IoT MUD Enforcement in the Edge Cloud Using Programmable Switch

Harish S A
IIT Hyderabad, India

Hemanth Kothapalli
IIT Hyderabad, India

Shubham Lahoti
IIT Hyderabad, India

Kotaro Kataoka
IIT Hyderabad, India

Praveen Tammana
IIT Hyderabad, India

ABSTRACT

Targeted data breaches and cybersecurity attacks involving IoT devices are becoming ever more concerning. To combat these threats and risks, the IETF standardized Manufacturer Usage Description (MUD), which allows IoT device vendors to specify the intended communication patterns (MUD profile) of an IoT device. MUD profile enables validation of the actual communication pattern of an IoT device with the intended behavior at run-time. However, the MUD specification was primarily intended for enforcement at the Local Area Network (LAN) of the IoT device, thus fragmenting the solution across multiple heterogeneous networks. MUD enforcement at higher levels in the network hierarchy (*e.g.*, private edge for enterprise networks) eases security policy management and reduces processing overheads on the existing security infrastructure.

To realize MUD enforcement at the edge, there are mainly two challenges: (1) How to identify an IoT device at the edge so that enforcing device-specific MUD profile on the IoT traffic is possible. (2) How to scale MUD enforcement to a large network of IoT devices. In this paper, we present our approach to address these challenges and validate IoT device communication at the edge. In order to scale MUD enforcement to a large IoT network, we leverage multi-stage pipeline architecture and stateful ALUs of P4 programmable switch and process IoT traffic in the dataplane.

CCS CONCEPTS

• **Networks** → **Programmable networks**; **Network monitoring**; **In-network processing**; • **Security and privacy** → **Firewalls**.

KEYWORDS

Programmable networks, Internet of Things, Network security, Manufacturer Usage Description.

ACM Reference Format:
Harish S A, Hemanth Kothapalli, Shubham Lahoti, Kotaro Kataoka, and Praveen Tammana. 2022. IoT MUD Enforcement in the Edge Cloud Using Programmable Switch. In *ACM SIGCOMM 2022 Workshop on Formal Foundations and Security of Programmable Network Infrastructures (FFSPIN '22), August 22, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3528082.3544832

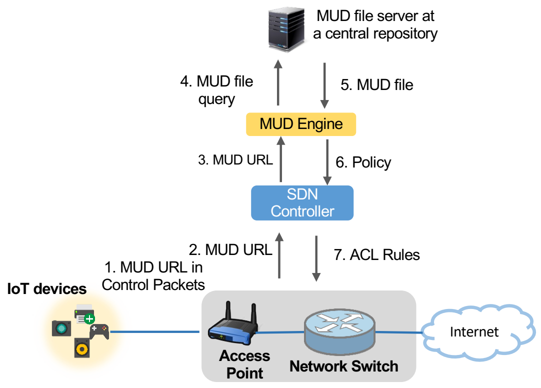


Figure 1: Enforcing MUD profiles using SDN

1 INTRODUCTION

The Internet of Things (IoT) connect a massive number of smart devices enabling services in a wide range of application domains such as e-governance, agriculture, transportation, education, smart-homes, health care, and e-shopping. Despite the advantages of IoT, the majority of IoT device manufacturers do not pay much-needed attention to security, thus increasing the attack surface. Adversaries can exploit device vulnerabilities and launch network-based attacks that have serious negative implications for critical infrastructure (*e.g.*, DDoS attack [28], Mirai [18], VPNFilter [26]).

The main pain point is that we cannot protect what we cannot see. More specifically, a few concerns of IoT device owners are: to which entities an IoT device is talking to? what kind of data the IoT device is transferring to those entities? To address this problem, the National Cybersecurity Center of Excellence (NCCoE) advocates Manufacturer Usage Description (MUD) to mitigate network-based attacks [4]. The key idea is to let device manufacturers formally specify the communication pattern of an IoT device, called the MUD profile. Network operators would use MUD profiles as a reference and ensure that the IoT device is indeed talking to intended entities (*e.g.*, IoT device manufacturer servers, devices within the LAN, other IoT devices from the same manufacturer).

Incentives for MUD enforcement at higher levels in network hierarchy. The MUD specification focused on the possibility of enforcing the rules of the concerned IoT devices at the local network such as WiFi access point and Customer Premise Equipment (CPE) [9] where IoT device identity (*e.g.*, MAC address) is available as in Figure 1. Consequently, the majority of works [19–21, 25] in this space also concentrate on frameworks and methods involving MUD that function within the local network. However, MUD enforcement at an elevated vantage point higher in the network hierarchy can present unique opportunities from an administrative viewpoint. More specifically, the overheads of existing security infrastructure (*i.e.*, middleboxes) can be greatly reduced by enforcing MUD at the edge. For instance, MUD can filter unintended traffic with destination IPs not specified in MUD profile such that a DDoS detection system would only need to monitor IoT traffic destined to IP addresses specified by MUD. This will reduce memory overheads of DDoS monitoring and detection systems. Similarly, MUD-based filtering can reduce bandwidth and processing overheads of Deep Packet Inspectors (DPI) by inspecting only unintended traffic (*e.g.*, destination IP not specified in MUD).

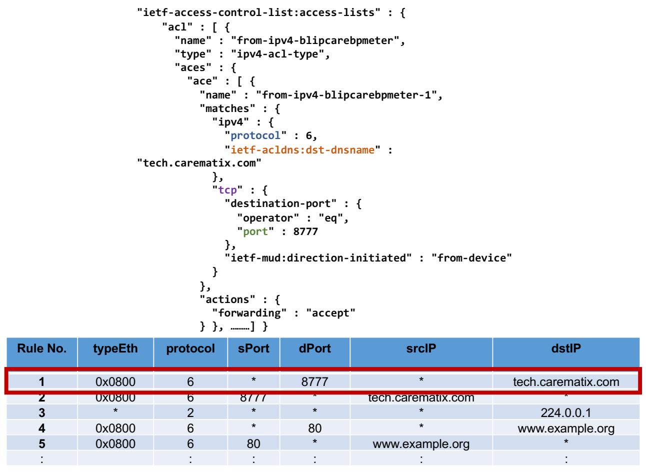
Challenges. To realize this idea, we observe that identifying individual IoT devices beyond the local network (*i.e.*, WiFi access points) is challenging. First, traffic originating from the IoT device loses its identifying factor (*e.g.*, source MAC address) outside the local network. This implies an increased difficulty for applying the rules specified in an IoT device MUD file. That is, we have to classify IoT and non-IoT traffic, then dissect the IoT traffic (based on types of devices), and then apply device-specific MUD rules. Moreover, Network Address Translator (NAT) rewrites both the source IP address and the port number. Therefore all traffic will appear to have the same source IP address (*i.e.*, WiFi access point IP or CPE IP). Second, enforcing MUD rules at higher levels of network hierarchy means that MUD enforcement system needs to scale to support a large number of IoT devices and process high volumes of traffic.

The first work that proposes a framework to implement MUD-based filtering beyond local network [10, 11] does so using on-path ISP core switch and off-path Virtual Network Function (VNF). But this approach has large resource overheads as it invokes off-path VNF for every new connection from each IoT device.

In this paper, we present our approach and system design for enforcing IoT device-specific MUD rules in the edge network. Our system first identifies IoT devices and their type, and then applies MUD rules specific to an IoT device type. The system can be adapted and deployed at ISP edge for home networks, or at the private edge for campus and enterprise networks. The main ideas are: (1) For IoT device identification, we rely on packet marking capability of CPEs or WiFi access points and stateful packet-processing features of P4 programmable switch (more details in §3.2, §3.3). (2) For scaling MUD enforcement to a large number of IoT devices, we leverage the multi-stage pipeline of P4 programmable switch and design an SRAM-based packet classification algorithm (more details in §3.4).

2 BACKGROUND

IoT devices. It is expected that the number of IoT devices used worldwide will exceed 24 billion by 2030 [16]. The work done by [24] provides a fine-grained view into the IoT device demographic. They report that out of a total 14.3k unique vendors, 90% of all devices globally are manufactured by only 100 of them. This observation is relevant to our work, where we can conservatively state that the endpoints that IoT devices talk to may repeat across devices from the same vendor. For example, different devices from the same vendor may contact the same remote endpoint. In this case, the number of MUD rules required may have overlapping subsets.



```
"ietf-access-control-list:access-lists" : {
    "acl" : [ {
      "name" : "from-ipv4-blipcarebpmeter",
      "type" : "ipv4-acl-type",
      "aces" : {
        "ace" : [ {
          "name" : "from-ipv4-blipcarebpmeter-1",
          "matches" : {
            "ipv4" : {
              "protocol" : 6,
              "ietf-acldns:dst-dnsname" :
"tech.carematix.com"
            },
            "tcp" : {
              "destination-port" : {
                "operator" : "eq",
                "port" : 8777
              },
              "ietf-mud:direction-initiated" : "from-device"
            }
          },
          "actions" : {
            "forwarding" : "accept"
          } }, ……] }
```

Rule No.	typeEth	protocol	sPort	dPort	srcIP	dstIP
1	0x0800	6	*	8777	*	tech.carematix.com
2	0x0800	6	8777	*	tech.carematix.com	*
3	*	2	*	*	*	224.0.0.1
4	0x0800	6	*	80	*	www.example.org
5	0x0800	6	80	*	www.example.org	*
:		:	:	:	:	:

Figure 2: MUD file to ACL rules

IoT device discovery and MUD profile retrieval. An IoT device can be uniquely identified using its MAC address (also called hardware address) on the local network. According to the MUD standard [4], the HTTPS MUD URL from the IoT device is carried through either DHCP, LLDP or an X.509 constraint. A network device in the middle parses the URL, and its corresponding signed MUD file is downloaded from the MUD server. In our system design, we assume the use of DHCP discovery messages containing either the option 161 in case of IPv4 or 112 in the case of IPv6 where the MUD URL is encoded as a string. Typically, a network device can be configured to forward DHCP requests to an SDN controller, and the controller will download the MUD profile from the manufacturer's website or a central repository.

MUD profiles. The MUD specification [4] explicitly characterizes the network behaviour of IoT devices. Contained in the JSON formatted MUD file are key-value pairs, which are intended to be processed as ACL rules for enforcement (highlighted with colours in Figure 2). More specifically, the MUD profile has MUD access control entries (ACEs) which specify the expected behaviour of the device. For example, the profile may state that the device can communicate with a specific set of domains (e.g., google, amazon, NTP servers, device vendors, etc.) on a specific port, or/and devices that belong to the same manufacturer, or/and devices connected to the same LAN. Figure 2 shows an example MUD profile and associated access control rule (rule 1), where the rule is a tuple with 6 fields: typEth, protocol, sPort, dPort, srcIP and, dstIP. The ($*$) represents a wildcard character signifying that header field could take any arbitrary value. In our work, we are only concerned with the layer 3 header fields due to visibility constraints outside the LAN.

UNSW IoT device MUD Profiles. The work done in [21] has been instrumental for the proliferation of MUD-based IoT security solutions. A total of 28 real IoT devices have been profiled and published at [2]. The subset of rules that specify device behaviour on the LAN are not relevant to our system. Among the other rules, we observe that the highest count of forward traffic rules (traffic originating from the IoT device) is 102 (seen in Chromecast Ultra).

Further observing the MUD ACEs, the protocol field is always specified for rules addressing non-local communication (*i.e.*, remote endpoint domain name or IP address). Port and IP address details may contain don't cares (∗) in certain cases involving ICMP (protocol 1) and IGMP (protocol 2) or DNS (dPort 53). We explain in §3.4 the nuances involved in scaling the number of rules stored per programmable switch.

Network Structure. As shown in Figure 3, to access the internet, a CPE (*e.g.*, WiFi access point) is placed on the LAN which typically marks its periphery [9]. The CPEs are directly configurable by the network edge through the TR-069 management protocol [22]. They periodically initiate communication with an Auto-Configuration Server (ACS) through a HTTP-based client-server protocol that treats the CPEs as the client. The ACS also exposes a northbound interface, through which remote applications (SDN controller in our case) can control the managed devices (*i.e.*, WiFi access points). For instance, the controller can send configuration requests (through ACS) to the CPE to configure parameters (*e.g.*, IPtables, DNS, etc.).

3 SYSTEM DESIGN

3.1 Overview

Our main idea is to enforce IoT MUD files at the edge. This means the MUD enforcement system has to scale to a large network with many CPEs and many IoT device types across CPEs. One approach is to distribute MUD enforcement load across multiple virtual network functions (VNFs). But this approach has high operational costs (*e.g.*, management, CPU processing costs) and introduces delays. In this paper, we propose a system design that converts MUD files to whitelist ACL rules at an SDN controller and installs them on a P4 programmable switch placed at the edge. However, to implement this design in a real network, there are a few challenges that we will address in this paper.

Firstly, each device type has its own MUD file, so before applying device-specific MUD rules, our system has to identify IoT device and its type. The challenge arises due to the non-visibility of IoT device MAC address (identifier) outside the local network, that is, beyond CPE. We address this problem by forwarding DHCP packets with IoT device MAC address from CPE to an SDN controller. Next, we instruct the CPE to mark subsequent data traffic (from that IoT) such that the marking enables the identification of IoT device type in the edge network. We use DSCP bits for packet marking and custom values to determine IoT device type. This approach is similar to the one proposed in [10].

Secondly, though the device identification is made possible for traffic in forward direction via packet marking at CPEs, the traffic in backward direction from a remote endpoint may not be marked. This is because, the remote endpoints are typically not under the control of the edge network to perform any kind of marking. Thus, identifying MUD rules for reverse traffic becomes a challenge. To address this, in the switch dataplane, we keep track of connections initiated from IoT devices in the forward direction and lookup these connections while processing traffic in the backward direction. If found, the associated MUD rules for traffic towards IoT devices are applied. To realise this, we leverage hash constructs and stateful registers and design a bloom filter-based data structure which is updated by forward traffic and queried by backward traffic.

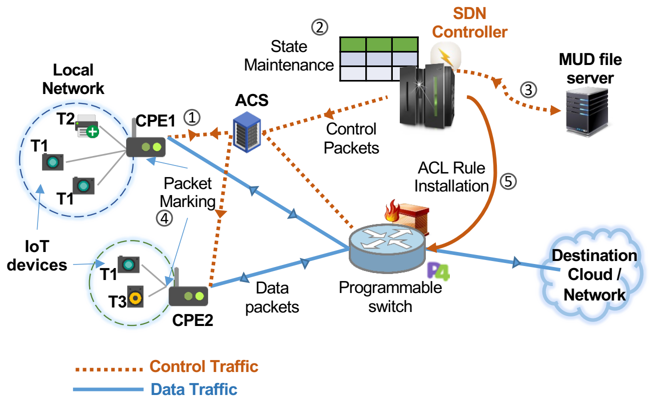


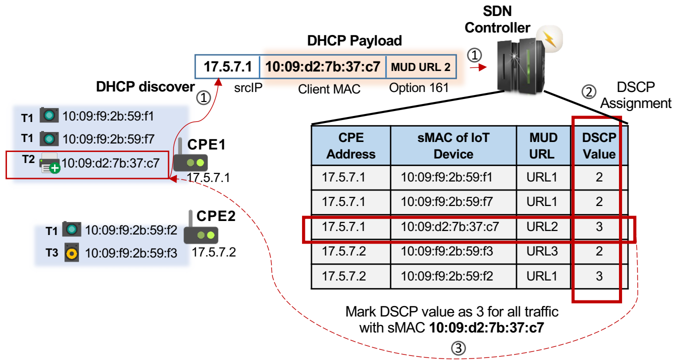
Figure 3: MUD enforcement at the edge using P4 switch

Finally, we observe that many header fields in MUD-based ACL rules have few distinct values. This means placing ACL rules in a single large and wide TCAM table is inefficient as the same header values are repeated across multiple entries. Also, header values are only of two types, either exact or don't care (∗), but no prefix-based values (*e.g.*, /24). This means fast parallel lookup supported by TCAM can be replaced with SRAM-based exact match tables. Based on these observations, we propose a decision tree based technique that carefully splits header fields and places their values (exact) in multiple switch stages so that SRAM memory associated to the stage can be leveraged. Compared to using TCAM for match-action table, this approach supports more CPEs and IoT device types.

3.2 Enrollment of IoT devices

Our design is meant to scale for a large number of CPEs but we restrict a generalized representative example using two CPEs (CPE1 and CPE2) shown in Figure 3. Through this example, we try to cover all scenarios. There are three devices in CPE1 (T1, T1 and T2) and two devices in CPE2 (T1, T3). We repeat the annotation to denote that two individual devices are of the same type. The edge network consists of components like Auto-Configuration Server (ACS), SDN controller, and the P4 programmable switch. As mentioned earlier, consider that IoT device MAC address is not visible outside the CPE boundaries and the traffic going out of a CPE has CPE's IP address as the source IP address. This section explains our approach to classifying IoT traffic followed by the identification of device type.

Enrollment at the control plane. Consider that an IoT device informs the network of its underlying MUD URL via DHCP options. Note that MUD URL is the device type identifier and all devices of the same type have the same MUD URL. We configure all the CPEs in the network to mirror a copy of all DHCP packets to the controller through the ACS. This is accomplished through the management capabilities exposed by the TR-069 protocol, which creates IP table records to forward all DHCP packets to our SDN controller. Since a DHCP packet inherently contains the MAC address of the IoT device (as part of the payload), we obtain important telemetry details for a particular IoT device. Figure 4 shows the flow of events and the telemetry data received through the DHCP packet: (1) Source IP address of the CPE Gateway (from DHCP IP packet) (2) IoT device MAC address (present in the DHCP payload) and (3) Device type (through the MUD URL in options of DHCP payload).



CPE Address	sMAC of IoT Device	MUD URL	DSCP Value
17.5.7.1	10:09:f9:2b:59:f1	URL1	2
17.5.7.1	10:09:f9:2b:59:f7	URL1	2
17.5.7.1	10:09:d2:7b:37:c7	URL2	3
17.5.7.2	10:09:f9:2b:59:f3	URL3	2
17.5.7.2	10:09:f9:2b:59:f2	URL1	3

Figure 4: Enrollment and state maintenance

Packet marking at CPE. The controller, now on receiving the DHCP packets, maintains a record of CPE address, the device type, and it's MAC address. The table present in Figure 4 shows the mappings corresponding to the two CPEs in the example. We mark packets using the 6-bit Diffserv DSCP field. We observe that only 23 values are reserved and in use as per RFCs [3]. We make use of the remaining 41 values. Each of the 41 custom DSCP values will denote a device type in a CPE. Further, the controller assigns unused DSCP values to each new device type it encounters (a new DSCP value for every new MUD URL).

DSCP values are assigned on a first come first serve basis as seen in the fourth column of the table in Figure 4. We also note that the value assigned is endemic to each CPE IP address. For example, an IoT device of type T1 in CPE1 could be mapped to DSCP value 2 and the same device type in CPE2 could be mapped to either 2 or 3 depending on the order of enrollment in that CPE. In Figure 4, the value of 2 has been assigned for device type T1 in CPE1, whereas the same device type has been assigned a value of 3 in CPE2. The idea is to differentiate between the types of devices within a CPE. Essentially, each CPE maintains a separate set of DSCP markings for different device types.

Once mapped, the CPE is instructed to mark all traffic (with custom DSCP values) from a particular IoT device type based on it's MAC address. The TR-069 management protocol can configure CPEs to mark packets based on the source MAC address of the IoT (as per mapping assigned at the control plane). Therefore, all traffic originating from a particular CPE are essentially partitioned into marked and non-marked packets based on their DSCP value. They respectively correspond to the IoT that supports MUD and other traffic (non-IoT or IoT without MUD support). Moreover, we gain visibility over the type of IoT device the traffic belongs to. At this point in time, we can claim that an IoT device has been identified and enrolled within our system. The traffic is now processed at the network edge as per the markings and CPE identity.

3.3 MUD rule enforcement

Simultaneous to the creation of the mapping described earlier, the corresponding MUD server is queried using the recorded MUD URL (from DHCP options). Thus, the controller also maintains a repository of MUD files corresponding to each device type. The MUD profile entries are converted to MUD ACL rules (details in the next subsection) as per the relevant access control entries. As mentioned earlier, the destination endpoints (*i.e.*, source or destination IP address) in the MUD ACL rules could be domain names as well. Thus, resolving them at run-time leads to multiple addresses per domain (we handle this scenario later).

The final ACL rules are then formulated as a combination of the assigned DSCP value (Table 2 in Figure 5) and the freshly derived MUD rules as shown in Table 1 in Figure 5. We classify the bi-directional IoT traffic as forward (egress from IoT device) and backward (reply traffic ingress to IoT device). Table 1 in Figure 5 contains one MUD ACL rule each for forward traffic (rule 1) and backward traffic (rule 2) at the control plane.

Forward traffic. From the sample traces and traffic from real IoT devices, we observe most of the connections are initiated by IoT devices and uses either TCP or UDP as the underlying protocol. Therefore, the dataplane uses the following header fields from the forward traffic for match-action: (1) source IP address (CPE IP address) (2) DSCP value (contains device type marking) (3) protocol (4) destination port and (5) destination IP.

In order for the dataplane to maintain state and keep track of forward and backward flows, we use bloom filters [15]. Bloom filters are space efficient data structures used to test the membership of an element in a set. Therefore they are suitable for resource-constrained dataplanes (*e.g.*, memory) compared to a regular hash table. On packet arrival, we hash 4-tuple (srcIP, protocol, dPort, dstIP) and the bit(s) in bloom filter indexed by the hash value is set to 1. The packet subsequently traverses the MUD MAT (Table 3 in Figure 5) where the validity of the IoT traffic is determined using installed MUD ACL rules (by matching header fields described above). Most importantly, the above steps are applied only if the DSCP value is a custom value (marked by CPE). Else, the traffic is forwarded normally (signifies non-IoT traffic).

The CPE IP address and DSCP values are the primary identifiers and are rightly placed at the first two stages of the MAT for forward traffic. The next three stages contain values directly extracted from the MUD file entries. We thus, classify the MUD rules according to the device type (identified by the DSCP marking) which are in turn classified according to the CPE source IP address. The hierarchical nature of the rules could be exploited to optimize the repeated occurrences and thus elicit memory savings (described in §3.4).

The bloom filter size can be determined based on the maximum bound on the number of IoT traffic flows that is expected out of a particular CPE. Earlier we reported a maximum of 102 forward rules for one of the 28 IoT devices we examined [2]. Fixing this as the upper bound for number of rules per IoT device (in forward direction), we calculate the flow upper bound based on the remaining 41 custom values used to mark IoT packets. Thus, the maximum number of flows expected per CPE is 41 * O (maximum number of MUD rules) which in turn translate to 41 * 102 totalling 4182 flows. We can then estimate the total number of rules (flows) required to be handled depending on the number of CPEs an ISP/edge network manages. For example, an ISP handling 100 CPEs would need to store 418200 rules in the worst case.

We however note that hash calculation does not involve the DSCP mark and thus, the number of active flows stay well below the calculated upper bound. To reduce false-positive rate, we can use multiple bloom filter bits indexed by multiple hash values.

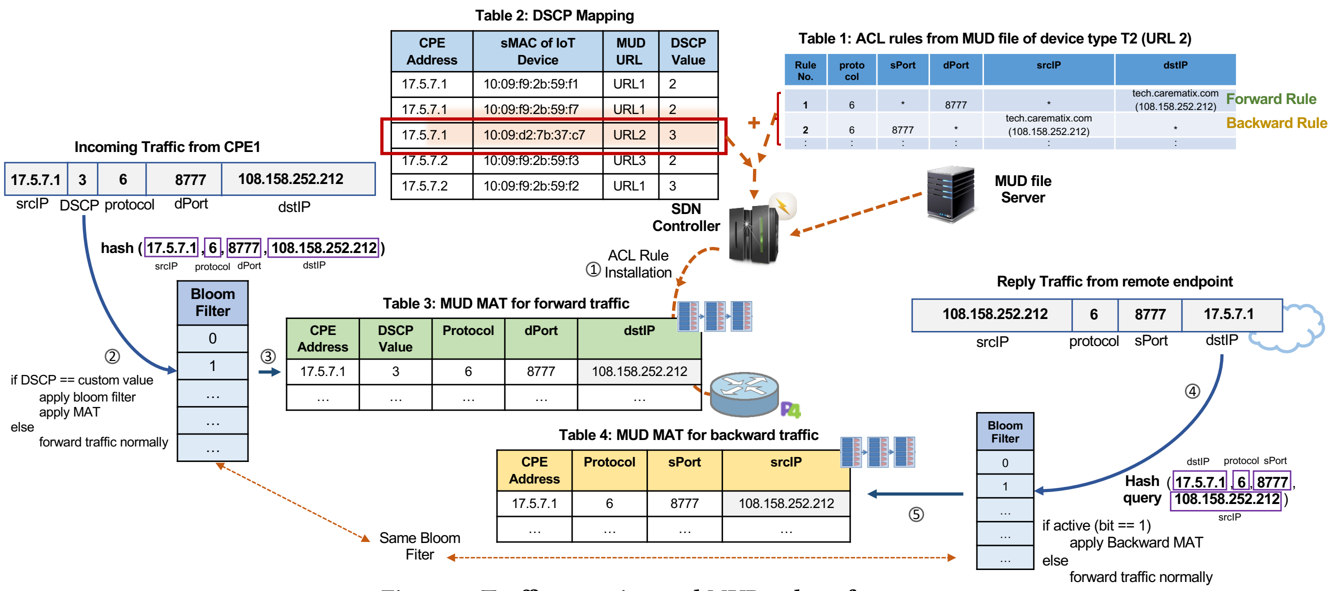


Table 2: DSCP Mapping

CPE Address	sMAC of IoT Device	MUD URL	DSCP Value
17.5.7.1	10:09:f9:2b:59:f1	URL1	2
17.5.7.1	10:09:f9:2b:59:f7	URL1	2
17.5.7.1	10:09:d2:7b:37:c7	URL2	3
17.5.7.2	10:09:f9:2b:59:f3	URL3	2
17.5.7.2	10:09:f9:2b:59:f2	URL1	3

Table 1: ACL rules from MUD file of device type T2 (URL 2)

Rule No.	proto col	sPort	dPort	srcIP	dstIP
1	6	*	8777	*	tech.carematix.com (108.158.252.212)
2	6	8777	*	tech.carematix.com (108.158.252.212)	*
:	:	:	:	:	:

Table 3: MUD MAT for forward traffic

CPE Address	DSCP Value	Protocol	dPort	dstIP
17.5.7.1	3	6	8777	108.158.252.212
…	…	…	…	…

Table 4: MUD MAT for backward traffic

CPE Address	Protocol	sPort	srcIP
17.5.7.1	6	8777	108.158.252.212
…	…	…	…

Figure 5: Traffic mapping and MUD rule enforcement

However, there could be chances of collisions where two flows could map to the same index in the bloom filter. With respect to our scenario we enumerate the possibilities: (1) IoT traffic originating from different CPEs from the same device are always differentiated by the CPE source IP address (thus different hash value). (2) Traffic from two same device types in the same CPE has high collision probability and rightly so as the set of MUD rules to be applied would be same for devices of same type. (3) Traffic from two different IoT device types from the same source CPE has a probability of collision only if the same port, protocol and destination endpoints are used (two different devices from the same vendor). Moreover, the bloom filter has to be reset periodically by control plane.

Backward traffic. A packet in the backward direction queries the bloom filter to check whether the switch has seen associated flow in the forward direction. If so, we apply MUD MAT for backward traffic (Table 4 in Figure 5). Otherwise, we allow the traffic. We note that the bloom filter query is based on the following packet headers: destination IP (contains CPE IP address), protocol, source port, and source IP address (contains the remote endpoint IP address).

For backward traffic, if corresponding bloom filter bit is set, we apply backward MAT for all traffic (irrespective of whether it is going to IoT or non-IoT). To understand this, let us consider two devices, one IoT and one non-IoT device both talking to the same endpoint (from the same CPE). In this case, the backward traffic will be identical with respect to the hash inputs (header fields) we use. Since the endpoint is most likely whitelisted (by implication from forward MAT), the traffic is allowed in the backward direction by backward MAT. However, false positives are possible for non-IoT device traffic, that is, non-IoT traffic is misclassified as IoT traffic. In this case, if there is no matching entry in the backward MAT, which is most likely the case, then this traffic is treated as suspicious and sent to the control plane for further analysis (slow path). To reduce such events, we can use either multiple hash functions to reduce the false-positive rate or keep track of connections of non-IoT devices using another bloom filter. In our future work, we plan to study the tradeoff among bloom filter memory, false-positive rate, and reset frequency for different IoT workloads.

3.4 MUD-based rules in MAT

In the MUD rules for 28 IoT devices [2], we observe port number and IP address fields contain don't cares ($*$) in certain cases to allow protocol traffic like DNS, ICMP and IGMP. For example, consider that an IoT device sends DNS requests to a remote endpoint whose exact destination IP or domain name is not specified in the MUD profile in advance. In this case, the destination IP address could be $*$. Similarly, ICMP traffic (protocol 1) should be allowed with dport and dstIP set to $*$. In a P4 program, when we define a common table for rules with both don't care values (to allow protocol traffic) and other MUD rules with exact values (to allow data traffic), we notice that TNA-based tofino model [12] switch uses TCAM memory.

We see from the ACL rule table in Figure 2 that protocol values are repeated. Similarly, CPE address and DSCP values (Table 3 in Figure 5) also recur. Therefore, we are not efficient when we repeat values in a TCAM-based single table implementation of MUD ACL rules. Thus, using TCAM for rule enforcement (at the network edge) for a large network would not scale well as available TCAM memory is limited, and is usually shared with other core network functions (*e.g.*, routing). As an alternative, we explore SRAM-based packet classification algorithm that runs entirely in the switch data plane. To be specific, the multi-stage programmable switch pipeline could be used to carefully place ACL rules in multiple SRAM-based exact match tables. By doing so, we can scale MUD rule enforcement to a large number of IoT devices.

It is well known that decision trees are often used to represent functions on a wide input domain and save memory by avoiding common path repetition. We expect to exploit this property and represent ACL rules in a decision tree followed by encoding the decision tree in exact match tables (*i.e.*, SRAM memory) in the data plane. The basic idea of using a decision tree in the control plane is not new by itself [27], but the realization that we can tilt the data structure sideways in the P4 data plane and handle one header field per pipeline stage is new. The decision tree would be amenable to drive per-packet decision (*i.e.*, forward or deny) while allowing dynamic rule addition and rule deletion at run time. In our future work, we will expose APIs for the same and study memory overhead reduction using decision tree-based approach.

3.5 Discussion

Device-to-device communication. Traffic between IoT devices under the same CPE are not monitored in our design. We concentrate on the IoT traffic that traverses the edge access switch (outside the LAN).

DSCP value count. Our system is limited by the availability of only 41 values to represent device types per CPE. This is sufficient in a smart home like setup where all IoT devices connected to one or two wifi access points (or CPEs). In setups with many IoT devices under a CPE, our system could benefit by using alternative reserved bit fields which could be used for packet making.

System scale. We estimate the scale of rules stored at an edge switch with SRAM memory of 100MB and 41 device types per CPE. We could estimate the number of rules as No. of CPEs * 41 + (No. of IoT device types * MUD rules per IoT device). Assuming there are 14.3K IoT device types [24] and 100 MUD-based ACL rules per IoT device where the size of each rule is approximately 12 bytes, we can support up to a total of 0.43 million CPEs. We also estimate that applying optimization techniques (like decision trees) would further increase the number of CPEs supported.

DHCP overhead. Our system design proposes mirroring of all DHCP discover packets from the CPE to an SDN controller at the edge. However, it can be noted that DHCP requests are generated whenever a new device joins the network. Assume a new IoT device joins a CPE for every one hour, which means 1 DHCP packet per hour per CPE. For a scale of a million homes (*i.e.*, one million CPEs), we can observe 1 million DHCP requests per hour, that is, the SDN controller receives about 278 DHCP packets per second which can be handled with enough compute at the edge.

DHCP security concerns. We acknowledge that exposing DHCP server details could lead to DHCP starvation and spoofing attacks [13]. However, in our system design, we do not forward subsequent local DHCP communication containing server address (*i.e.*, offer, ack, etc) to the SDN controller. Thus we reduce attack possibilities for an adversary intercepting the DHCP packets outside the CPE's local network. As a definitive defense, we could use gRPC based encrypted channels to forward DHCP packets [17].

Attacks on the system. The state maintained at the control plane and the rules installed in the data plane are initiated by MUD URLs carried by DHCP packets from the source CPE. An attacker could send spurious DHCP packets with MUD URLs of devices not present in the CPE with an aim to either exhaust DSCP bits or pollute dataplane tables, leading to resource exhaustion and DoS like attacks. To address this, we need a self-defense mechanism to authenticate IoT devices before they are enrolled into the network.

4 RELATED WORK

IoT MUD profiling. MUD maker [1] and Mudgee [21] can be used to generate MUD files. Such tools are very useful due to the lack of universal vendor support for MUD profiles. Our work uses the MUD profiles from these tools and focuses on a systematic and scalable framework for enforcing MUD profiles at the network edge.

MUD enforcement using openflow. [19, 20, 25] propose techniques to enforce and realize MUD profiles on OpenFlow-based virtual switch (OVS) in a local network. Since these systems are designed to work in local networks, they use MAC addresses in the MAT to identify IoT devices. Also, they use a single common MAT to accommodate all header fields. A similar approach in our system may not scale well as it needs TCAM memory in the dataplane when don't cares are present. In contrast, our system enables MUD enforcement on P4 switches at the network edge where lack of device identity and scaling to a large network are the two important problems addressed in this paper.

MUD enforcement at ISP level. [10] enforces MUD rules at the ISP level using off-path NFVs in the control plane. The ISP maintains whitelist MUD rules at the control plane and applies them on IoT traffic (as blacklist rules) from all its customer networks. However, this approach relies on a copy of a packet from every flow traversing a router in the ISP network, thus having high bandwidth overhead between data plane and control plane and processing overhead at the control plane. In comparison, our approach adopts their control (*i.e.*, TR-069) and packet marking schemes. However, we place whitelist rules in the P4 switch data plane and apply them to IoT traffic at line rate. By doing so, our approach does not send a packet for each flow, thus scales well.

Middleboxes to secure IoT. The works presented in [5–8, 14, 23, 29, 30] use middleboxes at the network edge for deep packet inspection and fine-grained analysis. Our work complements these efforts by filtering most of the traffic in the data plane, thus reducing processing delays and resource overheads.

5 CONCLUSION AND FUTURE WORK

We propose a system design for MUD enforcement in the edge network using SDN control plane and P4-based programmable data plane. We address challenges related to IoT device identification and scalability by leveraging packet marking at the CPEs and multi-stage pipeline and stateful ALUs at the switch. We believe our approach complements the existing middlebox-based security infrastructures at the network edge (*e.g.*, firewalls, IDS, IPS) for multiple classes of networks (*e.g.*, home, campus, enterprise) and reduce the overheads on such infrastructure by filtering most of the traffic in the data plane. In our future work, we plan to implement the proposed system on a testbed with IoT devices, CPEs, SDN controller, and a programmable switch. In addition to the implementation of the core functionality, we will study the system performance interms of memory overheads, false positive rate, reset frequency, feasibility of decision tree based rules, and control- and data-plane bandwidth.

6 ACKNOWLEDGEMENTS

We thank Dhiraj Saharia and Shiv Kumar for their valuable feedback on the final draft of the paper. We also thank Achmad Basuki, Yung-Wey Chong and Selvakumar Manickam for their insights. The work is supported by the SERB ASEAN (CRD/2020/000347) initiative.

REFERENCES

[1] 2018. MUD Maker tool. https://www.mudmaker.org/ Accessed: 2022-05-05.

[2] 2018. UNSW MUD Profiles of 28 IoT devices. https://iotanalytics.unsw.edu.au/mudprofiles

[3] 2019. Differentiated Services Field Codepoints (DSCP). https://www.iana.org/assignments/dscp-registry/dscp-registry.xhtml Accessed: 2022-05-05.

[4] 2019. RFC 8520. https://rfc-editor.org/rfc/rfc8520.txt Accessed: 2022-05-12.

[5] 2022. Barracuda. https://www.barracuda.com/products/cloudgenfirewall/use_cases/internet-of-things-security Accessed: 2022-05-05.

[6] 2022. Bitdefender. https://www.bitdefender.com/smart-home/ Accessed: 2022-05-05.

[7] 2022. Cujo. https://cujo.com/ Accessed: 2022-05-05.

[8] 2022. Premier Network Access Control (NAC) Solutions Security. https://www.fortinet.com/products/network-access-control Accessed: 2022-05-05.

[9] 2022. What is CPE and Why Does It Matter? https://www.promptlink.com/media-library/blog/what-is-cpe-and-why-does-it-matter.html Accessed: 2022-05-10.

[10] Yehuda Afek, Anat Bremler-Barr, David Hay, Ran Goldschmidt, Lior Shafir, Gafnit Avraham, and Avraham Shalev. 2020. NFV-based IoT security for home networks using MUD. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9.

[11] Yehuda Afek, Anat Bremler-Barr, David Hay, Lior Shafir, and Ihab Zhaika. 2020. NFV-based IoT Security at the ISP Level. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–2.

[12] Anurag Agrawal and Changhoon Kim. 2020. Intel tofino2 – A 12.9 tbps p4-programmable ethernet switch. In *IEEE HCS*.

[13] Manar Aldaoud, Dawood Al-Abri, Ahmed Al Maashri, and Firdous Kausar. 2021. DHCP attacking tools: an analysis. *Journal of Computer Virology and Hacking Techniques* 17, 2 (2021), 119–129.

[14] David Barrera, Ian Molloy, and Heqing Huang. 2018. Standardizing IoT network security policy enforcement. In *Workshop on decentralized IoT security and standards (DISS)*, Vol. 2018. 6.

[15] Andrei Broder and Michael Mitzenmacher. 2004. Network applications of bloom filters: A survey. *Internet mathematics* 1, 4 (2004), 485–509.

[16] Jie Ding, Mahyar Nemati, Chathurika Ranaweera, and Jinho Choi. 2020. IoT connectivity technologies and applications: A survey. *arXiv preprint arXiv:2002.12646* (2020).

[17] Sang Gyun Du, Jong Won Lee, and Keecheon Kim. 2018. Proposal of GRPC as a new northbound API for application layer communication efficiency in SDN. In *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*. 1–6.

[18] Harm Griffioen and Christian Doerr. 2020. Examining mirai's battle over the internet of things. In *ACM CCS*.

[19] Ayyoob Hamza, Hassan Habibi Gharakheili, Theophilus A Benson, and Vijay Sivaraman. 2019. Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity. In *ACM SOSR*.

[20] Ayyoob Hamza, Hassan Habibi Gharakheili, and Vijay Sivaraman. 2018. Combining MUD policies with SDN for IoT intrusion detection. In *Workshop on IoT Security and Privacy*.

[21] Ayyoob Hamza, Dinesha Ranathunga, Hassan Habibi Gharakheili, Matthew Roughan, and Vijay Sivaraman. 2018. Clear as MUD: Generating, validating and applying IoT behavioral profiles. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*. 8–14.

[22] BAG Hillen, I Passchier, EF Matthijssen, FTH den Hartog, and F Selgert. 2008. Remote management of mobile devices with broadband forum's TR-069. In *Networks 2008-The 13th International Telecommunications Network Strategy and Planning Symposium*. IEEE, 1–7.

[23] Ronny Ko and James Mickens. 2018. Deadbolt: Securing iot deployments. In *Proceedings of the Applied Networking Research Workshop*. 50–57.

[24] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. 2019. All Things Considered: An Analysis of {IoT} Devices on Home Networks. In *28th USENIX Security Symposium (USENIX Security 19)*. 1169–1185.

[25] Mudumbai Ranganathan et al. 2019. Soft MUD: Implementing manufacturer usage descriptions on OpenFlow SDN switches. (2019).

[26] Sergei Shevchenko. 2018. "VPNFilter" botnet: a SophosLabs analysis. *A SophosLabs technical paper* (2018).

[27] George Varghese. 2005. *Network Algorithmics: an interdisciplinary approach to designing fast networked devices*. Morgan Kaufmann.

[28] Christos Xenofontos, Ioannis Zografopoulos, Charalambos Konstantinou, Alireza Jolfaei, Muhammad Khurram Khan, and Kim-Kwang Raymond Choo. 2021. Consumer, Commercial and Industrial IoT (In) Security: Attack Taxonomy and Case Studies. *IEEE IoT* (2021).

[29] Tianlong Yu, Seyed Kaveh Fayaz, Michael P Collins, Vyas Sekar, and Srinivasan Seshan. 2017. PSI: Precise Security Instrumentation for Enterprise Networks.. In *NDSS*.

[30] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. 2015. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. 1–7.