables the switch to extracts the MUD URL from the DHCP packets. The Cisco implementation shows deployment for DHCP and LLDP for the URL emission approaches. The manager communicates with the FreeRADIUS [123] server for authenticating the MUD URLs received from the catalyst switch. MongoDB [124] has been used by the manager to store the policy information of the device that is received from the MUD server [120]. The MUD-capable IoT device will be able to communicate with authorized local hosts and internet hosts as specified in the MUD file after the device's traffic filters are added to the router or switch, and any unapproved communication attempts will be blocked [125].

Open Issues: In this PoC implementation, the main limitation is the configuration of the MUD manager. As the boot process of the MUD manager requires DNS service to be configured manually. This involves human intervention in the system, causing restart of the MUD manager each time some update is required. Further, the static configurations possess serious scalability issues in the context of IoT environments. Also, the real-time management of such systems is impractical for critical applications and processes. Also, the static rules for the ACLs are a limitation in the current implementation. Though the ingress traffic is dictated according to the given policies the system allows the dynamic rules for the egress traffic causing the attackers to invade the system [120].

B. NIST MUD

The national institute of science and technology (NIST) PoC [120]implementation uses the Software-Defined Networking approach. The model took the benefit of isolation between the control and data plane using the OpenFlow switches. The deployment organizes the traffic flows in more than one flow table. This makes it possible for the SDN controller to, dynamically: after the packet arrives at the controller and proactively: before the switch connects to the controller, update the rules. This study in [122] shows one of the manifestations of this paradigm.

The logical architecture of NIST implementation as shown in Figure 16, includes a single device used both for SDN controller or MUD manager. This device is responsible to manage and control the OpenFlow-enabled SDN switch in the network. The SDN switch is used to enforce the MUD policy for the MUD-capable IoT devices. However, this implementation only supports DHCP for the MUD URL emission capability not the LLDP or certificate-based URL discovery. This deployment can manually associate a MUD file to the devices that are unable to emit MUD URL, using the device MAC address along with the file.

Open Issues: This implementation of the MUD is based on SDN but has some problems regarding packet processing as elaborated in the SDN-based PoC given by [122]. The first rule in the MAC address (first tables) classification process, which is installed when the switch connects, could allow the packet to be sent to the controller but not to the next table by following the table's pipeline. Therefore, until it can be classified, a packet may not proceed in the table's pipeline, which means performance implications, i.e. switch failure as no packets from a newly connected system can go through the first table before the rule is installed. When strict ACLs are required, the behavior described may be necessary. "However, by loosening the ACE concept, the study suggested [122] a "relaxed" style to handle these circumstances. Packets will continue in this mode in the pipeline during the implementation of classification flow rules, which can result in a breach of the MUD ACEs, provided that the system is ultimately consistent with the MUD ACEs.

TABLE 5. MUD proof of concept implementations.

Implementation	Technology	URL emilting technique			MUD Manager Implementation			Signal threatining support	Software update support
		DHCP	LLDP	MAC/ Others	Router	Device	ISP/ Other		
CISCO	• Catalyst 3850-S • FreeRADIUS server	✓	✓	✗	✗	✓	✗	✗	✓
NIST	• SDN • OpenFlow	✓	✗	✗	✗	✓	✗	✗	✓
MasterPeace	• Yikes Router • Yikes cloud • Yikes mobile App	✓	✗	✗	✓	✗	✗	✓	✓
CableLabs	• Micronet Manager • Micronet Gateways • Micronet Mobile App	✓	✗	✗	✗	✗	✓	✗	✓
Open Source	• OpenSource • OpenWRT • dnsmasq	✓	✗	✗	✓	✗	✗	✗	✗

C. MasterPeace MUD

This MUD implementation as shown in Figure 17 also supports the threat signaling and the update server. The system implements the MUD manager inside the router along with the DNS and threat signaling service. In addition to this, the deployment uses the cloud and mobile applications for the configuration of simple and MUD-enabled devices. The cloud provides the traffic rules to the router for the devices attached to it, based on the type of device. The threat-signaling capabilities in the router enable it to refrain router from connecting to the compromised domains that the threat intelligence services have marked and flagged as potentially dangerous. One important aspect is that it ensures that the devices remain updated using security patches provided by the update server [120].

Open Issues: The MUD manager in this implementation does not maintain MUD file caching hence request a new copy of the MUD file at every MUD request. Another limitation is the URL emission method, which is DHCP only, and support for LLDP and X.509 is not available. Further, The PoC uses a threat signaling service, this service works when the device tries to access a certain domain using the domain name that is further resolved to IP address by the DNS server. If the domain is on the threatened list, the access is denied. But the issue is that if the device uses the IP address of the domain rather than the name the process will not block any request to such domain, this opens up a loophole in the network for the attackers [120].

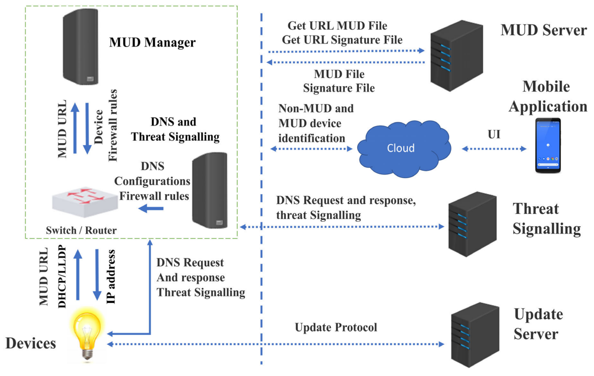

FIGURE 17. MasterPeace implementation of MUD [120].

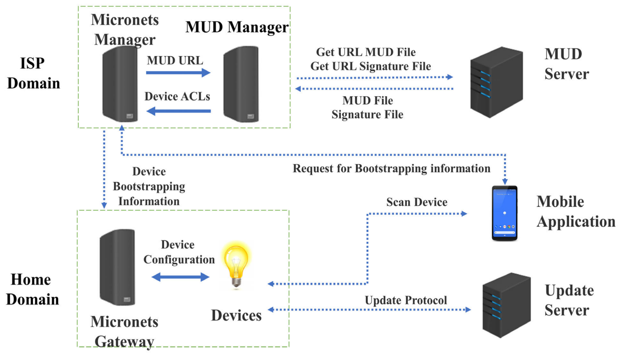

FIGURE 18. CableLabs implementation of MUD [120].

D. CableLabs MUD

This system as shown in Figure 18 works by providing MUD services by the service provider, known as micronets services. The device when first attaches to the network scan for the bootstrapping information. This bootstrap information is then sent to the service provider using the micronets gateway. Base on this information a MUD URL is generated for the MUD manager. The MUD manager gets the MUD file and the signature files from the MUD server. The MUD file is then translated to the ACL policies by the MUD manager. These ACL policies along with Micronet services are sent to the gateway for setting up the permission to the device accordingly. The devices are also attached to the update servers to keep them update against the security vulnerabilities [120].

Open Issues: One of the issues in this PoC of MUD implementation is in the context of the MUD manager security. MUD manager does not support network traffic filtering based on ports or protocols. This has an implication of the MUD process and MUD policies implementation, if the

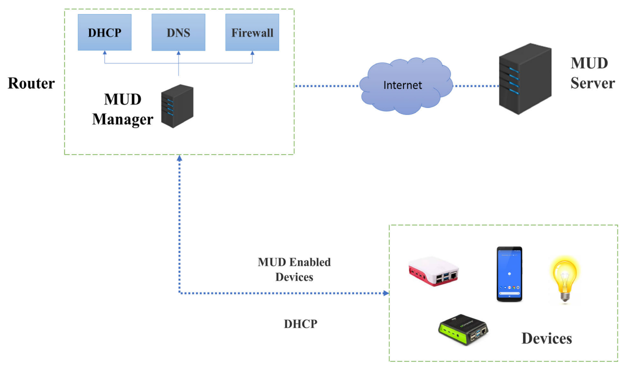

FIGURE 19. Open Source implementation of MUD [121].

MUD file allows some port and certain protocol for communication among devices, the MUD manager will erroneously allow the device to communicate using all the rest of the ports and all the available protocols. Configuring devices to the Micronet services is the responsibility of the user and this is not an ideal solution as the users don't have the technical information or know-how about the Micronet systems. A more effective solution could be to automate the device configuration system for the Micronet services [120].

E. OPEN SOURCE MUD

An open-source implementation called Open Source MUD (osMUD) [121] is developed by a consortium of companies involved in device manufacturing and network security companies. The implementation as shown in Figure 19 proposes that the MUD manager runs inside the router. Also, the system is designed such it runs on the open wireless router (OpenWRT). This also keeps the router choices limited to the routers supporting this platform only. The architecture uses *dnsmasq* service to extract DHCP packets and the MUD URL. The complete architecture is suitable for the resource-constrained routers and other devices like a firewall.

Open Issues: The limits resulting from this implementation are linked to the specifications for deployment. Only OpenWRT compliant routers that use a specific version of dnsmasq can be used in this deployment to host the MUD manager. To run the MUD Manager outside of OpenWRT, a helpful solution offered by osMUD developers is to compile it in C environments. This, however, introduces new requirements: it needs the use of a compatible firewall and a DHCP server to retrieve the MUD-URL from the MUD Enabled Devices DHCP header packet. As a result of usability for the majority of general-purpose routers [126] and making the deployment of the MUD manager simpler, OpenWRT remains the most user-friendly option.

Finally, there are no MUD file rules for lateral movement in the current implementation; therefore, attackers may gradually travel across a network, looking for targeted key data and properties.

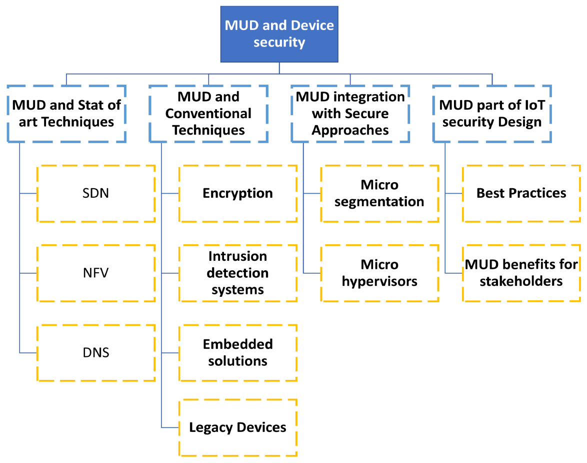
MUD and Device security
MUD and Stat of art Techniques
MUD and Conventional Techniques
MUD integration with Secure Approaches
MUD part of IoT security Design
SDN
NFV
DNS
Encryption
Intrusion detection systems
Embedded solutions
Legacy Devices
Micro segmentation
Micro hypervisors
Best Practices
MUD benefits for stakeholders

FIGURE 20. Taxonomy of MUD in device security.

The summary of the MUD implementations is given in Table 5, which shows the technologies used for the MUD URL discovery along with the infrastructure used for the communication of the MUD file. The following section focuses on the MUD application in the security of IoT networks.

VII. ROLE OF MUD IN DEVICE SECURITY

MUD has been used with different technologies like SDN, NFV, IDS and other embedded solutions for the security of the IoT devices as shown in Figure 20. The combined solutions are claimed to be more effective, yet need to be mature over time as discussed in the following subsections.

A. MUD AND STATE OF ART TECHNOLOGIES

MUD, when combining with Software defined networking along with the Virtual Network Functions, becomes a comprehensive security solution for the IoT devices [127]. The following subsections discuss MUD applications in SDN, NFV, and domain name systems.

1) MUD AND SDN

Software-Defined Network (SDN) is recently a developing technology with different management and design approaches for networking. The design paradigm of this technology decouples the data and control planes. This gave the centralized and global view of the network [128]. The controller is the decision-making authority while the switches and routers are the forwarding devices that handle data forwarding only. The forwarding devices can be programmed using one of the well-known interfaces using OpenFlow [129]. As shown in Figure 21, SDN helps MUD to monitor the device profiles using flow rules against the network traffic. Another application of SDN is the ability of the technology to enforce the MUD policies in the network for each device in online mode. The researcher starts focusing on this technique as MUD has been implemented in an SDN network [15] to

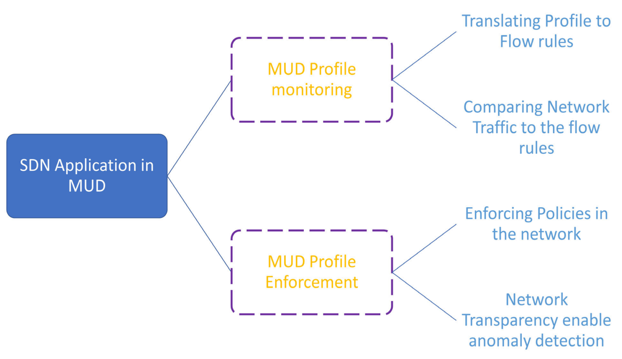
SDN Application in MUD
MUD Profile monitoring
MUD Profile Enforcement
Translating Profile to Flow rules
Comparing Network Traffic to the flow rules
Enforcing Policies in the network
Network Transparency enable anomaly detection

FIGURE 21. SDN application in MUD compliant networks.

detect volumetric attacks. The telemetry of SDN devices is based on the device level and flow level to give a transparent view of the device activity and flows. This helps to determine the malicious activity and accomplice devices.

2) MUD AND NFV

Virtual network function (VNF) is the software manifestation of network functions (like DHCP, firewall, etc) deployed on the network function virtualization (NFV) infrastructure [130]. The device security can be ensured using VNF and MUD for the home user [131]. The system is developed using the VNF and MUD for the IoT devices at the customer premises of ISP level. It modifies the MUD to be used outside the customer premises. This embeds the required intelligence to distinguish between IoT devices. The logic is based on analyzing the flows among different devices, this is achieved by using the traffic marking rules in the packet header. Normally this monitoring is executed when the traffic leaves the NAT from the IoT gateway [127].

3) MUD AND DNS

The Domain Name System (DNS) is a globally multi-operator, hierarchical, distributed infrastructure. DNS is used by IoT devices to look up the IP addresses of the remote services and hosts they require to accomplish their tasks, for instance, data analysis, security services, or other cloud services. This DNS system has some applications for IoT devices, such as checking the authenticity of devices, malicious activity. Further by adding some functions like response verification and encryption to the DNS can help to mitigate the risk of the DDoS attack on the IoT devices. The inclusion of MUD with DNS help to whitelist the devices and reduce the opportunity for the malicious devices to participate in DDoS kind of attacks [132].

B. MUD AND CONVENTIONAL TECHNOLOGIES

One of the challenges for MUD is to integrate with the conventional technology being used for the security frameworks like intrusion detection systems, legacy devices, and other embedded solutions. The flexibility of the MUD modal allows many existing technologies to merge this standard for an end to end security systems.

1) MUD AND IDS

Conventional practices for securing the traditional networks mainly rely on Intrusion Detection Systems (IDS). These systems inspect network traffic to detect malicious activity. However, most of the solutions are hardware-based that are very expensive and if the solution is software-based then the scalability is a challenge. Further, the resource heterogeneous constraint devices like IoT devices, make the task more daunting. Using MUD along with the power of SDN provides an efficient and secure intrusion detection system for IoT devices [14] by translating the MUD profile into the OpenFlow rules. The switches are proactively pre-configured with the policies and also the rules can be inserted reactively into the switches using DNS runtime bindings.

2) MUD AND LEGACY DEVICES

Managing MUD over the personal area networks as compared to the enterprise networks is a challenge. Similarly, the security of the legacy devices, since these are incompatible with the MUD added to this challenge. There are some techniques to identify and secure [133] legacy devices, like device profiling and fingerprinting. The use of SDN to determine device behavior and manage IP traffic. Using the intrusion detection system along with MUD can give real and accurate network device behavior by comparing the device's expected behavior to the current device behavior. So MUD along with detection systems provides a very efficient solution for IoT security.

3) MUD AND ENCRYPTED TRAFFIC

In the case of encrypted traffic when device identification is hard using conventional packet analysis, MUD has been used as a last line of defense for the IoT security [134]. A privacy threat mitigation technique is used using MUD for encrypted traffic. The user data and activities can be decoded using machine learning and statistical tools even over encrypted traffic. There are three ways for threat mitigation: a proactive tool based on machine learning to detect privacy threats, a Policy-based component to enforce the policies in the network to defend the attacks in the network; and if both of the above-mentioned modules fail to stop the attack then the system uses MUD based node profiles.

4) MUD AND EMBEDDED SOLUTIONS

Embedded solutions are quite common in IoT security, as localized security is better in the IoT environment than relying on remote security services. One of the well-known attacks Man in The Middle (MiTM) is an attack common to the IoT devices as they are connected with the remote services on the cloud and can be compromised. To defend against this kind of attack the research proposes a Black Pi [135] a local agent that resides on the IoT device, that detects the malicious and mutated connections from the network using the history and the list from the blacklisted databases. However, it shows that using the MUD along with the BlackPi will be a better solution to secure the IoT devices from external attacks.

C. MUD INTEGRATION IN SECURITY FRAMEWORKS

The new concept of secure by design is getting focus nowadays, many secure frameworks are designed in a way to make them secure at the very stage they were designed. In the next, subsections the study will discuss some of such efforts and the involvement of MUD in them.

1) MICROSEGMENTATION

Checking the device vulnerabilities now becoming a very important area of study. Most of the security architecture is building around this concept [136]. Security architecture for IoT devices using micro-segmentation is becoming common. Micro-segmentation is the isolation of IoT devices into segments based on the network vulnerability information of the devices. For this, the system uses two network functions, one for building the complete network device inventory and the other to check the device vulnerabilities. The model is also equally capable for the devices having MUD profiles and those who don't have MUD compatibility.

2) MICROHYPERVISORS

Aside from the security of a device, the security of gateways is much more important as they are the lifeline of a network. One of the ways to secure gateways is by using the micro hypervisors approach [137]. This helps to answer the research challenge to ensure the security of the IoT gateway itself. As the gateway that is widely used for the IoT network security, if compromised, will cause the whole network to be compromised. In this context, an idea of micro-hypervisor-based IoT gateways comes to the rescue. Micro-hypervisor is efficient as it requires fewer resources that can be used at the edge of IoT devices to implement the complete system. The use of MUD as part of the envisioned trusted security gateway architecture for IoT devices is the future security framework's goal.

3) BEST PRACTICES

One of the significant ways to provide IoT security is by using the best practices [138]. Device security can be divided into two parts: one part is to identify the device vulnerabilities, second is to determine the best practices to mitigate the potential vulnerabilities, like privacy, authentication, system operation, device policies, vulnerability mitigation, and device operation. MUD has been considered one of the best practices but still requires additional security measures to provide complete IoT security.

D. MUD BENEFITS FOR THE STAKEHOLDERS

From the point of view of stakeholders, such as service providers, manufacturers, consumers, and authorities, the need for IoT security and the required components for the IoT security solutions is inevitable. Due to the weakness of the currently available solutions, new IoT secure frameworks are the demand for this new era. The inclusion of MUD in the design of IoT framework and systems reduces the surface attack vectors and plays an important part in the manufacturer domain as it is also an essential network security solution [139].

1) BENEFIT TO THE MANUFACTURER

The majority of solutions for DDoS mitigation are focused on the filtering of problematic traffic that, if it is a network-oriented attack, is not valid for normal services. By flooding the service with distributed botnets that can make legitimate requests, the attackers can prevent such mitigation. The MUD-Manager knows the appropriate range of requests an IoT system can send, using the recently implemented Peak request rate, which is a metric suggested by the manufacturer. Based on the type of device that is inferred from the request for the MUD-File, device address, and peak request rate. The firewall could be correctly configured by both the MUD-Manager in the user network as well as the Fog MUD-Manager, enabling unnecessary requests to be dropped. Naturally, the impact of blocking flooding requests on the user side is more successful as the traffic will not even enter the external network, it will discharge the traffic on the connection between the user network and the producer network [140].

2) BENEFIT TO THE END-USER

With the dual configuration of the MUD-Manager, both the client and the manufacturer are shielded from a variety of attacks that require adversary contact. Another potentially malicious danger that can be mitigated includes backdoor, keylogger, spyware, and reconnaissance malware variants to name instances where the attacker needs a command and control infrastructure. Outgoing traffic is only permitted in the case of a smart security camera as it departs for other IoT end devices separated by MUD. It would not be possible to communicate with other servers even though the attacker took possession of the other end-devices, since they are only permitted to send traffic to the manufacturer's Fog network when both sides of the connection are separated [140].

VIII. MUD INDUSTRIAL APPLICATIONS

MUD strength has been its flexibility and adaptiveness according to the security domain. As shown in Figure 22 MUD finds its application in several industrial application like smart home, telecommunications, 5G, Fog and edge computing etc. Each of them is explained in the following subsections.

A. INDUSTRIAL IoT

Enabling industry with the power of the Internet of things is termed industrial IoT [141]. The security of the Industrial Internet of Things (IIoT) is more complex due to the availability, integrity, and confidentiality of industrial data. In this context, the security requirements for the IIoT for industrial standards need to be considered seriously. Two of such standards, OpenFog Consortium [142] and Industrial

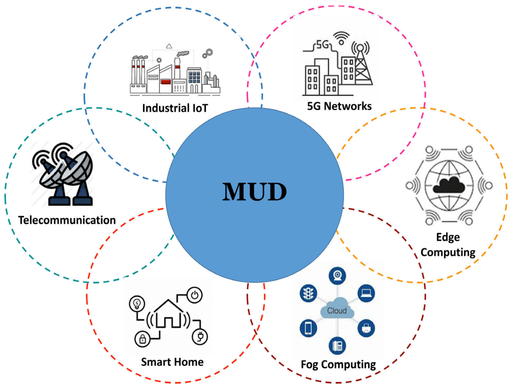

FIGURE 22. MUD applications in industry.

Internet Consortium (IIC) [143], are discussed and compared to the security protocols and platforms available for IIoT. One of the secure connectivity techniques in IIoT is MUD [144]. The MUD plays one important part for IIoT security framework solutions [144]. MUD has been one of the main standards to enforce the behavioral profiles for IoT devices to ensure security. This technique also helpful in protecting the IoT devices against the DDoS attacks [145].

B. TELECOMMUNICATION NETWORKS

As the devices are increasing day by day and the telecommunication network is overwhelmed with so many connections, there is a risk of compromised devices connected to the telecommunication service. To handle this challenge one of the ways is to detect the device vulnerability before attaching it to the telco services using machine learning-based models [146]. The Light Gradient Boosting Machine (LBMG) algorithm gave the best for the said purpose. MUD in IoT network along with some packet monitoring technique provides a more secure framework and ensure less vulnerable IoT devices get connected to other services.

C. SMART HOME

Smart home security is a serious concern, many security frameworks for the home network are presented to make them resilient to vulnerable devices. Selected IoT devices are taken and then investigated for security and privacy purpose to detect the exploitable flaws in them. This allows the system to design a security framework using the firewall, router, and other mechanisms to minimize the security flaws within the IoT devices. The addition of MUD in the security framework will help to automate the firewall policies to be configured in the newly attached network devices [147].

D. MUD IN FOG COMPUTING

Fog computing paradigm (FCP) is used for industrial IoT applications, where time-critical and deterministic networking is required. One of the models proposed a system [148] based on fog computing platform for the industrial applications using Architecture Analysis Design Language (AADL) [149], to analyze the suitability of FCP in industrial automation. AADL is used to model the software and hardware for embedding systems for real-time applications. AADL is compliant with the MUD as it has to deal with the heterogeneity challenge of IoT devices. There is a need to find the efficacy of MUD for IoT security in fog computing. In Fog computing there are many ways to secure it from DDoS attacks. However, using MUD a rate-limiting method is proposed for the IoT devices to minimize the DDoS attacks in the network even using Fog computing. MUD is capable of reducing the attack surface for IoT devices in the context of fog computing [140]. One of the antimalware systems for IoT security called antibiotic 2.0 is based on Fog computing. The system overcomes different weaknesses in its predecessor and presents a more secure solution. However, end-to-end security is still not possible, if the system is fully able to integrate with MUD the combined solution becomes much more effective and secure [150].

E. MUD IN EDGE COMPUTING

Edge computing focuses on decentralized and distributed approaches when using federated learning with MUD combine plays a vital role in providing security in the insecure IoT environment [151]. Nowadays a popular machine learning technique called Federated learning or collaborative learning is being used for model learning to train the algorithm among the distributed node. The edge nodes and the server keep the data sample and do not share it on the network, reducing the training cost in the context of network resources required [152]. One of the manifestations of combined edge computing and federated learning (FL) is to presents an IoT security solution known as colearn [151]. The co-learn system enables the FL in the MUD-compliant IoT-based edge device networks. The system use osMUD (an open-source implementation of MUD) and PySyft (implementation of FL). The system allows only the authenticated devices having the MUD profiles to participate in the machine learning and training model. This reduces the attack surface for the IoT devices. Similarly, another autonomous self-learning system called DIOT [153] also uses the federated learning technique to detect the vulnerable IoT devices in the network. DIOT develops device profiles specific to the communication with other nodes without human interaction and detects the malicious nodes. The system uses the federated learning approach to aggregate the node profile in the network.

Another decentralized approach called DARE [154] for learning the device profile has been proposed to address the issue of extracting the usage pattern of IoT devices without using machine learning technique, to make intelligent decision making, as it is more expensive to be used on the resource constraint devices. The model presents an embedded associative analysis for mining the implicit correlations among the IoT device actions. The proposed technique uses a decentralized approach and shows the ability to identify the correlations among the data that are not possible in the centralized approach. DARE when used with WoT and MUD can give a comprehensive security solution to IoT security.

F. MUD IN MOBILE & 5G APPLICATIONS

The SxC approach, originally developed for mobile applications, is now being used for IoT security. Some security systems use the security by contract [155] paradigm for IoT security. There are a lot of devices that are using MUD as a device profile for the legitimate use of the network. This system makes the SxC technique such that the legacy devices using MUD can also be used in this new security paradigm of SxC approach. The study uses the ACL, DHCP, and fingerbank API to implement this integration. The combination of MUD along with SxC contracts provides a comprehensive security solution for mobile applications.

MUD plays a vital role in the security manifest of 5G systems. The Liability-Aware Security Management (LASM) system for 5G takes into account the security by contract, manifest, remote attestation, proof of transit and reputation, and trust models. The manifest uses MUD for the control of the components. These components can be enhanced to take proper responsibilities of liability and security management systems [156].

IX. MUD AND SECURE UPDATING OF IoT DEVICES

The software/firmware update process for IoT devices The research group has gained considerable interest in recent years. Indeed, different challenges related to the constraints of IoT devices and networks must be faced by such a method. The increasing complexity of current IoT systems and implementations, on the one hand, involves dealing with the management of various software versions as well as the dependencies between device or system software components. In this context, it is necessary to ensure that upgrading the software component on a particular IoT system does not affect the normal operation or the level of protection of other components in a particular deployment. On the other hand, distributed approaches are required to minimize the effect of the updating process on resource constraints on IoT devices and networks. To do this, it is important to use lightweight and scalable methods in environments with different underlying communication technologies and protocols.

A. MUD IN CYBERSECURITY CERTIFICATION

Cybersecurity certification is getting very important in the context of information and communication technologies (ICT) security. So there is a need to distribute these artifacts within the network through a reliable and secure process. Some systems use the blockchain approach for the distribution and updating of ICT artifacts, vulnerabilities, and certificates within the network [157]. Blockchain is an open, distributed ledger that is used to record transactions among the nodes within the network efficiently, also the data is verifiable [158]. Many security solutions use Blockchain for a secure and reliable update of artifacts [159]. Also, the cybersecurity certificate should contain pertinent data for a device for a secure and automated deployment. To address this issue MUD can be integrated into the certificates to become a part of the certificate for a particular device. In this way, MUD helps to cover the gap for the security certificates in the context of secure and automated deployment of IoT devices [157].

B. MUD AND SOFTWARE UPDATES FOR INTERNET OF THINGS (SUIT)

The software or firmware update is still a challenge and many software update approaches and challenges for the IoT devices in the context of security have been discussed [160]. One of the approaches is the IEEE Software Updates for Internet of Things (SUIT) [161] working group and the blockchain. The current approaches are facing challenges for the scalability, management of versions, and efficiency. The automated installation of updates is possible and can be achieved if integrated with MUD, this restricts the device communication. This made the IEEE SUIT working group combine MUD with the SUIT approach for reliable automation in software updates [160].

X. MUD SHORTCOMINGS AND ENAHACEMENTS

MUD provides a very effective lightweight approach to provide baseline security among IoT devices. However, the standard is relatively new, the manufacturer and operators of IoT devices are still facing some limitations while deploying this standard. Some of the observed limitations in the standard are described in Table 6 below and the rest of these limitations along with the proposed solutions are explained in the following subsections. Further, the summary of proposed extensions in the MUD model has been summarised in Table 7.

A. UNABLE TO DETECT FIRMWARE VULNERABILITY

MUD is unable to identify the vulnerabilities and bugs in the device firmware before it requests the MUD profile. However, there are some advancements on the MUD [163], that only allowing the device to request a MUD profile when the vulnerability factor of the device is below a certain threshold. Also, it presents an authentication mechanism for MUD profiles for secure and reliable communication of device profiles.

B. LIMITED TO ACCESS CONTROL POLICY

The MUD standard focused on the network level access policies only with some other security attributes that are very limited in the context of the heterogeneous environment of IoT. A research [164] suggests the upgraded model. The extended MUD model uses flexible Medium-level Security Policy Language (MSPL) [165] policy language to include parameters like channel protection, resource authorization, and data privacy. The implementation is based on the SDN technique and blockchain for data safety and integrity. The system enforces the MUD at the bootstrapping of the IoT device.

TABLE 6. MUD limitations [162].

S#	Limitations
1	Automatically generate MUD profiles for the devices is not addressed in the standard.
2	MUD files generated for the compromised devices using the device's traffic trace can include malicious flows and could generate incorrect device profiles.
3	The MUD profiles are only limited to the input network traces with limited network attributes.
4	IoT devices use heterogeneous protocols like Zigbee, Z-wave, and Bluetooth, etc., thus limiting the MUD profiles to be defined for the hub devices used for internet connectivity.
5	Currently, MUD allows both "accept" and "drop" rules without specifying the priority, creating ambiguity.
6	The MUD proposal also does not support private IP addresses, instead, profiles are made readily transferrable between networks via support for high-level abstractions. For instance, to communicate with other IoT devices in the network, abstractions such as the same-manufacturer is provided.
7	Explicit IP address assignment in MUD can compromise the device and it can become a DoS target.

C. LIMITED MUD PROFILES

As the attacks are getting more complicated and hard to detect, there is a need for IoT device security testing methodology [166]. The behavior of IoT devices is critical for detecting and mitigating the security attacks, but the MUD only addresses the device network-level restrictions. There is a need to add more device attributes in the device profile, for this a model-based testing (MBT) has been introduced. The results of the automated testing methodology were added as additional attributes to augment in device MUD profile. This way study proposes an extension in the MUD model. One of the research integrates the automated IoT security testing methodology to enforce the extended MUD profiles using the Concise Binary Object Representation (CBOR) [167] based CBOR-encoded and eXtensible Access Control Markup Language (XACML) [168] technique. The resultant approach has application to the Elliptic Curve Diffie-Hellman over COSE (EDHOC) [169] protocol used for the key exchange of IoT devices.

D. STATIC ACL CONFIGURATION

The manufacturer does not know the exact parameters of the device when used in a real-time environment like IP address, host, and controller IP addresses. This lets MUD use the classes to define certain intents. Intents for a device to host communication, so that the device can communicate to certain hosts, or the device can communicate to the devices of the same manufacturer using certain ports. For this ACLs use placeholders called classes, but these classes help only for static configuration. The MUD model has been enhanced for dynamic configuration [122], shows upgrade the MUD model by first implementing MUD in the SDN network for the security of IoT network and scale up the MUD model by organizing the flow tables. This is possible by updating MUD ACL tables dynamically at runtime without requiring reconfiguration of the rules.

E. DYNAMIC POLICY UPDATE

As Amazon suffers 2.3 Tbps of DDoS attacks launched from thousands of hijacked IoT devices. One way to avert this situation is the Auto policy system [170]. The system limits the device bandwidth usage and other network resources. This policy is based on the fact that the resource constraint device requires very little bandwidth and network resources to accomplish its tasks, also the architecture uses the software-defined network. The auto policy when compared to MUD shows that the auto policy system is specifically made to address the DDoS attacks and make the device identity files more visible on the network.

One approach is dynamically updating the policies by automating the device selection based on the user-defined function. Also, the devices are selected based on the given functionality to maximize user preferences over device usage. For device selection, the research proves the genetic algorithms as the best candidate. Also, another module in the system enables systematic network policies to be generated to enforce privileges among the devices. In comparison to this, MUD is not suitable for the dynamic IoT environment. As the MUD profiles are predefined and not updated dynamically [171].

F. DEVICE IDENTIFICATION: IoT OR NON-IoT

MUD is useful for device identification but not able to identify between IoT or non-IoT devices. Since all the devices in the IoT network may not be compatible with MUD, this causes big concern to handle non-MUD compliant devices in a MUD-based environment. Some research [172] focus on the identification of IoT and non-IoT devices using a classifier-based machine learning technique. The first logistic regressional classifier uses the traffic features. While the second classifier gets the features from the DHCP packets and uses the decision tree. However to identify IoT and non-IoT devices properly both classifiers are used in the system.

XI. OTHER TECHNIQUES USING DEVICE PROFILE APPROACH

The following section discusses other proposed models like MUD. Some of them address the shortcoming of MUD. Still, MUD has the privilege to be approved as a standard.

A. DEVICE USAGE DESCRIPTION (DUD)

Most of the malware like Mirai control the device and perform some malicious behavior. This kind of behavior is not expected by the device owners. This is the reason, why the detection of malicious behavior of the device becomes critical

TABLE 7. MUD shortcomings and proposed extensions.

References	MUD Limitations	MUD Extentions	Technology
[122]	• Limited Intents • Intent for Same manufacturer device • Using same network • Using certain Port • Limited to communicate to certain IP • Specifying certain internet host	• Soft MUD model • Define intents between device and host • Define intents between device and devic	• SDN
[164]	• Network access control only • No MUD initiating process • No MUD policy enforcing	• Extended MUD model • Adding MUD management • Enforcing MUD policies process • Sharing device information among other devices	• MSPL • CoAS-EAP • Blockchain • DLT • IPFS
[166]	• Network level restriction only • No fine grain policy description method	• Extended MUD profiles • Augmented MUD profiles • Automated IoT testing Methodology	• MBT • EDHOC
[131]	• Scalability issue of MUD • Cannot distinguesh between • IoT and Non-IoT devices	• NFV based system • ISP level implementation • Distinguish IoT and non-IoT devices • MUD enforcement • White list monitoring (WLM) • White list enforcement (WLE)	• NFV • OpenFlow
[173]	• Secure communication of MUD URL issue • Secure MUD file specification issue • Update MUD file after expiration	• MUD user interface • OpenWRT router • osMUD Manager • MUD file server • User Policy Server (UPS)	• osMUD • MUDgee
[163]	• Configuration and firmware vulnerabilities • No Authorization mechanism in MUD • No secure enforcement process in MUD	• eMUD • OWASP firmware testing methodology for discovering vulnerabilities • Secure MUD file distribution using authentication process	• osMUD • Blockchain
[174]	• MUD for COTS devices • Not have Fine Grain Traffic filtering ability	• MLogger • Converts the MUD specification into the firwall polices. • User access to the COTS devices to define policies.	• OpenWRT • Zest
[145]	• Obtaining generating MUD profile process not define • Enforcing the MUD profile process not define	• MUD-enhanced bootstrapping • Smart object authentication • MUD obtaining and translation • Enforcement of the MUD file	• SDN • AAA Server • MUD file server

for IoT security. IoT-Praetor [175] can detect undesired device behavior of a device using a device usage description (DUD) model to construct the device behavior specification. Also, the system can extract the device behavior in real-time. The system uses SAMSUNG smart things, yet it used the DUD technique rather than MUD for the device profile.

B. DATA-DRIVEN MODEL

The data-driven model is about a specific system, the model finds the relationship between the system state input and output variables, the model doesn't have any knowledge about the behavior of the system. One of the works [176] proposes the SDN and machine learning-based telemetry of network flow and data-driven models to monitor the IoT devices and their communication. Monitor the traffic flows from a device over a specific period scale from minute to hour. The model inference is based on the flow attributes used to classify the IoT and non-IoT devices and the type of devices.

C. NETWORK SYSTEM MODEL

Using the Home Area Network Zero Operations (HANZO) controller [177], the network system model offers computer setup, security, and operations management. The controller manages IoT devices and creates HANZO profiles, dynamic system profiles that are identical to MUD profiles. From network traffic observation, all necessary MUD profile parameters can be derived. Contact endpoints and system recognition fingerprint data are included in the HANZO profile. Using traffic observation time windows, the endpoints are extracted. A combination of protocol heuristics and traffic classification methods is used to extract the fingerprinting of the system. The HANZO profiles are made into network-supported ACLs that restrict the endpoints of system communication. Subsequently, by explicitly mapping ACLs into flow-table rules, the solution implements ACLs using an SDN infrastructure for better programmability.

XII. OPEN ISSUES, CHALLENGES, AND FUTURE RESEARCH DIRECTIONS

In the previous sections, this research reviewed MUD applications and shortcomings as well as the proposed improvements. However, there are still many challenges for the adoption of MUD in the exploding world of heterogeneous IoT devices. In this section, the paper highlight some of the future research issues and challenges in this domain.

A. MUD URL LEARNING TECHNIQUES

There are currently three ways for the MUD URL emitting techniques, i.e DHCP, LLDP, and X.509 [8]. However, it is quite possible to devise new means for MUD URLs to be learned by the network. In the case of resource constraint devices, some devices are so limited that they have limited ability to communicate using MUD URL. However, there are few ways to discover such devices like public keys or serial numbers. Such ways can be embedded into the MUD URL or MUD files. Sometimes devices do not have the means to communicate with the Internet or the Internet does not exist, this scenario is an open challenge for the manufacturers and the implementers of the MUD process.

B. DEVICE REGISTERATION FAILURE

Some time the MUD manager is not able to retrieve the MUD file, causing the device registration failure. This scenario can happen due to many reasons. The reason could be the MUD manager is offline, network connectivity failure or internet connectivity got disconnected or probably some power failure [122]. This can become the reason for device registration failure. Such issues need to be addressed in the future implementation of MUD-based systems.

C. EXTENDED POLICY BASED MODELS

The diversity of existing and upcoming IoT devices and the involvement of new state of art technologies made network security a huge challenge. The focus of MUD on the network level access policies does not meet the future demands for the security of IoT devices. The current MUD model required some additional attributes in the TCP/IP stack layers to make it more secure and flexible. But there is a need to add more features to develop a more secure MUD model [178]. Further, the inclusion of the quality of service parameters in the model will likely enhance the security and reliability of the system.

D. FINGERPRINTING ALONG WITH MUD

Some of the management systems use methods for fingerprinting the devices in the network [75]. In this scenario, MUD-based devices can be identified from the non-MUD compliant devices. Fingerprinting when used with MUD can provide other benefits like monitoring the behavior of the device as they behave according to the organization policy. For this, the reserved fields can be used, but the complete procedure can be discussed in future versions of MUD.

E. MUD MANAGER SECURITY

There is a single point of failure issue in the current MUD standard. The MUD manager can be compromised by the attacker and all the information about the network can be leaked [8]. This scenario can be addressed by following certain SOP's when accepting the MUD URL. The MUD manager should not accept device descriptions from the same MAC address claiming to have different URL domain authority without further validation. However, MUD manager security requires more description in upcoming versions of the standard.

F. MUD PROFILE UPDATION

As the MUD profile describe the policies for the device and the file description must be updated only if the device functionality is changed or updated [162]. But MUD standard does not give any clear guidelines for the file description updates for a device. The device description should not be updated based on the change of the network. So the MUD file updating process needs to be considered in the future versions of the standard for the clear guidelines of the manufacturer and developers of MUD-based security systems.

G. MUD DEVICE DESCRIPTION AUTOMATION

The file description contains important information about the identity, functionality, and manufacturer of the device. MUD standard explains all the aspects in detail along with the format of writing such profiles. The missing part is the how-to automate the process of description file generation [153]. Several manufacturers are developing IoT devices and are increasing day by day. It's quite challenging for every manufacturer to understand and produce the same MUD description for every device as intended. There is a need to develop a standardized automated description file tool using the network traces as one such effort has been done in [179].

XIII. CONCLUSION

IoT security has increasingly become a challenge and a two-sided effort is underway, the attackers are searching for more vulnerabilities in IoT devices while security providers are trying to develop more mature and reliable technologies that become the bases of less vulnerable IoT devices. One such technique is device identification, which plays a role in reducing the attack surface in IoT devices. This paper presented an overview of the premise of this hypothesis, the survey thoroughly discussed device profiling and device identification techniques and their role in IoT device security. Further, the paper elaborated on open issues and future research gaps in such approaches. The major part of this research investigated the recent standard Manufacturer Usage Description (MUD) that helps to identify IoT devices and prescribe their intended use. The study analyzed MUD architecture and its building blocks along with its applications in the industry. MUD comes out to be very useful for basic device security, especially due to the lightweight MUD model and simplicity of implementation. Also, some of the problems and issues related to the design and implementation of the MUD model were discussed. Finally, there was a discussion of the MUD integration with state of art technologies and other security frameworks.

The overall study provides a good foundation for researchers and developers of IoT security systems to get an overview of device identification technologies and the MUD approach, without going deep down in reading a plethora of technological reviews and standard details. The research effort discusses many research gaps in this domain to give future research a proper roadmap to follow.

ACKNOWLEDGMENT

The authors of this research would like to thank the anonymous reviewers for the valuable comments and constructive suggestions, and their insights to improve the quality of the manuscript.

REFERENCES

[1] Gartner. *Gratner Newsroom*. Accessed: 2015. [Online]. Available: https://gartner.com/en/newsroom/id/283971

[2] T. Shah and S. Venkatesan, "Authentication of IoT device and IoT server using secure vaults," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 819–824.

[3] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[4] D. Yu, P. Li, Y. Chen, Y. Ma, and J. Chen, "A time-efficient multi-protocol probe scheme for fine-grain IoT device identification," *Sensors*, vol. 20, no. 7, p. 1863, Mar. 2020.

[5] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "HoMonit: Monitoring smart home apps from encrypted traffic," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1074–1088.

[6] Y. Tian, N. Zhang, Y.-H. Lin, X. Wang, B. Ur, X. Guo, and P. Tague, "Smartauth: User-centered authorization for the Internet of Things," in *Proc. 26th Secur. Symp. (USENIX Secur.)*, 2017, pp. 361–378.

[7] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging iot application frameworks," in *Proc. 25th USENIX Secur. Symp. (USENIX Secur.)*, 2016, pp. 531–548.

[8] E. Lear, R. Droms, and D. Romascanu, *Manufacturer Usage Description Specification*, document RFC 8520, Internet Engineering Task Force, 2019.

[9] T. Bray, *The Javascript Object Notation (JSON) Data Interchange Format*, document RFC 8259, Internet Engineering Task Force (IETF), 2017.

[10] M. Jethanandani, D. Blair, L. Huang, and S. Agarwal, *Yang Data Model for Network Access Control Lists*, document RFC 8519, 2019.

[11] Cisco. *Manufacturer Usage Descriptions*. Accessed: 2018. [Online]. Available: https://developer.cisco.com/site/mud/

[12] M. N.-B. A. U. Manufacturer, "Securing small-business and home Internet of Things (IoT) devices," Approach, Archit., Secur. Characteristics, NIST Special Publication, Sep. 2020, p. 15A.

[13] K. Benzekki, A. El Fergougui, and A. E. Elalaoui, "Software-defined networking (SDN): A survey," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5803–5833, Dec. 2016.

[14] A. Hamza, H. H. Gharakheili, and V. Sivaraman, "Combining MUD policies with SDN for IoT intrusion detection," in *Proc. Workshop IoT Secur. Privacy*, Aug. 2018, pp. 1–7.

[15] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on IoT devices via SDN-based monitoring of MUD activity," in *Proc. ACM Symp. SDN Res.*, Apr. 2019, pp. 36–48.

[16] OWASP. (2018). *Owasp Internet of Things (IoT) Project*. [Online]. Available: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project

[17] N. Mazhar, R. Salleh, M. Asif, and M. Zeeshan, "SDN based intrusion detection and prevention systems using manufacturer usage description: A survey," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 12, pp. 717–737, 2020, doi: 10.14569/IJACSA.2020.0111283.

[18] FORTUNE. (2019). *Internet of Things Market Size, Share & COVID-19 Impact Analysis*. [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307

[19] K. Gyarmathy. (2020). *Comprehensive Guide to IoT Statistics You Need to Know in 2020*. [Online]. Available: https://www.vxchnge.com/blog/iot-statistics

[20] A. Rock. (2020). *Report: 57% of IoT Devices Vulnerable to Severe Attack*. [Online]. Available: https://www.newsbreak.com/news/0OanKNMX/report-57-of-iot-devices-vulnerable-to-severe-attack

[21] *Exploit Database Local*. Accessed: 2020. [Online]. Available: https://www.exploit-db.com/local/

[22] *Exploit Database Remote*. Accessed: 2020. [Online]. Available: https://www.exploit-db.com/remote/

[23] S. L. Ziv Chang. *The IoT Attack Surface: Threats and Security Solutions—Security News*. Accessed: May 30, 2019. [Online]. Available: https://www.trendmicro.com/vinfo/gb/security/news/internet-of-things/the-iot-attack-surface-threats-and-security-solutions

[24] D. Geneiatakis, I. Kounelis, R. Neisse, I. Nai-Fovino, G. Steri, and G. Baldini, "Security and privacy issues for an IoT based smart home," in *Proc. 40th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2017, pp. 1292–1297.

[25] M.-K. Choi, R. J. Robles, C.-H. Hong, and T.-H. Kim, "Wireless network security: Vulnerabilities, threats and countermeasures," *Int. J. Multimedia Ubiquitous Eng.*, vol. 3, no. 3, pp. 77–86, 2008.

[26] B. Khoo, "RFID as an enabler of the Internet of Things: Issues of security and privacy," in *Proc. Int. Conf. Internet Things, 4th Int. Conf. Cyber, Phys. Social Comput.*, Oct. 2011, pp. 709–712.

[27] A. Mitrokotsa, M. R. Rieback, and A. S. Tanenbaum, "Classification of RFID attacks," *Gen*, vol. 15693, no. 14443, p. 14, 2010.

[28] L. Li, "Study on security architecture in the Internet of Things," in *Proc. Int. Conf. Meas., Inf. Control*, vol. 1, May 2012, pp. 374–377.

[29] R. Uttarkar and R. Kulkarni, "Internet of Things: Architecture and security," *Int. J. Comput. Appl.*, vol. 3, no. 4, pp. 12–19, 2014.

[30] M. Burmester and B. De Medeiros, "RFID security: Attacks, countermeasures and challenges," in *Proc. 5th RFID Acad. Convocation, RFID J. Conf.*, 2007, pp. 1–10.

[31] P. I. Radoglou Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the Internet of Things: Challenges, threats and solutions," *Internet Things*, vol. 5, pp. 41–70, Mar. 2019.

[32] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: Attack and defense strategies," *IEEE Netw.*, vol. 20, no. 3, pp. 41–47, May 2006.

[33] S. Vadlamani, B. Eksioglu, H. Medal, and A. Nandi, "Jamming attacks on wireless networks: A taxonomic survey," *Int. J. Prod. Econ.*, vol. 172, pp. 76–94, Feb. 2016.

[34] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the RPL-based Internet of Things," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 8, Aug. 2013, Art. no. 794326.

[35] L. K. Bysani and A. K. Turuk, "A survey on selective forwarding attack in wireless sensor networks," in *Proc. Int. Conf. Devices Commun. (ICDeCom)*, Feb. 2011, pp. 1–5.

[36] A. Mayzaud, R. Badonnel, and I. Chrisment, "A taxonomy of attacks in RPL-based Internet of Things," *Int. J. Netw. Secur.*, vol. 18, no. 3, pp. 459–473, 2016.

[37] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, Oct. 2002.

[38] K. Zhang, X. Liang, R. Lu, and X. Shen, "Sybil attacks and their defenses in the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 5, pp. 372–383, Oct. 2014.

[39] R. John, J. P. Cherian, and J. J. Kizhakkethottam, "A survey of techniques to prevent sybil attacks," in *Proc. Int. Conf. Soft-Comput. Netw. Secur. (ICSNS)*, Feb. 2015, pp. 1–6.

[40] T. Bhattasali, R. Chaki, and S. Sanyal, "Sleep deprivation attack detection in wireless sensor network," *Int. J. Comput. Appl.*, vol. 40, no. 15, pp. 19–25, 2012.

[41] P. Nagrath and B. Gupta, "Wormhole attacks in wireless adhoc networks and their counter measurements: A survey," in *Proc. 3rd Int. Conf. Electron. Comput. Technol.*, Apr. 2011, pp. 245–250.

[42] A. S. A. M. S. Ahmed, R. Hassan, and N. E. Othman, "IPv6 neighbor discovery protocol specifications, threats and countermeasures: A survey," *IEEE Access*, vol. 5, pp. 18187–18210, 2017.

[43] N. Ahmed, A. Sadiq, A. Farooq, and R. Akram, "Securing the neighbour discovery protocol in IPv6 state-ful address auto-configuration," in *Proc. IEEE Trustcom/BigDataSE/ICESS*, Aug. 2017, pp. 96–103.

[44] A. Hoehn and P. Zhang, "Detection of replay attacks in cyber-physical systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2016, pp. 290–295.

[45] B. Chen, D. W. C. Ho, G. Hu, and L. Yu, "Secure fusion estimation for bandwidth constrained cyber-physical systems under replay attacks," *IEEE Trans. Cybern.*, vol. 48, no. 6, pp. 1862–1876, Jun. 2018.

[46] V. Dragoi, T. Richmond, D. Bucerzan, and A. Legay, "Survey on cryptanalysis of code-based cryptography: From theoretical to physical attacks," in *Proc. 7th Int. Conf. Comput. Commun. Control (ICCCC)*, May 2018, pp. 215–223.

[47] M. T. Manavi, "Defense mechanisms against distributed denial of service attacks: A survey," *Comput. Electr. Eng.*, vol. 72, pp. 26–38, Nov. 2018.

[48] P. S. Fulare and N. Chavhan, "False data detection in wireless sensor network with secure communication," *Int. J. Smart Sensor Adhoc Network.*, vol. 1, no. 1, pp. 66–71, Jan. 2011.

[49] J.-W. H. Bullée, L. Montoya, W. Pieters, M. Junger, and P. Hartel, "On the anatomy of social engineering attacks—A literature-based dissection of successful attacks," *J. Investigative Psychol. Offender Profiling*, vol. 15, no. 1, pp. 20–45, 2018.

[50] J. Saleem and M. Hammoudeh, "Defense methods against social engineering attacks," in *Computer and Network Security Essentials*. Cham, Switzerland: Springer, 2018, pp. 603–618.

[51] S. William, *Computer Security: Principles And Practice*. London, U.K.: Pearson, 2008.

[52] R. Kissel, *Glossary of Key Information Security Terms*. Darby, PA, USA: Diane, 2011.

[53] C. Cowan, F. Wagle, C. Pu, S. Beattie, and J. Walpole, ''Buffer overflows: Attacks and defenses for the vulnerability of the decade,'' in *Proc. DARPA Inf. Survivability Conf. Expo. (DISCEX)*, vol. 2, 2000, pp. 119–129.

[54] P. Anu and S. Vimala, ''A survey on sniffing attacks on computer networks,'' in *Proc. Int. Conf. Intell. Comput. Control (IC)*, Jun. 2017, pp. 1–5.

[55] K. Nirmal, B. Janet, and R. Kumar, ''Phishing—The threat that still exists,'' in *Proc. Int. Conf. Comput. Commun. Technol. (ICCCT)*, Feb. 2015, pp. 139–143.

[56] K. Nirmal, B. Janet, and R. Kumar, ''Analyzing and eliminating phishing threats in IoT, network and other Web applications using iterative intersection,'' *Peer Peer Netw. Appl.*, vol. 14, pp. 1–13, Jun. 2020.

[57] R. K. Aggarwal, ''A survey on comparative analysis of tools for the detection of ARP poisoning,'' in *Proc. 2nd Int. Conf. Telecommun. Netw. (TEL-NET)*, Aug. 2017, pp. 1–6.

[58] J. D. Brown and T. J. Willink, ''ARP cache poisoning and routing loops in ad hoc networks,'' *Mobile Netw. Appl.*, vol. 23, no. 5, pp. 1306–1317, Oct. 2018.

[59] Q. Hu and G. P. Hancke, ''A session hijacking attack on physical layer key generation agreement,'' in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Mar. 2017, pp. 1418–1423.

[60] Z. Lu, F. Chen, G. Cheng, and S. Li, ''The best defense strategy against session hijacking using security game in SDN,'' in *Proc. IEEE 19th Int. Conf. High Perform. Comput. Commun., IEEE 15th Int. Conf. Smart City, IEEE 3rd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2017, pp. 419–426.

[61] S. Ohata, Y. Kawai, T. Matsuda, G. Hanaoka, and K. Matsuura, ''Re-encryption verifiability: How to detect malicious activities of a proxy in proxy re-encryption,'' in *Proc. Cryptographers' Track RSA Conf.* Cham, Switzerland: Springer, 2015, pp. 410–428.

[62] B. Libert and D. Vergnaud, ''Tracing malicious proxies in proxy re-encryption,'' in *Proc. Int. Conf. Pairing-Based Cryptogr.* Berlin, Germany: Springer, 2008, pp. 332–353.

[63] 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)*, Jun. 2017, pp. 2177–2184.

[64] M. N. Aman, B. Sikdar, K. C. Chua, and A. Ali, ''Low power data integrity in IoT systems,'' *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3102–3113, Aug. 2018.

[65] T. Kohno, A. Broido, and K. C. Claffy, ''Remote physical device fingerprinting,'' *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 2, pp. 93–108, Feb. 2005.

[66] D. L. Mills, ''Network time protocol (version 3) specification, implementation and analysis,' Internet request for comments,'' Netw. Working Group, Univ. Delaware, Newark, Delaware, Tech. Rep. RFC-1305, 1992.

[67] D. Mills, *Simple Network Time Protocol (SNTP) Version 4 for IPV4, IPV6 and OSI*, document RFC 2030, Oct. 1996.

[68] J. Whatley, ''W.C.Jakes: 'Microwave mobile communications,'' *J. Inf. Technol. Educ.*, vol. 1, pp. 53–63, 1974.

[69] M. N. Aman, M. H. Basheer, and B. Sikdar, ''A lightweight protocol for secure data provenance in the Internet of Things using wireless fingerprints,'' *IEEE Syst. J.*, early access, Jul. 6, 2020, doi: 10.1109/JSYST.2020.3000269.

[70] J. François, H. Abdelnur, and O. Festor, ''Automated behavioral fingerprinting,'' in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Berlin, Germany: Springer, 2009, pp. 182–201.

[71] J. Francois, H. Abdelnur, R. State, and O. Festor, ''Machine learning techniques for passive network inventory,'' *IEEE Trans. Netw. Service Manage.*, vol. 7, no. 4, pp. 244–257, Dec. 2010.

[72] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, ''GTID: A technique for physical device and device type fingerprinting,'' *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 519–532, Sep. 2015.

[73] S. Siby, R. R. Maiti, and N. Tippenhauer, ''IoTScanner: Detecting and classifying privacy threats in IoT neighborhoods,'' 2017, *arXiv:1701.05007*. [Online]. Available: http://arxiv.org/abs/1701.05007

[74] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal, ''IoT device identification via network-flow based fingerprinting and learning,'' in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 103–111.

[75] N. Ammar, L. Noirie, and S. Tixeuil, ''Autonomous identification of IoT device types based on a supervised classification,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[76] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, ''ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis,'' in *Proc. Symp. Appl. Comput.*, Apr. 2017, pp. 506–509.

[77] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici, ''Detection of unauthorized IoT devices using machine learning techniques,'' 2017, *arXiv:1709.04647*. [Online]. Available: http://arxiv.org/abs/1709.04647

[78] A. Aksoy and M. H. Gunes, ''Automated IoT device identification using network traffic,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[79] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, ''A machine learning based framework for IoT device identification and abnormal traffic detection,'' *Trans. Emerg. Telecommun. Technol.*, p. e3743, Sep. 2019.

[80] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, ''IoTSense: Behavioral fingerprinting of IoT devices,'' 2018, *arXiv:1804.03852*. [Online]. Available: http://arxiv.org/abs/1804.03852

[81] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, ''DEFT: A distributed IoT fingerprinting technique,'' *IEEE Internet Things J.*, vol. 6, no. 1, pp. 940–952, Feb. 2019.

[82] S. A. Hamad, Q. Z. Sheng, W. E. Zhang, and S. Nepal, ''Realizing an Internet of secure things: A survey on issues and enabling technologies,'' *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1372–1391, 2nd Quart., 2020.

[83] G. Baldini, R. Giuliani, G. Steri, I. Sanchez, and C. Gentile, ''The application of the symbolic aggregate approximation algorithm (SAX) to radio frequency fingerprinting of IoT devices,'' in *Proc. IEEE Symp. Commun. Veh. Technol. (SCVT)*, Nov. 2017, pp. 1–6.

[84] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, ''AuDI: Toward autonomous IoT device-type identification using periodic communication,'' *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1402–1412, Jun. 2019.

[85] F. Hosseinpour, P. V. Amoli, J. Plosila, T. Hämäläinen, and H. Tenhunen, ''An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach,'' *Int. J. Digit. Content Technol. Appl.*, vol. 10, pp. 1–14, Dec. 2016.

[86] S. Rathore and J. H. Park, ''Semi-supervised learning based distributed attack detection framework for IoT,'' *Appl. Soft Comput.*, vol. 72, pp. 79–89, Nov. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494618303508

[87] S. Zeng, X. Tong, N. Sang, and R. Huang, ''A study on semi-supervised FCM algorithm,'' *Knowl. Inf. Syst.*, vol. 35, no. 3, pp. 585–612, Jun. 2013.

[88] J. Bao, B. Hamdaoui, and W.-K. Wong, ''IoT device type identification using hybrid deep learning approach for increased IoT security,'' in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2020, pp. 565–570.

[89] L. Fan, S. Zhang, Y. Wu, Z. Wang, C. Duan, J. Li, and J. Yang, ''An IoT device identification method based on semi-supervised learning,'' in *Proc. 16th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2020, pp. 1–7.

[90] T. T. Lu, ''Fundamental limitations of semi-supervised learning,'' M.S. thesis, Dept. Comput. Sci., Univ. Waterloo, Waterloo, ON, Canada, 2009.

[91] A. Sadarangani and A. Jivani, ''A survey of semi-supervised learning,'' *Int. J. Eng. Sci. Res. Technol.*, vol. 5, no. 10, pp. 138–143, Oct. 2016.

[92] S. Aneja, N. Aneja, and M. S. Islam, ''IoT device fingerprint using deep learning,'' in *Proc. IEEE Int. Conf. Internet Things Intell. Syst. (IOTAIS)*, Nov. 2018, pp. 174–179.

[93] J. Ortiz, C. Crawford, and F. Le, ''DeviceMien: Network device behavior modeling for identifying unknown IoT devices,'' in *Proc. Int. Conf. Internet Things Design Implement.*, Apr. 2019, pp. 106–117.

[94] J. Kotak and Y. Elovici, ''IoT device identification using deep learning,'' 2020, *arXiv:2002.11686*. [Online]. Available: http://arxiv.org/abs/2002.11686

[95] P. Angelov and A. Sperduti, ''Challenges in deep learning,'' in *Proc. ESANN*, 2016, pp. 1–8.

[96] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, ''Deep learning for IoT big data and streaming analytics: A survey,'' *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[97] S. McCloskey, ''Probabilistic reasoning and Bayesian networks,'' in *Proc. Neural Netw. Mach. Learn. (ICSG)*, 2000.

[98] A. G. Sreedevi and T. R. Rao, "Reinforcement learning algorithm for 5G indoor device-to-device communications," *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 9, p. e3670, Sep. 2019.
[99] A. Joy. *Pros and Cons of Reinforcement Learning*. Accessed: 2020. [Online]. Available: https://www.pythonistaplanet.com/pros-and-cons-of-reinforcement-learning/
[100] J. Koo and Y.-G. Kim, "Interoperability of device identification in heterogeneous IoT platforms," in *Proc. 13th Int. Comput. Eng. Conf. (ICENCO)*, Dec. 2017, pp. 26–29.
[101] J. Koo, S.-R. Oh, and Y.-G. Kim, "Device identification interoperability in heterogeneous IoT platforms," *Sensors*, vol. 19, no. 6, p. 1433, Mar. 2019.
[102] K. Singla and J. Bose, "IoT2Vec: Identification of similar IoT devices via activity footprints," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2018, pp. 198–203.
[103] H. Zhou, N. Cam-Winget, J. Salowey, and S. Hanna, *Tunnel Extensible Authentication Protocol (TEAP) Version 1*, document RFC 7170, May 2014.
[104] S. Alexander and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, document RFC 2132, 1997, pp. 1–32.
[105] P. Congdon, "Link layer discovery protocol and MIB," *V1. 0 May*, vol. 20, no. 2002, pp. 1–20, 2002.
[106] E. Rescorla, *HTTP Over TLS*, document RFC 2818, Internet Engineering Task Force, 2000.
[107] R. Droms, "Dynamic host configuration protocol," Netw. Working Group, Bucknell Univ., Lewisburg, PA, USA, Tech. Rep. RFC-2131, 1997.
[108] T. Dierks and E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2*, document RFC 5246, The Internet Engineering Task Force (IETF), 2008.
[109] *Link Layer Discovery Protocol*, IEEE Standard 802.1 ab, Mar. 2016.
[110] Y. Li, Z.-P. Cai, and H. Xu, "LLMP: Exploiting LLDP for latency measurement in software-defined data center networks," *J. Comput. Sci. Technol.*, vol. 33, no. 2, pp. 277–285, Mar. 2018.
[111] D. Eastlake, III, and J. Abley, *IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters*, document RFC 7042, IETF, Geneva, Switzerland, 2013.
[112] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC2616: Hypertext transfer protocol–HTTP/1.1," *Status, Standards Track*, vol. 1, no. 11, pp. 1829–1841, Jun. 1999.
[113] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, *Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, document RFC 5280, IETF, May 2008.
[114] J. S. P. Hoffman, *New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)*, document RFC 5912, IETF, Jun. 2010.
[115] S. T. J. Schaad, *Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)*, document RFC 6268, IETF, Jul. 2011.
[116] M. Seaman, *802.1AR: Secure Device Identity*, IEEE Standard 802.1AR-2018, 2018.
[117] N. Elkins and R. Hamilton, *IPV6 Performance and Diagnostic Metrics (PDM) Destination Option*, document RFC 8250, Sep. 2017. [Online]. Available: https://www.rfc-editor
[118] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, *Extensible Authentication Protocol (EAP)*, document RFC 3748, NetworkWorking Group, 2004.
[119] Intel. *802.1X Overview and EAP Types*. Accessed: 2020. [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000006999/network-and-io/wireless.html
[120] D. Dodson, D. Montgomery, W. Polk, M. Ranganathan, M. Souppaya, S. Johnson, A. Kadam, C. Pratt, D. Thakore, and M. Walker, "Securing small business and home Internet of Things (IoT) devices: Mitigating network-based attacks using manufacturer usage description (MUD)," National Institute of Standards and Technology, Gaithersburg, MD, USA, Tech. Rep. SP 1800-15, 2020.
[121] (2018). *Open Source MUD Manager*. [Online]. Available: https://osmud.org
[122] M. Ranganathan, "Soft MUD: Implementing manufacturer usage descriptions on OpenFlow SDN switches," in *Proc. Int. Conf. Netw. (ICN)*, 2019, pp. 1–6.
[123] A. DeKok. (2008). *Freeradius*. [Online]. Available: Http://Freeradius.org
[124] I. MongoDB. (2016). *Mongodb*. [Online]. Available: https://www.mongodb.com/.Citedon
[125] A. Mortensen and R. T. Moskowitz, *Ddos Open Threat Signaling (DOTS) Requirements*, document RFC 8612, May 2019.
[126] OpenWRT. *Openwrt Wireless Freedom*. Accessed: 2020. [Online]. Available: https://openwrt.org/toh/start
[127] Y. Afek, A. Bremler-Barr, D. Hay, R. Goldschmidt, L. Shafir, G. Avraham, and A. Shalev, "NFV-based IoT security for home networks using MUD," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2020, pp. 1–9.
[128] M. McBride, M. Cohn, S. Deshpande, M. Kaushik, M. Mathews, and S. Nathan, "SDN security considerations in the data center," *Open Netw. Found.-Onf Solution Brief*, pp. 15–16, Oct. 2013.
[129] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[130] Techplayon. *Virtual Network Function (VNF) Definition, Architecture and Design*. Accessed: 2019. [Online]. Available: http://www.techplayon.com/virtual-network-function-vnf-definition-architecture-and-design/
[131] Y. Afek, A. Bremler-Barr, D. Hay, L. Shafir, and I. Zhaika, "Demo: NFV-based IoT security at the ISP level," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp. (NOMS)*, Apr. 2020, pp. 1–2.
[132] C. Hesselman, M. Kaeo, L. Chapin, K. Claffy, M. Seiden, D. McPherson, D. Piscitello, A. McConachie, T. April, J. Latour, and R. Rasmussen, "The DNS in IoT: Opportunities, risks, and challenges," *IEEE Internet Comput.*, vol. 24, no. 4, pp. 23–32, Jul. 2020.
[133] K. Henderson and K. Kornegay, "Improving architectures for automating network security using specification-based protocols," in *Proc. 7th Symp. Hot Topics Sci. Secur.*, Sep. 2020, pp. 1–2.
[134] G. Baldini, J. L. Hernandez-Ramos, S. Nowak, R. Neisse, and M. Nowak, "Mitigation of privacy threats due to encrypted traffic analysis through a policy-based framework and MUD profiles," *Symmetry*, vol. 12, no. 9, p. 1576, Sep. 2020.
[135] D. Kim, V. Andalibi, and L. J. Camp, "Fingerprinting edge and cloud services in IoT," in *Proc. 13th Int. Conf. Systematic Approaches Digit. Forensic Eng. (SADFE)*, May 2020, pp. 13–21.
[136] A. Osman, A. Wasicek, S. Köpsell, and T. Strufe, "Transparent microsegmentation in smart home IoT networks," in *Proc. 3rd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2020, pp. 1–6.
[137] M. McCormack, A. Vasudevan, G. Liu, S. Echeverría, K. O'Meara, G. Lewis, and V. Sekar, "Towards an architecture for trusted edge IoT security gateways," in *Proc. 3rd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2020, pp. 1–10.
[138] B. Momenzadeh, H. Dougherty, M. Remmel, S. Myers, and L. J. Camp, "Best practices would make things better in the IoT," *IEEE Secur. Privacy*, vol. 18, no. 4, pp. 38–47, Jul. 2020.
[139] A. Hamza, H. H. Gharakheili, and V. Sivaraman, "IoT network security: Requirements, threats, and countermeasures," 2020, *arXiv:2008.09339*. [Online]. Available: http://arxiv.org/abs/2008.09339
[140] V. Andalibi, D. Kim, and L. J. Camp, "Throwing MUD into the FOG: Defending IoT and fog by expanding MUD to fog network," in *Proc. 2nd USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2019, pp. 1–6.
[141] D. Rawat, C. Brecher, H. Song, and S. Jeschke, *Industrial Internet of Things: Cybermanufacturing Systems*. Cham, Switzerland: Springer, 2017.
[142] OpenFog Consortium Architecture Working Group and Others, "Openfog reference architecture for fog computing," in *Proc. Archit. Work. Group*, 2017, pp. 1–162.
[143] Industrial Internet Consortium. (2015). *Industrial Internet Reference Architecture*. Accessed: Aug. 29, 2018. [Online]. Available: http://www.iiconsortium.org/IIRA.htm
[144] T. Gebremichael, L. P. I. Ledwaba, M. H. Eldefrawy, G. P. Hancke, N. Pereira, M. Gidlund, and J. Akerberg, "Security and privacy in the industrial Internet of Things: Current standards and future challenges," *IEEE Access*, vol. 8, pp. 152351–152366, 2020.
[145] 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, no. 21, p. 4576, Oct. 2019.

[146] Y. Meidan, V. Sachidananda, H. Peng, R. Sagron, Y. Elovici, and A. Shabtai, "A novel approach for detecting vulnerable IoT devices connected behind a home NAT," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101968.
[147] C. Ye, P. P. Indra, and D. Aspinall, "Retrofitting security and privacy measures to smart home devices," in *Proc. 6th Int. Conf. Internet Things: Syst., Manage. Secur. (IOTSMS)*, Oct. 2019, pp. 283–290.
[148] P. Pop, B. Zarrin, M. Barzegaran, S. Schulte, S. Punnekkat, J. Ruh, and W. Steiner, "The FORA fog computing platform for industrial IoT," 2020, *arXiv:2007.02696*. [Online]. Available: http://arxiv.org/abs/2007.02696
[149] S. As, "Architecture analysis and design language (AADL)," Embedded Comput. Syst. Committee, SAE, Warrendale, PA, USA, Tech. Rep. SAE AS5506, 2004.
[150] M. De Donno, J. M. D. Felipe, and N. Dragoni, "ANTIBIOTIC 2.0: A fog-based anti-malware for Internet of Things," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS PW)*, Jun. 2019, pp. 11–20.
[151] A. Feraudo, P. Yadav, V. Safronov, D. A. Popescu, R. Mortier, S. Wang, P. Bellavista, and J. Crowcroft, "CoLearn: Enabling federated learning in MUD-compliant IoT edge networks," in *Proc. 3rd ACM Int. Workshop Edge Syst., Analytics Netw.*, Apr. 2020, pp. 25–30.
[152] *5 November 2020 23:54 UTC*. Accessed: Jan. 1, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Federated_learning&oldid=987270512
[153] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 756–767.
[154] M. Alencar, R. Barreto, H. Fernandes, E. Souto, and R. Pazzi, "DARE: A decentralized association rules extraction scheme for embedded data sets in distributed IoT devices," *Int. J. Distrib. Sensor Netw.*, vol. 16, no. 10, 2020, Art. no. 1550147720962999.
[155] G. Matthíasson, A. Giaretta, and N. Dragoni, "IoT device profiling: From MUD files to $S \times C$ contracts," in *Proc. Open Identity Summit*, 2020, p. 305.
[156] C. Gaber, J. S. Vilchez, G. Gur, M. Chopin, N. Perrot, J.-L. Grimault, and J.-P. Wary, "Liability-aware security management for 5G," in *Proc. IEEE 3rd 5G World Forum (5GWF)*, Sep. 2020, pp. 133–138.
[157] R. Neisse, J. L. Hernandez-Ramos, S. N. Matheu, G. Baldini, and A. Skarmeta, "Toward a blockchain-based platform to manage cybersecurity certification of IoT devices," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Oct. 2019, pp. 1–6.
[158] M. Iansiti and K. R. Lakhani, "The truth about blockchain harvard business review," Harvard Univ., Cambridge, MA, USA, Tech. Rep. BR1701, 2017. Accessed: Feb. 2, 2019. [Online]. Available: http://www.org/2017/01/the-truth-about-blockchain
[159] R. Neisse, J. L. Hernandez-Ramos, S. N. Matheu-Garcia, G. Baldini, A. Skarmeta, V. Siris, D. Lagutin, and P. Nikander, "An interledger blockchain platform for cross-border management of cybersecurity information," *IEEE Internet Comput.*, vol. 24, no. 3, pp. 19–29, May 2020.
[160] J. L. Hernandez-Ramos, G. Baldini, S. N. Matheu, and A. Skarmeta, "Updating IoT devices: Challenges and potential approaches," in *Proc. Global Internet Things Summit (GIoTS)*, Jun. 2020, pp. 1–5.
[161] B. Moran, M. Meriac, H. Tschofenig, and D. Brown, *A Firmware Update Architecture for Internet of Things Devices*, document Internet-Draft draft-moran-suit-architecture-02, IETF, 2019.
[162] 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 Secur. Privacy*, Aug. 2018, pp. 8–14.
[163] S. M. Sajjad, M. Yousaf, H. Afzal, and M. R. Mufti, "EMUD: Enhanced manufacturer usage description for IoT botnets prevention on home WiFi routers," *IEEE Access*, vol. 8, pp. 164200–164213, 2020.
[164] S. N. Matheu, A. R. Enciso, A. M. Zarca, D. Garcia-Carrillo, J. L. Hernández-Ramos, J. B. Bernabe, and A. F. Skarmeta, "Security architecture for defining and enforcing security profiles in DLT/SDN-based IoT systems," *Sensors*, vol. 20, no. 7, p. 1882, Mar. 2020.
[165] A. M. Zarca, J. B. Bernabe, R. Trapero, D. Rivera, J. Villalobos, A. Skarmeta, S. Bianchi, A. Zafeiropoulos, and P. Gouvas, "Security management architecture for NFV/SDN-aware IoT systems," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8005–8020, Oct. 2019.
[166] S. N. Matheu, J. L. Hernandez-Ramos, S. Perez, and A. F. Skarmeta, "Extending MUD profiles through an automated IoT security testing methodology," *IEEE Access*, vol. 7, pp. 149444–149463, 2019.
[167] C. Bormann and P. Hoffman, *Concise Binary Object Representation (CBOR)*, document RFC 7049, Oct. 2013.
[168] *Extensible Access Control Markup Language (XACML) Version 3.0*, Standard XACML Version 3.0, 2013.
[169] G. Selander, J. Mattsson, and F. Palombini, *Ephemeral Diffie-Hellman Over Cose (EDHOC)*, document IETF, ID draft-selander-lake-edhoc-00, 2019.
[170] P. Foremski, S. Nowak, P. Fröhlich, J. Hernández-Ramos, and G. Baldini, "Autopolicy: Automated traffic policing for improved IoT network security," *Sensors*, vol. 20, no. 15, p. 4265, Jul. 2020.
[171] M. Al-Shaboti, A. Chen, and I. Welch, "Automatic device selection and access policy generation based on user preference for IoT activity workflow," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 769–774.
[172] A. Bremler-Barr, H. Levy, and Z. Yakhini, "IoT or NoT: Identifying IoT devices in a short time scale," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2020, pp. 1–9.
[173] A. Feraudo, P. Yadav, R. Mortier, P. Bellavista, and J. Crowcroft, "SoK: Beyond IoT MUD deployments—Challenges and future directions," 2020, *arXiv:2004.08003*. [Online]. Available: http://arxiv.org/abs/2004.08003
[174] P. Yadav, V. Safronov, and R. Mortier, "Enforcing accountability in smart built-in IoT environment using MUD," in *Proc. 6th ACM Int. Conf. Syst. Energy-Efficient Buildings, Cities, Transp.*, Nov. 2019, pp. 368–369.
[175] J. Wang, S. Hao, R. Wen, B. Zhang, L. Zhang, H. Hu, and R. Lu, "IoT-praetor: Undesired behaviors detection for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 927–940, Jan. 2021.
[176] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Managing IoT cyber-security using programmable telemetry and machine learning," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 1, pp. 60–74, Mar. 2020.
[177] A. Singh, S. Murali, L. Rieger, R. Li, S. Hommes, R. State, G. Ormazabal, and H. Schulzrinne, "HANZO: Collaborative network defense for connected things," in *Proc. Princ., Syst. Appl. IP Telecommun. (IPTComm)*, Oct. 2018, pp. 1–8.
[178] 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*. Cham, Switzerland: Springer, 2019, pp. 43–57.
[179] A. Hamza. *Mudgee*. Accessed: 2018. [Online]. Available: https://github.com/ayyoob/mudgee

NOMAN MAZHAR received the B.E. degree in software engineering and the M.S. degree in information technology from the National University of Science and Technology (NUST), Islamabad, Pakistan, in 2002 and 2011, respectively. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. He has worked in the industry and academia. He is currently working as a Graduate Research Assistant with the Center for Research in Industry 4.0, University of Malaya. His research interests include industry 4.0, the Internet of Things, information security, software-defined networks, machine learning, and wireless networking. He has been awarded the HEC International Fellowship for Ph.D. studies, in 2019.

ROSLI SALLEH received the B.Sc. degree in computer science from the University of Malaya, in 1994, and the master's degree in data communication and the Ph.D. degree from the University of Salford, U.K., in 1997 and 2001, respectively. After finished his Ph.D. studies, he started as a Junior Lecturer and was appointed to the current position as an Associate Professor in 2013. Until 2019, he has supervised to completion of nine Ph.D. students and more than 30 master students. He is currently an Associate Professor with the Department of Computer Systems and Technology, Faculty of Computer Science and Information Technology, University of Malaya. He has published more than 30 ISI articles in various ISI journals. His main research interests include software defined networking, mobile IP, and network security. He has received more than RM600K Grants, many awards, and recognitions, and was appointed as a Panel in many national and international institutions.

MUHAMMAD ZEESHAN (Senior Member, IEEE) received the M.Sc. degree in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2009, and the Ph.D. degree in information technology from the National University of Science and Technology (NUST), Islamabad, Pakistan, in 2016. He bears experience both in industry and teaching, earlier he worked as an Assistant Professor with Iqra University, Islamabad, for four years. He also worked as a Design Engineer, a Senior Design Engineer, a Project Manager, and a Research Fellow at the research and development industry. He is currently working as an Assistant Professor with the School of Electrical Engineering and Computer Science (SEECS-NUST). His main research interests include wireless networks, networks data analysis, and the Internet of Things. He was awarded the HEC Indigenous Fellowship for M.S. degree leading to Ph.D. studies, in 2007 and 2015, and also visited Michigan State University, USA, for six months, funded by the HEC IRSIP Fellowship Award.

M. MUZAFFAR HAMEED received the bachelor's degree in computer science from Bahauddin Zakariya University (BZU), Multan, Pakistan, and the master's degree in software engineering from the Blekinge Institute of Technology (BTH), Karlskarona, Sweden. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. He has 12 years of experience related to software development, teaching, and research. He is also working as an Assistant Professor with BZU. He has been awarded the Foreign Ph.D. Scholarship by HEC, Pakistan. His research interests include machine learning, deep learning, digital forensics, and information security.