NFV-based IoT Security for Home Networks using MUD

Yehuda Afek*, Anat Bremler-Barr‡, David Hay†, Ran Goldschmidt‡, Lior Shafir*, Gafnit Avraham‡, Avraham Shalev‡

*The School of Computer Science, Tel Aviv University, Tel Aviv, Israel.
†School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel.
‡Computer Science Department, Interdisciplinary Center, Herzliya, Israel.

***Abstract*—We present a new system to protect IoT devices in multiple premises by a single Virtual Network Function (VNF) deployed in the ISP network. The system is based on the Manufacturer Usage Description (MUD) framework, a white-list IoT protection scheme that has been proposed in recent years.**

While MUD is designed for on-premise deployment, here we adapt it to work as a scalable, managed service in the ISP level. Our service does not require any cooperation or installation on the client premise or on the IoT devices themselves. Furthermore, it monitors the IoT traffic and detects malicious behavior, including outgoing DDoS traffic, without being on the critical path, and it filters bad traffic by ACLs on either the POP router or the client CPE. The CPE itself is considered an IoT device and traffic destined or that originates at the CPE is monitored as well. For the white-list method we extend the MUD architectural framework to support peer to peer communicating IoT devices (e.g., direct mobile device to IoT device communication).

The system includes a mechanism to distinguish between flows of different devices at the ISP level despite the fact that most home networks (and their IoT devices) are behind a NAT and all the flows from the same home come out with the same source IP address. Moreover, the NFV system needs to receive only the first packet of each flow/connection at the VNF, and rules space is proportional to the number of unique types of IoT devices rather than the total number of IoT devices (which is much larger).

A PoC with a large national level ISP proves that our technology works as expected, identifying the various IoT devices that are connected to the network and detecting any unauthorized communications.

I. Introduction

IoT devices are strong enough to host malicious code or become a zombie, but, for economic and technological reasons, they do not have the means to protect themselves from being hacked. Thus, the billions of IoT devices (estimates are from 10 to 50 billion by 2020, 50% being consumer devices) are fertile ground for many attacks of different kinds: DDoS attacks, privacy infringing attacks, data leakage, and physical break-ins. Several different systems and methods to provide protection to IoT devices have been proposed in recent years [1]–[6]. In this paper, we present an ISP level system that protects the IoT devices in a large number of homes. This effective, scalable system should be a key part of protecting and stopping attacks on and from IoT devices.

978-1-7281-4973-8/20/$31.00 © 2020 IEEE

The underlying property that distinguishes most IoT devices from home computers, such as laptops, desktops, and hand-held devices, is that in most cases an IoT device communicates with a limited number (most usually very small) of pre-defined domains. Furthermore, these devices, rarely if ever, go through a firmware update. These characteristics have been exploited to create a whitelist based protection methods for IoT devices. A recent initiative that receives a lot of attention, both in industry and academia, calls for IoT device vendors to provide a *Manufacturer Usage Description (MUD)* for their products [7]. These descriptions, called MUD files, consist of whitelists describing the devices' legitimate communication, specified by the domain names of legitimate endpoints. Methods to automatically learn and acquire the MUD file for a device type have also been developed [8].

To protect an IoT device, given its MUD file, or any other whitelist, it is necessary to resolve the domain names in the list to get the corresponding IP addresses (which might vary in different geographies and at different times), then to monitor the device traffic ensuring it communicates only with these IP addresses and complies with the MUD file (e.g., uses the specific ports and protocols specified in the file). If a deviation is detected, offending connections should be blocked and alerts on suspicious activity are issued. Suggested implementations of the MUD standard as well as other IoT protection systems, implement these steps within the internal/local area network (LAN), i.e., for home networks on the CPE, [1]–[3], [9]–[15], though sometimes on a separate device. In this paper, we provide a system architecture to implement these steps at the ISP level. The system combines an off-path Virtual Network Function (VNF) with the existing ISP's on-path enforcement capabilities. The VNF holds the MUD rules (where domain names are already resolved to IP addresses) and monitors a large number of home networks by these rules. Upon a violation, the corresponding connection is blocked in one of the ISP's on-path routers or switches, e.g., using ACLs.

As home networks apply Network Address Translation (NAT), implying all devices use the same IP address and port numbers are arbitrary, perhaps the major challenge when working outside the LAN is to distinguish between connections originating from different devices. Our design overcomes this

difficulty by dynamically installing *packet marking rules* on the home gateway router (often called a Customer Premise Equipment or CPE), using its standard configuration protocol TR-69 [16]. Marking is done on the DSCP header field (thus not introducing extra overhead) only on packets originating from IoT devices. Furthermore, marking rules are installed only when the device first connects to the CPE.

The IoT protection architecture suggested here has several advantages: First, it is not on the critical path of the traffic as the VNF takes only a copy of the first packet of each connection. Second, for any home whose CPE is managed by the ISP (the vast majority of homes) no cooperation from the client is necessary, the protection may be provided as a service in one click on the ISP management unit. Third, our system provides the customer and the ISP a complete view of the devices in the network. This is achieved by retrieving information from the CPE using TR-069 and identification of the IoT devices using existing MUD files (or other whitelists) installed for devices across the entire network. Moreover, the customer can receive an alert for any device that connects to its network. These views of customer devices are also an important and helpful service for customer help service at the ISP. Fourth, the number of rules held in the VNF is proportional to the number of IoT device types (unique MUD files), and not to the total number of IoT devices it is monitoring; moreover, domain resolution is done in the VNF once for all devices of the same type, thus reducing the load on the DNS server and enable using more secure DNS servers. The centralized, upstream location of the solution, observing many home networks at once, holds a great promise as the system can correlate traffic surges and patterns from multiple homes to detect and mitigate outgoing DDoS attacks and other global attacks. Furthermore, the system is situated at a good point to stop additional attack vectors, such incoming DDoS attacks on IoT devices and homes or attacks on (or that come out of) the CPE itself. Our system also monitors all connections that terminate in the CPE, as the CPE itself is considered an IoT device.

To cope with internal traffic, that is not visible outside the customer premise, we suggest a hybrid approach, where we deploy a lightweight component at the CPE, whose sole purpose is to monitor such traffic. We extend the MUD specification and its whitelisting capabilities to support P2P communication in which IoT devices act as servers and their users (e.g., another device within the local network) act as clients. P2P communication is most common in intra-LAN communication. Yet, our system also covers users' remote access (e.g., using a mobile phone), which can be granted using port forwarding on the CPE, UPnP [17], or STUN [18] with Hole-Punching [19]. We note that extending the MUD protocol to support such communication, and then to incorporate its monitoring and enforcement in the ISP level, is of great importance since recently-reported botnets, like MIRAI [20], targeted devices based on P2P communication. Notice that as reported in Shodan [21], there are still many IoT devices that are vulnerable. Besides covering more attack vectors, such a hybrid system is beneficial to provide a certain level of security even if one of its components fails.

The rest of the paper is organized as follows. In Section II we will discuss related work on MUD and IoT whitelisting. Section III describes our ISP-level system, which includes a virtual network function, whitelist enforcement component, and communication with the CPE to ensure correct packet marking. Section IV describes how we extend the MUD specification to support P2P communication (either internal traffic within the home network or P2P communication across the Internet). Section V provides details on our implementation and our PoC deployment with a large national-level ISP. Concluding remarks are given in Section VI.

II. Related Work

MUD (Manufacturer Usage Description) [7], has been officially approved to define IoT devices communication patterns in order to reduce the attack surface on IoT devices. In this standard, IoT vendors/providers are encouraged to provide a MUD file consisting of access control rules that describe the device's proper communication behavior [22]. Recently, Cisco announced that some of devices support MUD and some vendors already defined MUD files [23]. The MUD framework assume that the components at the network are MUD controller, that process the MUD information, and the CPE that run ACL. Thus, in the home-network environment, it assumes that the CPE deploys MUD. In contrast, our system is deployed within the ISP network and receives a copy of the traffic from a PoP router. As far as we are aware, this is the only MUD solution that is deployed at the ISP level. Moreover, all IoT security frameworks, including the ones that do not use MUD or whitelisting at all, that we are aware of (either from Academia or Industry) focus on solutions within the LAN and not in the ISP level [1]–[6], [11]–[13], [24].

MUD-based solutions are still incomplete, and there are several recent works that focus on extending or implementing MUD, some of them in SDN environment [25]–[27]. Important issues that are yet to be addressed are dealing with IoT devices that may be accessed directly by the owner mobile device(s) (in a P2P manner), dealing with services behind a cloud bucket (e.g., Amazon S3 bucket), where the exact bucket is specified in an encrypted manner, and dealing with devices in which the user can add apps/skills (such as a smart TV or Amazon Echo). In this paper we extend MUD to handle P2P traffic.

III. NFV-based System Description

The system goal is to ensure that all packets from and to an IoT device comply with the MUD file rules. This implies that, for each packet , the system needs to decide whether it conforms with a MUD file (or another form of a whitelist), and if not, to block the packet. The MUD enforcement has thus two logical components, monitoring and enforcing. In the *whitelist monitoring* (WLM), it is determined whether a packet/connection complies with a whitelist/MUD file or not; In the *whitelist/MUD enforcement* (WLE), based on the output of the WLM component, the packet is either dropped or

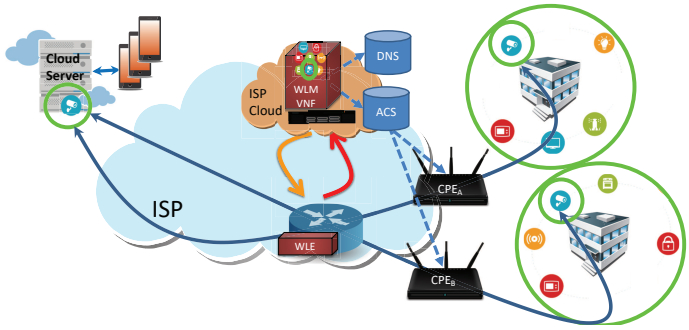

Fig. 1. Illustration of our Framework. The WLM's data plane is marked with solid lines, while the WLM's control plane is marked with dash lines. In this illustration, the WLM VNF protects two LANs that communicate with the Internet through two CPEs. Both LANs have a camera installed in them, which connects to the same service in the cloud. The WLM VNF contains whitelists/MUD files for all IoT devices in the system. It verifies that both cameras communicate only with endpoints within their whitelist/MUD file.

permitted. While WLE must be *on the traffic path* observing every packet originated or destined for IoT devices; the WLM may be performed on a copy of the traffic, thus being off the critical path. In our implementation WLM, is implemented as a VNF that receives only a single packet of each connection, sent from an IoT device, to decide whether the entire connection should be permitted or blocked.

The control-plane in our implementation includes the control-plane inside the VNF as well as control communication with other control-planes, such as on the CPEs and other components, see Figure 1. We next describe its operation when the IoT devices act as clients initiating connections to their cloud services (or other services). We assume that customers connect to the ISP network through a CPE with a unique IP address (which can change over time). As in the typical setting, the IoT devices reside behind a NAT (performed by the CPE), implying that outside the customer network all outgoing packets appear to come from the same IP address with arbitrary ports. We further assume that IoT devices have a MUD file (or another readily-available whitelist). Notice that our framework also provides the infrastructure to *learn* a new MUD file of devices whose manufacturer did not provide one. The method itself of learning the MUD files from deployed IoT devices in the network is out of the scope of this paper. We note that there are tools like MUDgee [8] that create a MUD file from traffic traces of labeled devices in the lab. However, it is not straightforward to apply these tools on unlabeled IoT devices, that are deployed in the wild, and one has no control over their operations (specifically, one cannot actively try specific device functionality and observe the resulting traffic). Nevertheless, we note that the centralized view of our system is better in learning the regular behaviour of IoT devices (i.e., generating a MUD file), since it can observe and analyze multiple devices of the same type.

A. Data Plane

The data-plane has three major parts: the customers' private networks, the ISP network, and the VNF, which monitors traffic and issues alerts (on suspicious traffic). We assume that a MUD file, containing a whitelist for each IoT device *type* M, is given as a set of domains WLD_M, that correspond to IP addresses WL_M with which the device is allowed to communicate; namely, $WL_M = \bigcup_{D \in WLD_M} \{L \in \text{DNS-response}(D)\}$. Moreover, WL_M may contain more parameters (such as port numbers) as defined by the MUD protocol specification. We note, however, that our framework does not support Layer-2 parameters (such as MAC address) as these are not visible outside the LAN.

1) Operations within the Customers' Network: IoT devices that are monitored by the framework are oblivious to its existence, and no special configuration needs to be done at the IoT device level. However, to be able to distinguish packets coming from different IoT devices at the ISP level, CPEs that are connected to our framework should *mark* packets originating from different IoT devices with different values in the DSCP field in the IP packet header. DSCP marking (by source MAC address) is a standard feature in most CPEs available today and requires only CPE *configuration*. We note that packets originating from non-IoT devices (such as desktops, laptops, or mobile phones) are not marked with a special DSCP value (and maintain the original value used by the device; 0 in most cases). A newly-connected device, for which there is no indication whether it is an IoT device or not, is marked and tracked until such a decision is made. The remote CPE configuration (namely, the assignment of a DSCP value for each device) is part of the control plane's operations as described in Section III-B.

2) Operations within the ISP network: After transmitted from the customer premise, packets are sent through a sequence of routers within the ISP network, towards their destination. Our framework requires that one router along this route will *copy* outgoing packets from all monitored customer premises and send them to the VNF. This can be done either using port mirroring or installing a specific filter on the ISP's router. Moreover, the VNF needs only *one packet from each outgoing connection*. We define a connection in this context as all packets that share the same $\langle src_ip, src_port, dst_ip, dst_port, protocol \rangle$ tuple. In case the router supports the OpenFlow protocols, for example, this can be done by dynamically installing a filter on every observed connection. We note that packets are *copied* (and not routed) from the router to the VNF, thus our VNF does not reside on the critical path of the packets and does not induce extra delay or other performance penalties on the traffic.

The VNF may issue either alert on abnormal behavior or access-control list (ACL) filters to block malicious traffic in the WLE. The ACLs can be either specific (e.g., block a specific connection; in most cases, both directions of the connection are blocked simultaneously) or aggregated (block all traffic to/from specific address/network; this is especially appealing

in case of DDoS attack where many malicious connections are created simultaneously).

It is important to notice that our WLM VNF observes only traffic originating from an IoT device (and therefore, is marked by the CPE). Since most known attack traffic is bidirectional (e.g., for TCP traffic connections, at least acknowledgments are sent) , our WLM VNF can detect it. As mentioned before, our WLE component blocks traffic from both directions.

3) Operations within the VNF: Packets received by the VNF are treated by their DSCP value, where classification actions are mapped to the multiple match-action table (mMT) model (e.g., as in OVS [28]), see Figure 2; table rules are updated by the control plane (see Section III-B). Specifically, we distinguish between the following cases:

Unmarked packets: Packets that have a DSCP value of 0 (or another commonly-used value) were originated from a device that is not an IoT device, and therefore, is not monitored by our framework, implying our VNF should discard these packets.

Marked packets of an identified IoT device: When a device joins a customer network, the CPE is configured to mark all its packet with a certain value. Through our control plane, we try to obtain the MUD file corresponding to this device and associate it with its specific type X. This implies that one should only verify that the packets comply with the corresponding MUD file/whitelist. This is done by matching the header of the packet to WL_X (e.g., verifying that the destination IP address is in the whitelist). In case any deviation is detected, an alert is issued and/or the connection is blocked in the WLE.

Marked packets of an unidentified device: In case no MUD file exists, we collect all of the device DNS queries and all the endpoints the device is communicating with. This information is sent to the control plane that analyzes this information and decides which further actions should be taken (See Section III-B2). Due to technical reasons, there might be a gap between the time a device joins a customer's network and the time the corresponding CPE starts marking its packets. In order not to treat these first packets as unmarked, we configure the CPE to have a default proprietary DSCP value for all packets originating from the CPE. All such packets are treated as marked packets of an unidentified device, and the VNF associates them with the recently joined device (if only one such device exists). See Section III-B3 for more details.

It is important to notice that we store only one whitelist (or MUD file) per IoT device type (namely, all IP cameras of a specific model are monitored with a single whitelist in memory, regardless of their number). Specifically, if there are 21 commonly-used DSCP values [29], C customers in the framework, each with d_i IoT (or unidentified) devices, and P whitelist/MUD-files, each with p_j entries (e.g., legitimate destination IP address, port number, etc.), then the total number of filters required to implement the entire data-plane is $23 + \sum_{i=1}^{C}(d_i+1) + \sum_{j=1}^{P}(p_j+1)$. Furthermore, all update events (such as a change in the external IP address of a customer, new device has joined a customer's network, a profile has a new legitimate IP address, or a device is identified as a non-IoT device) requires a single filter update (in a single table within the pipeline).

Table	Match	Action
0	DSCP value is commonly-used	drop
	otherwise	goto Table 1
1	source IP address is of a customer	apply unique metadata, goto Table 2
2	Meta-data, DSCP value	unidentified device: send to controller
		identified device (type X): goto Table X
$X \geq 3$	destination IP in WL_X	drop
	(dest port, protocol) in WL_X	drop
	otherwise	Output 1

Fig. 2. Our pipelined match-action implementation. An application that deals with traffic that violates the whitelist (e..g, generates ACL and sends it to the WLE) is connected to Output 1. In case the number of types exceeds the number of available tables, several types can reside on the same table, by applying a meta-data value to distinguish between them in Table 2. Counters of filters in Table 2 present per-device activity of each customer.

B. Control Plane

The control plane operations are divided into the following tasks that are executed periodically, when a new customer joins the framework, or when a new device joins an existing customer's network (namely, a new device is connected to the network for the first time).

1) Maintaining data plane's whitelists: The first task of the control plane is to retrieve, for each IoT type M, the whitelist WL_M from the given domain list WLD_M that is specified within its MUD file.

One possibility is by looking at the DNS responses to the DNS queries it already tracks. However, this implies that both traffic directions should be monitored (sometimes this is not feasible) and the whitelist is prone to contamination, e.g., by DNS poisoning attacks, as the DNS response might be bogus.

We chose an alternative way of performing *active DNS queries* of the domains in all the WLD lists. This is done periodically (every few minutes) to keep with changes in IP addresses. Moreover, upon the detection of communication of IoT device of type M to an unidentified endpoint, DNS queries of all WLD_M are immediately triggered. This is done to verify that IP addresses did not change since the last active DNS query and thus eliminating false positives alerts. We take extra care to avoid learning malicious IP addresses due to DNS poisoning, by performing the DNS queries to a secure DNS server (such as OpenDNS or Quad9) that is immune to such attacks. It is important to notice that since our WLM is monitoring a large number of home networks, active DNS queries are done once for every IoT type (and not for every IoT device), thus reducing the load on the DNS server.

2) Analyzing traffic of unidentified devices: Recall that for devices without MUD files, the control plane receives the list of domains and endpoints the device communicates with. The control plane compares (in an efficient manner) this list of to all known whitelists/MUD files. If the device matches

a whitelist of some IoT device type X, we associate these marked packets with X by rewriting the corresponding rule at Table 2 to point to Table X (see Figure 2) and by that ensuring that the device continues to comply with WL_X.

Otherwise, the device is either an IoT device type that has not been seen before (and therefore a new whitelist needs to be created for it) or it is not an IoT device. A copy of this device traffic is sent to a control application, that decides if the device is an IoT or not an IoT (using techniques such as in [30]). If the device is a new type of IoT device with no associated MUD file, we send the traffic (or part of the traffic) to a control application that automatically learns the white-list corresponding to this new device type. As mentioned before, the system described in this paper provides only the infrastructure for MUD file generation and the exact algorithmic way to do so (e.g., by variation of [8]) which is out of the scope of the current paper. Finally, after a MUD file is provided, we install the corresponding whitelist on a separate Table X', and associate the device with X'. If the device is not an IoT device, the control plane instructs the CPE to remove the DSCP mark from the device's packets, and therefore, all subsequent packets from this device will arrive at the VNF unmarked (see Section III-B3).

3) Remotely Configuring CPEs: One of the most important parts of the control plane is the communication between the VNF and the CPE, to configure the latter. Such a configuration can be done in various ways, depending on the CPE capabilities. However, in the vast majority of cases, the CPE supports the TR-069 protocol for its remote configuration and ISPs use this protocol to configure the CPE connected to their network. Thus, our description will focus on this protocol. Similar operations (and configurations) can be done using SNMP and proprietary management protocols. We note that once configuring a CPE, this configuration remains persistent (unless it is factory reset).

TR-069 [16] is a client-server protocol, where the CPE acts as a client and periodically initiates connections to an *Auto-Configuration Server* (ACS). When a connection is established, the ACS sends configuration requests to the CPE, which are executed one by one. The ACS also exposes a north-bound interface, so that management tools and applications can communicate with it. Thus, our VNF, like any other TR-69–based application, communicates with the ACS (in our case, the ISP's ACS), which in turn, passes the requests to the CPE. An important feature of TR-069 is the ability to set a specific parameter for "active notification". In such a case, if the parameter has been modified by an external cause (a cause other than the ACS itself), the CPE initiates a connection immediately to the ACS and notifies it on the parameter change. We will use this feature later to detect new devices that join the customer's network. The TR-069 supports data models that represent the entities that need to be configured. CPEs are modeled using the Internet Gateway Device (IGD) Data Model (TR-98) [31] and Device Data Model (TR-181) [32], and our VNF uses only mandatory parameters within these models.

Our VNF initiates configuration requests upon the following events:

New customer is added to the network: The external IP address of the customer is extracted (so that the VNF can identify connections from that customer). In addition, the respective TR-98 parameter is set to "active notification", so that future changes to the customer's external IP address will be reported automatically. Moreover, the VNF extracts basic details on all hosts that are connected to the CPE, and for each host, it adds a mapping between its MAC address and a DSCP mark.

Also, it adds a default DSCP mark (see Section III-A3) and sets active notifications for the number of connected hosts (to detect hosts that join the customer's network).

Finally, in many settings, devices that are connected to the network are configured (through DHCP) to use the CPE as their DNS resolver. This implies that each DNS request is terminating in the CPE, which, in turn, issues another DNS request to ISP's DNS resolver, if needed. This implies that DNS requests arrive at the CPE unmarked. Since our data-plane tracks DNS queries, and since it is important to associate each query with the device that issued it, we change this configuration, so that each device communicates directly with the ISP's DNS resolver.

An external IP address is changed: Recall that this parameter is marked for active notification so that as soon as this happens, the CPE notifies the ACS (and then our VNF) on this change. The VNF then updates its internal state to monitor connections/packets from the new external IP address rather than the old one (namely, in Table 1 of the pipeline described in Figure 2).

New device joins the network: As the number of hosts connected to the CPE is marked for active notification, the VNF is notified immediately upon such an event. Then, basic details of the device (such as its MAC address, hostname, the medium of connection, etc.) are extracted from the CPE, and the CPE is configured to mark all packets transmitted from this device. The VNF is configured to monitor these packets as well as to associate all packets with default mark to this device (to compensate on the time until the DSCP mark comes into effect).

As MUD files are specified by IoT devices through DHCP, we check the specific DHCP option (available in the Device data model) for the URL of the MUD file. If such exists, we check if the MUD file is already installed in the VNF and start monitoring the device by associating it to the correct table (see Table 2 in Figure 2). If the MUD file is not installed, we download, parse, and install it as a new table in the VNF. See Illustration of this process in Figure 3.

When no MUD file is specified, all packets transmitted from the new device are marked packets of an unidentified device. We note that we consider a device as new only on the first time it connects to the CPE. If the device disconnects from the network and re-connects again, the CPE still lists it under its hosts (as inactive hosts), implying DSCP marks are already available, and no further action is required.

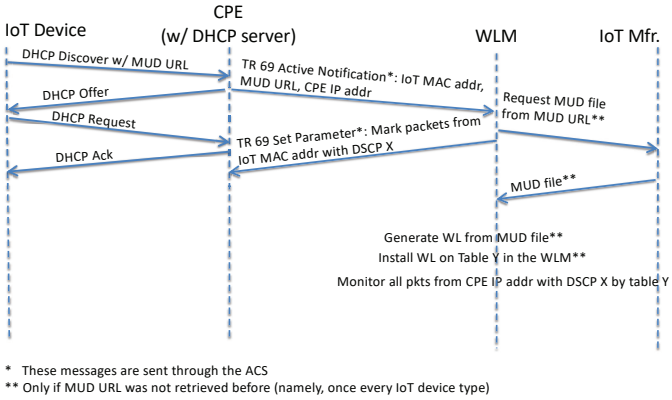

Fig. 3. Illustration of the messages sent after a new IoT device with preconfigured MUD-URL joins a network. Note that messages between the CPE and the WLM are sent through an ACS. Furthermore, MUD-URL is retrieved by the WLM only if it was not retrieved before (namely, only once for every IoT device type).

A device is classified as a non-IoT device by the VNF: In this case, the DSCP mark on packets transmitted for this specific device should be removed, so that the VNF will stop monitoring the device by discarding the unmarked packets. This is done by setting the `DSCPMark` within the respective classification object to value -1, indicating that the CPE should not change the DSCP value of packets.

IV. Extending MUD for P2P communication

One important limitation of MUD files is that it does not cover cases where the IoT device is a server and its users act as clients, whose IP addresses can change frequently. Such P2P communication is common when the IoT device and its users are in the same LAN (namely, the same customer network) or when a low latency connection between the device and its users is required (e.g., for improving QoS of a video streaming): In our study, of IoT traffic logs [33], [34] , [35] and IoT devices in our lab, out of 42 IoT devices, five IoT devices[1] and one IoT alarm system act as servers. We note that IoT P2P communication is less frequent nowadays, as most IoT devices communicate with their users through a proxy in the cloud (and are covered by our NFV-based solution as described in Section III). Yet, recent devastating IoT attacks (such as Mirai [20]) targeted exactly such devices.

In most cases, users access IoT devices through a dedicated application, and therefore, we will use the term *IoT app* to refer to these users and distinguish them from users of our security system. Furthermore, the device on which the IoT app is installed is called *IoT app device (IAD)*.

The main problem of MUD for P2P communication is that the IoT app that is allowed to access a certain IoT device, is typically not allowed to access all IoT devices of the same type (e.g., user of a camera of type X should not access other cameras of type X that it does not own). Thus, without additional mechanisms, the device manufacturer cannot specify in the MUD file any details on the IoT app. One common approach to cope with that, similar to what was shown in [26], is to allow incoming traffic from any IP in the internet (i.e., "*"); naturally, such a whitelist has close-to-zero precision.

We have taken a different approach and extend the MUD specification to allow a *secondary virtual manufacturer (SVM)*, whose sole purpose is to deal with P2P communication. Notice that this virtual manufacturer may deal with all P2P communication in the network (regardless of the manufacturer of IoT device). Adding such a secondary virtual manufacturer requires a minimal change in the specification and minimal cooperation from the primary (physical) manufacturer: In the MUD file, which the primary manufacturer provides to the WLM (often called MUD manager in this context), it should specify that its device supports direct connection. This can be done by inserting a unique placeholder (in our implementation, `$owner-unique-domain$`) to the file.

Upon encountering the unique placeholder, the WLM permanently replaces it with the specific domain (or domains) provided by the SVM, as will be described next. Upon such replacement, the WLM is left with a regular MUD file and can continue its normal operations without any change.

A. Deploying WLM and WLE for P2P Communication

As P2P communications include local traffic and, in any case, the MUD file of every device (and not every device type) is different, the WLM and WLE should be deployed within the Customer Premise Equipment (CPE); namely, the home gateway router. This can be done by installing a software on the CPE device, which can run either as a process or as a container in this machine. CPE-based solution implements WLM and WLE on the same box, but can be deployed as different processes, where WLE can sometimes use available hardware to enhance its performance.

CPE-based deployments have the best visibility as it can observe and block unauthorized internal traffic between devices in the same network. Moreover, as typically there is no NAT between the CPE and the IoT devices, it is easy to associate packets to their corresponding IoT device (no need for DSCP marking, etc.). On the other hand, WLM must be able to resolve all domain names in all its MUD files to maintain the correct mapping to IP addresses. This can be done either by extracting the addresses from DNS responses (which, in turn, requires deep packet inspection of packets) or periodically performing all the required DNS queries.

Recall that we envision a hybrid deployment, which has CPE-based component for P2P and internal communication of IoT devices and NFV-based component for all other traffic, thus achieving the best of both worlds (See Fig. 4). For IoT devices that have both types of communication, the whitelist is spread across the two components, which might cause conflicts in a naive implementation: connections that are approved by the whitelist in the CPE are later examined by the NFV-based

[1]The devices are Belkin Wemo switch, Belkin Wemo Motion Sensor, Motorola Hubble, TP-Link Day Night Cloud camera, and Samsung Smart Cam. All these cameras use STUN [18] to allow incoming connections to the home networks, as the cameras are typically behind NAT.

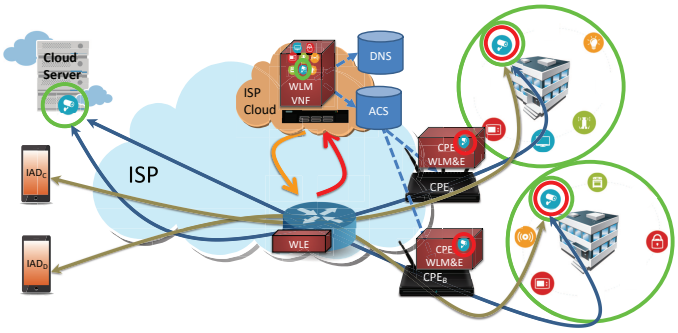

Fig. 4. Illustration of our framework, where WLM&E component in the CPE is responsible on P2P and local communication, while the VNF-based WLM is responsible for all other communication. In this illustration, the same camera is installed in two homes, where IAD$_C$ (IAD$_D$) connects directly to the camera at CPE$_A$ (CPE$_B$). A whitelist with the IP address of IAD$_C$ (IAD$_D$) is installed in CPE$_A$ (CPE$_B$). A single whitelist in the VNF-based WLM monitors all other communications for both cameras.

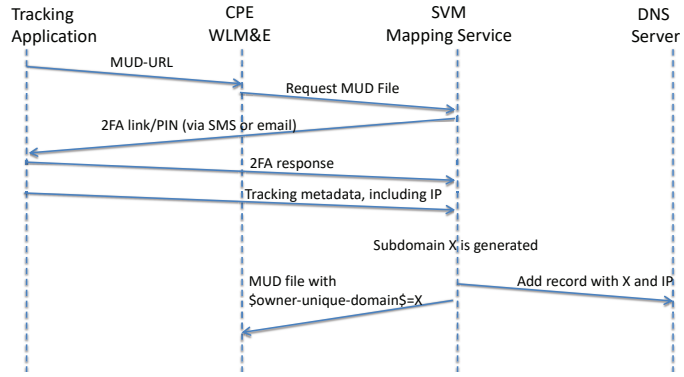

Fig. 5. Illustration of the sign-up process of a IAD with the SVM and the corresponding CPE. The sign-up process must be completed when the IAD is connected through the CPE.

WLM and are blocked. To avoid such conflict, we make sure that the CPE-based WLE component reset the DSCP value of all packets matching its whitelist (implying they will not be inspected or blocked by the NFV-based components).[2]

B. The Secondary Virtual Manufacturer (SVM)

The SVM is responsible for generating specific domain names for IADs and keeping a correct mapping between these domain names and their corresponding IP addresses. The main challenge is that, typically, IADs' IP addresses are changing very frequently (e.g., when the IAD is a mobile device and it moves between networks) implying the SVM should *track* the relevant IADs and update the corresponding mapping. In a nutshell, this is done by installing an *SVM tracking application* on the IAD, which consistently reports the IP address of the IAD (and other metadata). This data is sent to the *SVM's mapping service* (deployed in the cloud) which updates the DNS record that maps between the IAD's IP address to a unique domain. We note that, as we work with DNS, changes in IP addresses do not cause any change in the normal operations of the WLM, which include domain name resolutions and coping with DNS changes.

Specifically, in our implementation, the SVM owns a parent domain and given each IAD a unique randomly-generated sub-domain under this domain. The SVM maintains the authoritative DNS server for the parent domain (and all its sub-domains) to ensure that changes to DNS records, generated by the *SVM's mapping service*, are reflected in subsequent DNS queries. Yet, one major issue that can be caused due to this system architecture outdated resolution due to stale cached records in one of the DNS revolvers. This problem can be circumvented in standard techniques that include adding random value as a sub-domain so that each DNS query will bypass all caching mechanisms or pointing the WLM to the SVM's authoritative DNS server. The authoritative DNS server will then remove the random value and return update answer.

The most involved operation of the SVM is when a new user sign-up with the service, which is Illustrated in Fig. 5. A user who wants to use this protection needs to install the SVM tracking application and create an account with the SVM first. To enable two-factor authentication, users are required to provide either an email address and/or phone number. During the sign-up process a new unique sub-domain and identifier to the account are generated. To correlate the newly-correlated account with a specific LAN, we require that the SVM tracking application will be connected, in the first time, during that LAN. The operations continue in the following steps, where the SVM tracking application acts as a (virtual) IoT device, while the SVM mapping service acts as its manufacturer: (i) The SVM tracking application broadcasts a MUD URL, which includes the SVM parent domain and the identifier of the newly created account; (ii) The WLM fetches this MUD URL by contacting its manufacturer as specified in the URL. This request goes to the SVM's mapping service; (iii) The SVM's mapping service extracts the account identifier and initiates a two-factor authentication with the user, using the email address and/or phone number provided when the account was created; (iv) Upon successful completion of the two-factor authentication, the mapping service replies to WLM with the MUD file that includes the account unique domain (namely, the value of `$owner-unique-domain$`); (v) The WLM extracts this unique domain from the MUD file, replaces all pending MUD files with `$owner-unique-domain$`, and save the domain name for later usage.

After an account is created, the SVM tracking application reports to the SVM mapping service whenever the external IP address of the IAD is changed. The message sent includes the account identifier, the unique domain, the external and internal IP addresses. If there is a change, the mapping service simply updates the corresponding DNS record in the authoritative DNS server (See Fig. 6).

[2]This is done in two steps: the WLE at the CPE marks the packets by a designated DSCP marking, and the CPE, as instructed by the NFV-based WLM through TR-69, reset the DSCP marks of such packets regardless of their source MAC address.

C. Dealing with Internal traffic

When the IoT app is in the same LAN as the IoT device, mapping to IP addresses is not required, and instead one can use other information available within the LAN to define whitelist rules in the MUD file (e.g., the IAD's MAC address). This can be generated in exactly the same process as described above, with an additional place-holder. However, in some cases, mostly in the presence of extenders within the LAN that rewrite MAC addresses, the IAD's internal IP address is needed; in this case, we configure the tracking application to send both internal and external addresses and configure two DNS records for that IAD. Reducing the surface of the attack also inside the LAN is extremely important, since there are many attacks that abuse local devices to internally attack IoT devices [36], or to leak data from them [37].

V. Implementation and PoC Details

We have implemented our system, and deployed a proof-of-concept in a large national-level ISP network.Specifically, the data-plane was implemented using Open vSwitch (OVS) version 2.8.1 [28] with OpenFlow 1.3. The control plane was implemented as applications (in Python) over Ryu—a common open-source OpenFlow controller [38]. Our implementation leverages on both the caching capabilities of OVS, its supports of DPDK, and the pipelined architecture of OVS and OpenFlow, where packets traverse multiple tables, each contains several rules before they are classified.

DNS traffic and traffic from unidentified devices (namely, devices with no corresponding MUD file) is sent to the controller, on top of which there is an application that tries to decide whether an unidentified device is an IoT device or not, to match unidentified device activity with an existing MUD file, to derive new whitelists, and to continue monitor changes in whitelists. Identified devices' traffic that is destined for IP addresses not within the whitelist/MUD file is forwarded to another application, which re-verifies that the traffic is indeed legitimate and issues an alert to the customer and an ACL to ISP, which can block this specific connection. The latter application communicates with the applications on top of the controller (e.g., to force active DNS queries) using RESTful APIs. We have also used the built-in OVS counters to track the activity on each IoT device and present it to the customers and the ISP. We note that while we have implemented our solution with OVS, as the solution is OpenFlow based and does not use any proprietary fields of OVS, one can easily migrate it to use any (physical or virtual) switch that supports OpenFlow 1.3 and above. In our PoC deployment, we do not have access to the ISP's ACS servers. Thus, we have deployed our own ACS server, running open-source GenieACS 1.1 [39], and configured the CPEs to communicate with it. As GenieACS does not support active notification setting yet, we have shortened the `inform` interval and manually refresh the corresponding TR-98 fields, to capture the monitored events (mainly, new device joins the customer's network for the first time, or a new port is opened in the CPE). Our VNF communicates with GenieACS through GenieACS's north-bound interface. We have 4 VTech NB-403 CPEs, running for several months in different locations within the ISP network. The ISP routes all traffic of these CPEs through our VNF, which runs as a virtual machine on an VMware ESXi hypervisor of the ISP.

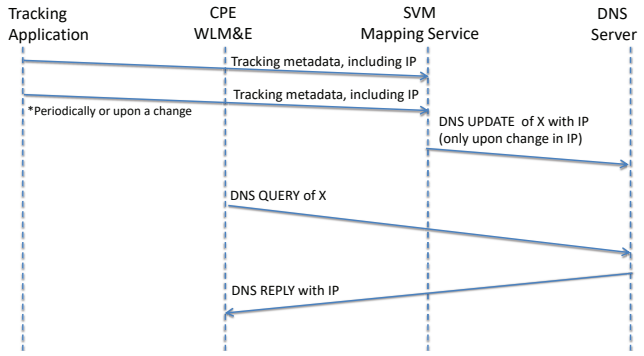

Fig. 6. Illustration of the messages sent after the initial sign-up process. The IoT device is oblivious to the system.

As for the extension to deal with P2P communication, we have implemented the tracking application and installed it with dozens of volunteers and checked some properties of the solution. We highlight here some observations from this experiment: First, DNS records were updates 15 times a day on average (with a maximum of 40 times). DNS record changes happen when IAD is switching between different networks (e.g., WiFi to cellular). On the other hand, the same IP address might belong to many records (namely, when IADs using the same WiFi networks or the same cellular provider, in case that provider use Carrier-Grade NAT [40]); this implies that users within the same network of the IAD might bypass our security mechanism (or any other MUD-based solution). Finally, in some cases DNS records must contain more than one IP address, since in some enterprise networks, due to load-balancing of outgoing connections.

VI. Discussion and Future Work

In this paper, we have presented a system to provide IoT security through a VNF deployed within the ISP. Our system is built upon the MUD specification, in which each device has a whitelist that describes its legitimate communication patterns, and any deviation from this whitelist is blocked. Our system may use a MUD file provided by the device (through DHCP, as specified in the MUD RFC). It also provides the infrastructure to derive MUD files from unidentified devices' traffic and to identify devices (that do not provide their MUD files by themselves) by matching their traffic to existing MUD files (or whitelists). Furthermore, as our system is deployed in a centralized location, monitoring a large number of home networks simultaneously, it is in an excellent position to detect global phenomena such as outgoing and incoming DDoS attacks. Our future work includes designing and deploying such mechanisms within our system.

Acknowledgments: This work was partly supported by the Federmann Cyber Security Resarch Center in conjunction with the Israel National Cyber Directorate.

References

[1] 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 *IEEE ICDCS*, 2019, pp. 756–767.

[2] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, and Y. Elovici, "N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[3] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, "IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT," in *IEEE ICDCS*, 2017.

[4] 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 *ACM/SIGAPP SAC*, 2017, pp. 506–509.

[5] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying iot traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 559–564.

[6] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer internet of things devices," in *IEEE DLS*, 2018.

[7] E. Lear, R. Droms, and D. Romascanu, "RFC 8520: Manufacturer Usage Description Specification," Internet Engineering Task Force, March 2019. [Online]. Available: https://datatracker.ietf.org/doc/rfc8520/

[8] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, and V. Sivaraman, "Clear As MUD: Generating, Validating and Applying IoT Behavioral Profiles," in *IoT S&P*, 2018.

[9] McAfee, " McAfee: Built-in Protection for Your Connected Devices," 2019, https://securehomeplatform.mcafee.com/.

[10] F. Roberts, "Trend micro partners with asus to beef up iot security in homes," *Internet of Business*, Jan 2017. [Online]. Available: https://internetofbusiness.com/trend-micro-asus-iot-security/

[11] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, pp. 76–79, 02 2017.

[12] J. Habibi, D. Midi, A. Mudgerikar, and E. Bertino, "Heimdall: Mitigating the internet of insecure things," *IEEE Internet of Things Journal*, vol. 4, no. 4, Aug 2017.

[13] 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.

[14] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: A Novel Honeypot for Revealing Current IoT Threats," *Journal of Information Processing*, vol. 24, no. 3, pp. 522–533, 2016.

[15] M. Özçelik, N. Chalabianloo, and G. Gür, "Software-Defined Edge Defense Against IoT-Based DDoS," in *IEEE CIT*, Aug 2017, pp. 308–313.

[16] The Broadband Forum, "TR-069: CPE WAN Management Protocol," 2018, Issue 1 Amendment 6. URL https://www.broadband-forum.org/download/TR-069_Amendment-6.pdf.

[17] M. Boucadair, R. Penno, and D. Wing, "RFC 6970: Universal plug and play (UPnP) internet gateway device-port control protocol interworking function (IGD-PCP IWF)," Internet Engineering Task Force, 2013. [Online]. Available: https://tools.ietf.org/html/rfc6970

[18] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "RFC 3489: STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," Internet Engineering Task Force, 2003. [Online]. Available: https://tools.ietf.org/html/rfc3489

[19] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators." in *USENIX Annual Technical Conference, General Track.*, 2015.

[20] C. Seaman, "Threat advisory: Mirai botnet," Akamai Threat Advisory, 2016. [Online]. Available: https://www.akamai.com/us/en/multimedia/documents/stateof-the-internet/akamai-mirai-botnet-threat-advisory.pdf

[21] "Shadon: The search engine for IoT devices ," https://www.shodan.io/.

[22] M. Jethanandani, L. Huang, S. Agarwal, and D. Blair, "RFC 8519: Network Access Control List (ACL) YANG Data Model," Internet Engineering Task Force, June 2019. [Online]. Available: https://datatracker.ietf.org/doc/rfc8519/

[23] L. Su, "MUD is officially approved by IETF as an internet standard, and cisco is launching MUD1.0 to protect your iot devices," *Cisco Blogs*, May 2019. [Online]. Available: https://blogs.cisco.com/security/mud-is-officially-approved-by-ietf-as-an-internet-standard-and-cisco-is-launching-mud1-0-to-protect-your-iot-devices

[24] J. Hong, A. Levy, L. Riliskis, and P. Levis, "Don't talk unless i say so! securing the internet of things with default-off networking," in *IEEE/ACM IoTDI*, April 2018, pp. 117–128.

[25] M. Ranganathan, "Soft MUD: Implementing Manufacturer Usage Descriptions on OpenFlow SDN Switches," in *International Conference on Networks (ICN)*, 2019.

[26] A. Hamza, H. H. Gharakheili, and V. Sivaraman, "Combining MUD Policies with SDN for IoT Intrusion Detection," in *IoT S&P*, 2018.

[27] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity," in *Proceedings of the 2019 ACM Symposium on SDN Research*, ser. SOSR '19. New York, NY, USA: ACM, 2019, pp. 36–48. [Online]. Available: http://doi.acm.org/10.1145/3314148.3314352

[28] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vswitch," in *NSDI*, 2015, pp. 117–130. [Online]. Available: http://www.openvswitch.org/

[29] Cisco Systems, Inc., *Cisco Nexus 1000V Quality of Service Configuration Guide, Release 4.0(4)SV1(3)*. Cisco Systems, Inc., 2016, ch. DSCP and Precedence Values.

[30] A. Bremler-Barr, H. Levy, and Z. Yakhini, "Iot or not: Identifying iot devices in a shorttime scale (technical report)," https://www.dropbox.com/s/fkznax3dzqlf13l/IoTorNot-report.pdf.

[31] The Broadband Forum, "TR-098: Internet Gateway Device Data Model for TR-069," 2014, Issue 1 Amendment 2 Corrigendum 1. URL https://www.broadband-forum.org/download/TR-098_Amendment-2_Corrigendum-1.pdf.

[32] ——, "TR-181: Device Data Model for TR-069," 2011, issue 2 Amendment 2. URL https://www.broadband-forum.org/download/TR-181_Issue-2_Amendment-2.pdf.

[33] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying IoT traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2017*, 2017.

[34] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *4th International Conference on Information Systems Security and Privacy*, 2018.

[35] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers and Security*, 2012.

[36] G. Acar, D. Y. Huang, F. Li, A. Narayanan, and N. Feamster, "Web-based attacks to discover and control local iot devices," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*. ACM, 2018, pp. 29–35.

[37] A. Schiffer, "How a fish tank helped hack a casino," 2017. [Online]. Available: https://www.washingtonpost.com/news/innovations/wp/2017/07/21/how-a-fish-tank-helped-hack-a-casino/?noredirect=on&utm_term=.fb9aad71c166

[38] Ryu SDN Framework Community, "Ryu SDN Controller," 2017. [Online]. Available: https://osrg.github.io/ryu

[39] GenieACS Inc, "GenieACS - fast , lightweight TR-069 ACS," 2019. [Online]. Available: https://genieacs.com

[40] I. Livadariu, K. Benson, A. Elmokashfi, A. Dhamdhere, and A. Dainotti, "Inferring carrier-grade nat deployment in the wild," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 2249–2257.