IDC HERZLIYA
Efi Arazi School of Computer Science

The Interdisciplinary Center, Herzliya
Efi Arazi School of Computer Science
M.Sc. Program - Research Track

MUDirect: Protecting P2P IoT Devices with MUD

by
Avraham Shalev

M.Sc. dissertation, submitted in partial fulfillment of the requirements
for the M.Sc. degree, research track, School of Computer Science
The Interdisciplinary Center, Herzliya

September 9, 2021

This work was carried out under the supervision of Prof. Anat Bremler-Barr from the Efi Arazi School of Computer Science, The Interdiciplinary Center, Herzliya.

Abstract

Manufacturer Usage Description (MUD) is a new, whitelist-based cybersecurity framework that was recently proposed by the IETF to cope with the huge attack surface and a constantly increasing number of IoT devices connected to the Internet. MUD allows the IoT manufacturers themselves to publish the legitimate communication patterns of their devices, making it easier for security devices to enforce this policy, filter out non-complying traffic, and block a device in case it has been compromised.

Typically, MUD includes a set of legitimate endpoints, specified either by domain names or by IP addresses, along with the legitimate port numbers and protocols. While these descriptions are adequate when IoT devices connect (as clients) to servers (e.g., services in the cloud), they cannot adequately describe the cases where IoT devices act as servers to which endpoints connect [1]. These endpoints (e.g., users' mobile devices) typically do not have fixed IP addresses, nor do they associate with a domain name. In this case, accounting for 78% of IoT devices we have surveyed, MUD degrades nowadays to allow all possible endpoints and cannot mitigate any attack.

In this work, we evaluate this phenomenon and show it has a high prevalence today, thus harming dramatically the MUD framework security efficiency. We then present a solution, MUDirect, which enhances the MUD framework to deal with these cases while preserving the current MUD specification. Finally, we have implemented our solution (extending the existing osMUD implementation [2]) and showed that it enables P2P IoT devices protection while having minimal changes to the osMUD code.

Contents

1 Introduction

IoT devices are powerful enough to host malicious code but, for economic and technological reasons, they do not have the means to protect themselves from being hacked. Thus, the billions of IoT devices are fertile ground for many attacks of different kinds: DDoS attacks, privacy infringing attacks, data leakage, and physical break-ins [3] [4]. A recent initiative that attracts a significant attention, both in industry and academia, calls for IoT device vendors to provide a *Manufacturer Usage Description (MUD)* for their products [5]. These descriptions, called MUD files, consist of whitelists describing the devices' legitimate communication, specified by the domain names (or seldom with IP or MAC addresses) of legitimate endpoints with which each device model may communicate and with which ports and protocols. The main idea behind MUD is that the vast majority of IoT devices have a very succinct and unique traffic characteristic (namely, the number of destination addresses it is authorized to communicate with is small). A typical IoT device, like a camera or a power outlet, may communicate with at most a handful endpoints in the Internet (e.g., the server that provides the device with computational power in the cloud, time services server (NTP), login services and DNS resolving services). Note that MUD is not feasible to standard PCs (desktops, laptops, or mobiles), as, in contrast to IoT devices, they may communicate with a huge number of different servers, for example, it may access nearly any HTTP server in the world.

While MUD is a significant first step in reducing the attack surfaces of IoT devices, its white-list approach does not cover all kinds of IoT devices [1].

Specifically, MUD is an adequate solution when the device acts as a *client* and the endpoints are known servers (such as manufacturer, time, login, and other servers) and thus can be defined a priori. To access the IoT device remotely, both users (through either their mobile devices or desktops) and IoT devices connect to a server (usually located in the cloud) and all communication is done through that server. This case has an important deployment advantage when the device is behind a Network Address Translation (NAT) service.

On the other hand, MUD is not adequate when the IoT device itself acts as a *server* to which the user connects *directly* (See Fig. 1). This is typically done when the IoT and the user endpoints are in the same LAN (implying the communication is intra-LAN traffic), or when the connection is done by P2P communication [6] that bypasses the NAT (using either static port-forwarding[7], UPnP dynamic port forwarding [8], or Hole-Punching techniques [9], such as STUN/ICE protocols[10, 11]). In this paper, we will provide a solution (through MUD extension) for both kinds of such direct communication, which we collectively call *direct communication*. We note that there are various reasons to implement such direct communication for IoT devices, including cost reduction for cloud server maintenance, latency reduction (e.g., for streaming application), and mitigating privacy concerns.

In MUD philosophy, there is one MUD file per firmware, describing all devices that use the same firmware. However, user endpoints in P2P communication on the *specific* user are not fixed. Thus, in order to allow legitimate communication for all users with the same firmware, the MUD rules degenerate to allow all possible endpoints and thus cannot mitigate any attack. Recall that the infamous Mirai [12] targeted devices through their P2P communication, and, as reported in Shodan [13], there are many IoT devices with iP2P communication patterns that are vulnerable. Moreover, there are known successful intra-LAN attacks on IoT devices [14, 15]. We note that most devices that act as *servers* for some endpoints, may act also as *clients* for other endpoints.

In this paper, we first examine the prevalence of direct communication (See Section 4). We take a variety of IoT devices [16, 17] and categorize the devices that use P2P communication and Intra-LAN communication. Our results shows that 78% of the devices are using Intra-LAN traffic, while 17% of the devices are using P2P communication. This implies that *78% of the devices have inadequate MUD description*.

We then present our solution, called MUDirect, that extends MUD while keeping MUD current specification (see Section 5). MUDirect has three parts: an application deployed in the user's device, MUDirect service (that can reside, for example, in the cloud), and a placeholder in the legitimate endpoint value in the MUD file. First, the MUDirect service generates a unique URL for each user device and notifies the application on that device. The application, is like a dynamic DNS client is responsible for keeping track of the user device's IP address (that changes over time, e.g., when switching between networks) and updating the MUDirect service, so that the mapping between the URL and the IP address is always up to date. This user-specific URL is registered in the MUD file of the IoT device as a legitimate domain. Note that the initial value of this field (that must be the same in all devices with the same firmware) is a placeholder (in our implementation, `$owner-unique-domain$`). The component in the MUD architecture that is responsible for replacing the placeholder is the MUD manager (which, in general, requests, receives and processes all MUD files); as the placeholder mechanism already exists in the MUD manager, our extension requires only a small modification in the manager. The MUD manager obtains the correct

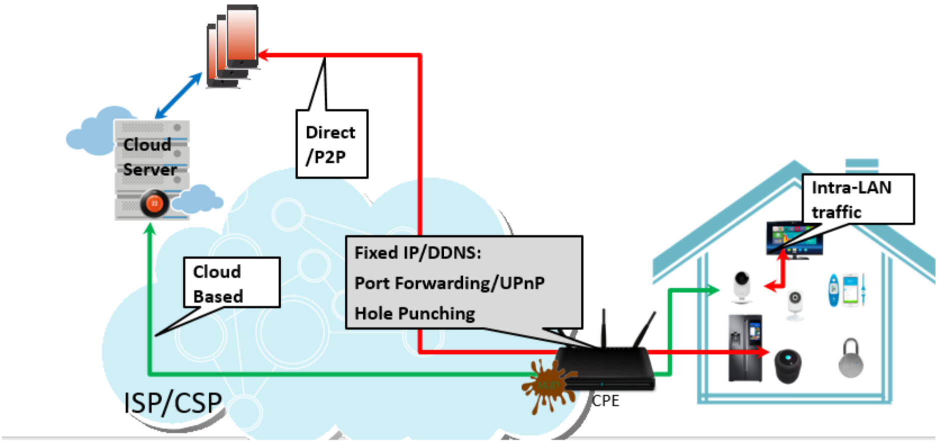

Figure 1: The three main communication patterns of IoT devices: (1) The IoT device and the user endpoint act as clients and connect to the cloud server, which acts as a proxy; (2) The IoT device acts as a server with P2P communication; (3) The IoT device acts as a server with intra-LAN communication.

URL from the MUDirect service, which stores the information of the users' endpoints. Notice that the MUDirect service can reside either within the IoT vendors' MUD servers, in the cloud (namely, in a "MUDirect as a service" offering), or in the MUD manager itself. The latter case is especially appealing for monitoring intra-LAN communication as it eliminates the need for communication outside the LAN, albeit it requires an additional modification to the MUD manager.

In Section 6, we present the implementation of MUDirect, whose code is available as open-source in [18]. Our implementation includes both the modification to the MUD manager (specifically, a modification of osMUD [2]), applications for both android and iOS devices, and the corresponding MUDirect services.

We then analyze the false positive and false negative rates of MUDirect and discuss approaches to minimize them. Uniquely, MUDirect accuracy challenges stem from the fact that, in MUDirect, the approved endpoints are of the *users* and not of *servers* (as in MUD). False positives (namely, alerting although the traffic is legitimate) happen when the user URL is not mapped to the up to date IP address of the endpoints. We discuss several cases when this inconsistency occurs due to some DNS phenomena and present ways to eliminate it. False negatives can happen when the legitimate user and an attacker have the same IP address, for example, in the rare case when they are behind the same (carrier grade) NAT or VPN [19] and receive the exact same IP from the IP address pool.

We conclude by showing that MUDirect is scalable due to the low change rate of IP addresses of user endpoints. Finally, we discuss the possibilities to extend MUDirect to include more types of traffic, most importantly when a manufacturer uses a *broker* to dynamically select servers for its client-based IoT device.

2 Background

2.1 Manufacturer Usage Description (MUD)

MUD is an Internet standard [5, 20] that aims at reducing IoT devices' attack surface, by describing the devices' proper traffic patterns. Any traffic that does not conform with this description is considered malicious and can be, for example, blocked. These descriptions are provided by the IoT manufacturers, in *MUD files*.

MUD files consist of an Access Control Lists (ACLs), each with several Access Control Entries (ACEs). Each ACE in a typical match-action rule whose matching predicate may consist of a MAC address, an IP address, an IP prefix, a domain name (which should be resolved to an IP address), a protocol number, port numbers (or range of numbers), and the traffic direction (namely, which device has initiated the connection). The corresponding action is typically either "accept" or "drop" (where the default rule is to drop traffic that does not correspond to any ACE, as the MUD file specifies a white-list). See example in Fig. 2.

The MUD framework itself consists of several components. A *MUD manager* (sometimes called *MUD controller*), is responsible for obtaining and processing the MUD information. For each IoT device, the MUD manager first obtains the MUD file from its manufacturer's *MUD server*. The location of the MUD server corresponding to a specific IoT device is stored as *MUD URI* in the device's firmware, and can be obtained by the MUD manager in a variety of ways as specified in [5] (most commonly, through a dedicated option in the DHCP protocol, which the IoT device executes to connect to the network). With the MUD file at hand, the MUD manager parses file and installs the corresponding ACL rules on a network security device (such as firewall or AAA server). We note that in some settings (e.g., home networks), the MUD manager, as well as the security device, may reside within the CPE (namely, the home gateway router).

```
{
  "ace": [
    {
     "name": "from-ipv4-blipcarebpmeter-1",
     "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"
     }
    }
  ]
}
```

Figure 2: An example of a single Access Control Entry (ACE) in a MUD file of a Blipcare BP meter. Note that the endpoint address is specified by a domain name, `tech.carematix.com`, that is resolved to an IP address. This specific MUD file consists of three ACLs, with a total of 10 ACEs, describing the entire legitimate traffic of the device [17].

2.2 P2P communication methods

We overview the common P2P methods used by legitimate endpoints to connect directly to IoT devices. The main challenge of such a direct connection is to bypass the NAT service, which is commonly deployed in the CPE. Recall that NAT entries are created when traffic is sent from the a device within the LAN externally and allow bidirectional traffic. When traffic is initiated externally, the corresponding NAT entry is not created, implying connections cannot be established. We note that intra-LAN communication is straightforward, as both peers are on the same LAN, and therefore, such communication does not traverse a NAT.

- *Static (Manual) Port Forwarding*. The CPE is configured manually, so that each packet sent from an external (namely, WAN) location with a specific destination port, will be forwarded to a specific device on the LAN side. In such a case the external device needs to know only the IP address of the CPE and that specific port, in order to reach the device.

- *Dynamic Port Forwarding* is similar to the static port forwarding, but does not require manual configuration. Instead, it uses the *Universal Plug and Play (UPnP)* [8] protocol, in which the (IoT) device sends a UDP packet that triggers a port forwarding entry setup.

- *Hole Punching* [21] techniques, which usually use the *Session Traversal Utilities for NAT (STUN)* protocol [10]. Both peers first open a connection to a third server, a STUN server [10]; as this traffic is initiated from the LAN side, a NAT entry is created by the device, but it allows only communication from the STUN server. In the second stage, the STUN server sends each of the two peers, the other peer's (public) IP address and port number (that it used to connect to the STUN server). Then, both peers try to connect to each other. While the first peer fails, as its traffic will be blocked by the NAT (only traffic from the STUN server is allowed), it will create an entry in the NAT with the other peer's correct IP address and port number (this operation is referred to as "punching" a hole in the NAT). This implies that the second peer will succeed to connect and the (bidirectional) connection can be established. We note that while STUN is more secure than static/dynamic port forwarding, it does not work in all NAT settings (for example, under Symmetric NAT [10]). Alternative, but more resource-intensive solution, is to use the external server as a relay between both peers (this is done, for example, in *Traversal Using Relays around NAT (TURN)* protocol [22]). To be on the safe side, many IoT manufacturers use *Interactive Connectivity Establishment (ICE)* methodology to discover the optimal means of connectivity; ICE basically first tries to use STUN to connect the peers and turns to TURN only upon a failure.

In all above cases, the legitimate endpoints are not known to the firmware (that is common to all devices of the same type). Hence, from MUD perspective, we need to allow *any* endpoint, no matter what its address is, to connect to the device. See example in Fig. 3. Naturally, this exposes the device to a variety of attacks and cast a doubt on the effectiveness of MUD in these cases.

```
{
  "ace": [
    {
        "name": "to-ipv4-samsungsmartcam-8",
            "matches": {
             "ipv4": {
                "protocol": 17
             },
             "udp": {
                "source-port": {
                    "operator": "eq",
                    "lower-port": 1024,
                    "upper-port": 65535
                 }
             }
            },
            "actions": {
              "forwarding": "accept"
            }
        }
     ]
}
```

Figure 3: An example of a single Access Control Entry (ACE) in a MUD file of a Samsung Smart Camera, that allows incoming UDP traffic under user ports (namely, any port larger than or equal to 1024) from *any* destination [17].

3 Related Work

MUD-based solutions are still being shaped, and there are several recent works that focus on extending MUD and implementing MUD in different settings. Such works include MUD for mobile 5G network [23], MUD in SDN environment [24, 4, 3, 25], using MUD to handle DDoS volumertic attacks [26], extending MUD to capture flow statistics [27], and enabling user-defined rules in MUD [28] [25]. At this stage, there are no sufficient number of IoT manufacturers that publish MUD files, putting a high barrier in adopting the technology. Thus, there is an extensive line of works that propose tools for easier configuration and creation of MUD files [29, 30], without cooperation of the manufacturers themselves.

The fact that MUD is not adequate for P2P traffic was first mentioned (without a solution) in [29].

In our previous paper [1], we have presented a system to implement MUD through a VNF deployed within the ISP. We have also described an initial design that deals with P2P traffic in this ISP setting, but without any implementation. This papers takes our previous design as a starting-point, provides a detailed implementation in the CPE and introduces MUDirect—a complete solution that addresses many challenges that were discovered during the implementation.

The work [31] suggests also a solution to P2P communication, but the suggested solution changes the MUD architecture and protocol drastically. Specifically, it requires to add two new entities to the MUD architecture: a *Mobile MUD Enforcement Engine (MMEE)* that runs on the smartphone and a *Local MUD Manager (LMM)* that runs on the MUD Manager. It also requires a new protocol between MMEE and LLM. MUDirect, on the other hand, provides a seamless integration into MUD framework.

4 Motivation: Prevalence of Direct Communication in IoT Devices

This section provides an intuition on the prevalence of IoT devices that use direct communication. Recall that direct communication includes two categories: intra-LAN communication (where both peers are on the same LAN) and P2P communication (where the peers resides in different networks). Since there are no currently-available MUD files, we have used the MUD profiles from [17], that were generated automatically by the MUDgee tool [29]. MUDgee generates the MUD file by observing the traffic to/from the IoT device and striving to characterize it in a succinct manner. It is important to notice that, in MUDgee, if an IoT device connects to multiple IP addresses without a preceding domain-name resolution (namely, a DNS request), an ACE that accepts traffic from all destinations (on the relevant ports) will be created. See Figure 3. In addition, MUDgee identifies intra-LAN traffic if the destination MAC address of a packet is different than the gateway router's MAC address. We note that using MUD files from MUDgee is customary to MUD research nowadays.

Analyzing the above-mentioned 28 MUDgee-created MUD files, corresponding to 28 different IoT devices, yields only a lower bound on the actual number of devices that use direct communication. Specifically, when direct communication is relatively rare, it might not appear in the traffic capture used to generate the MUD file, and therefore, be missed [32].

Thus, we have also looked for additional information using firewall configuration for these IoT devices over the Internet. Such information is publicly-available, as P2P communication often requires specific firewall rules configuration (e.g., port forwarding require allowing traffic to/from specific ports). Specifically, using vendor specifications [33], vendor knowledge-base and help centers [34, 35], unofficial support sites [36], other academic papers [37], and github code written for some of the IoTs (e.g. [38]), we have found additional IoT devices that are using direct communication.

Fig. 4 shows the prevalence of direct communication in the 28 IoT devices we have surveyed: 22 out of the 28 devices use intra-LAN communication, while 5 out of the 28 devices use P2P communication. P2P communication is mostly done by camera devices using STUN. Naturally, some devices use both intra-LAN and P2P communication (in fact, in our survey, all devices with P2P communication also used intra-LAN communication). We note that in our sample only 6 devices (namely, 21% of the devices) have neither intra-LAN nor P2P communication. These IoT devices act only as client, and thus only for these devices an adequate MUD can be created.

We note that although our survey is limited, similar findings were reported in [37], where 45 IoT devices were analyzed, showing that half of them are using intra-LAN communication.

5 MUDirect Design

In this section, we overview our MUDirect solution. In most cases, users access IoT devices through a dedicated application; therefore, we will use the term *IoT app* to refer to these apps and *IoT app devices (IADs)* for the devices with the IoT apps.

We extend the MUD specification to allow a *secondary virtual manufacturer (SVM)*, whose sole purpose is to deal with P2P communication. The SVM implements our MUDirect service that is able to deal with all P2P communication; it can be suggested as a service to the IoT manufacturer or being part of the IoT manufacturer's MUD server.

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

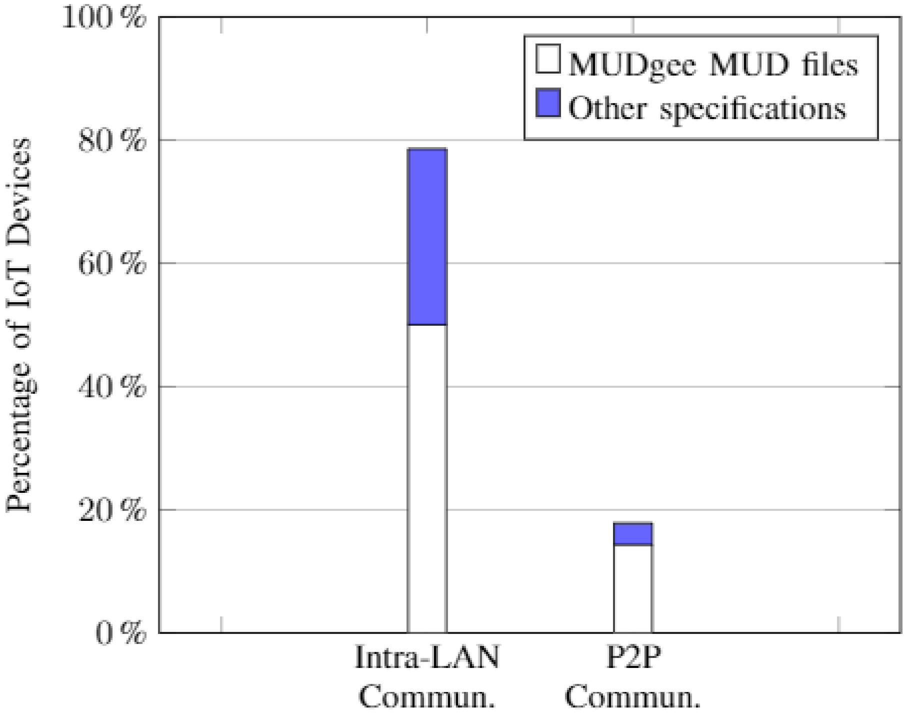

Figure 4: The percentage of IoT devices, out of the devices that appeared in [29], using both kinds of direct communication. The source of data is depicted in color. The white bars represent devices whose direct communication was observed by MUDgee; the blue bars represent devices whose direct communication was not observed by MUDgee but appears in one of the other sources of information we have surveyed.

the corresponding mapping. In a nutshell, this is done by installing a *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 and a unique domain.

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, it should specify that its device supports direct communications. This can be done by inserting a unique placeholder (in our implementation, `$owner-unique-domain$`) to the

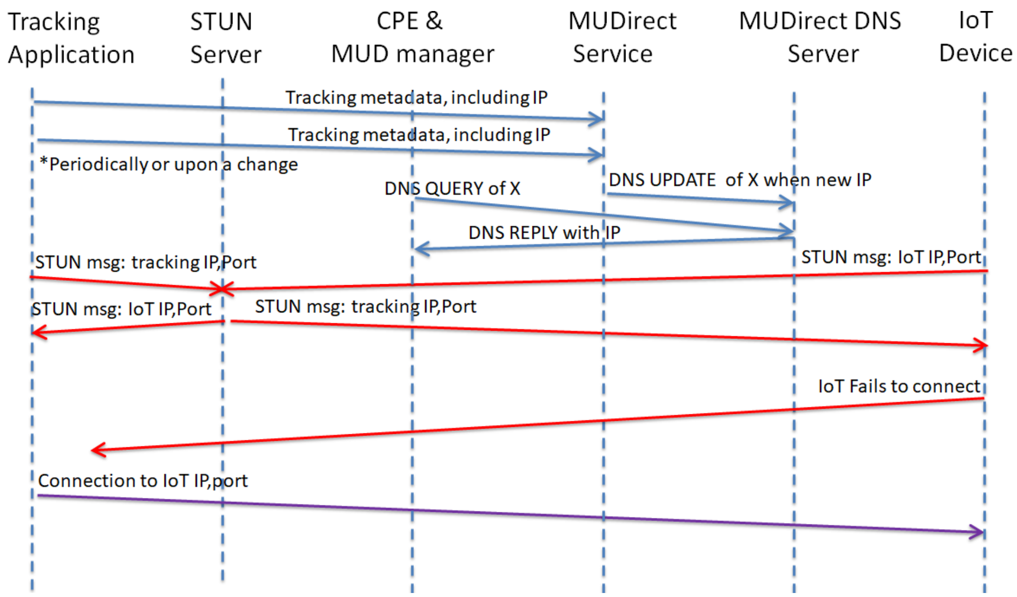
Tracking Application
STUN Server
CPE & MUD manager
MUDirect Service
MUDirect DNS Server
IoT Device
Tracking metadata, including IP
Tracking metadata, including IP
*Periodically or upon a change
DNS QUERY of X
DNS UPDATE of X when new IP
DNS REPLY with IP
STUN msg: tracking IP,Port
STUN msg: IoT IP,Port
STUN msg: IoT IP,Port
STUN msg: tracking IP,Port
IoT Fails to connect
Connection to IoT IP,port

Figure 5: Illustration of the MUDirect message exchange after the initial sign-up process is completed. The IAD initiates a P2P connection (using STUN) to the IoT device. Notice that the IoT device is not aware of either the MUD or the MUDirect systems.

file.

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

The tracking application reports to the SVM mapping service whenever the external IP address of the IAD is changed, along with the account identifier, the unique domain, and the external and internal IP addresses. Upon a change, the mapping service simply updates the corresponding DNS record in the authoritative DNS server. Fig. 5 shows the message exchange of our solution, where the IAD and the IoT device are connected with STUN. Note that the MUDirect technique is oblivious to the type of the P2P communication.

5.1 MUDirect sign-up process

The most involved operation of the SVM is when a new user sign-up with the service, which is illustrated in Fig. 6. A user who wants to use this protection needs to install the 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-created account with a specific LAN, we require that the tracking application will be connected, in the first time, from within that LAN. The operations continue in the following steps, where the tracking application acts as a (virtual) IoT device, while the SVM mapping service acts as its manufacturer: (i) The tracking application broadcasts a MUD URL, which includes the SVM parent domain and the identifier of the newly created account; (ii) The MUD manager, observing that LAN, fetches this MUD URL by contacting its manufacturer as specified in the URL. This request goes to the MUDirect service of the SVM; (iii) The SVM 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 MUDirect service replies to the MUD manager with the MUD file that includes the account unique domain (namely, the value of `$owner-unique-domain$`); (v) The MUD manager extracts this unique domain from the MUD file, replaces all pending MUD files with `$owner-unique-domain$`, and save the domain name for a later usage.

5.2 MUDirect solution for intra-LAN communication

When the IoT app resides on the same LAN as the IoT device, one can use, in some cases, the MAC address of the IAD instead of its IP address, implying a mapping between an IP address and a unique domain is no longer required. The IAD's MAC address can be inserted to the MUD file, using an additional placeholder.

However, in some cases, the IAD's MAC address is not visible to the MUD manager even if they reside in the same LAN. This is most common in the presence of extenders that rewrite MAC addresses within the LAN. Thus, in such settings, the IAD's internal IP address is needed, and we configure the tracking application to send both internal and external addresses, having at least two DNS records for that IAD. We note the case where the communication between different IoT devices in the same LAN should be monitored was recently solved in [24].

We can further improve the implementation efficiency of the intra-LAN case,

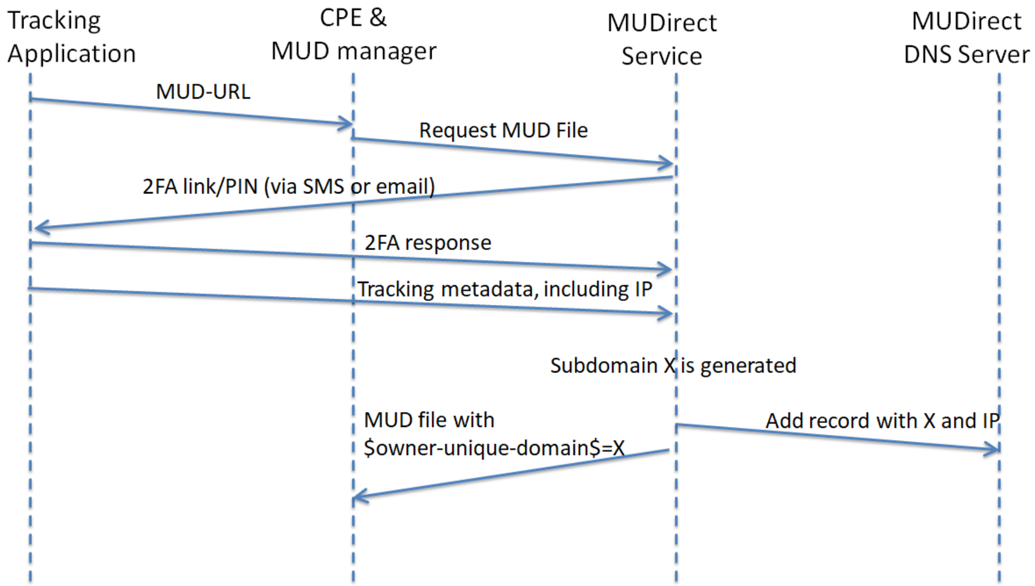

Figure 6: Illustration of the sign-up process of a IAD with the MUDirect Service and the corresponding CPE. The sign-up process must be completed when the IAD is connected through the CPE.

by implementing it entirely within the LAN. In such an implementation, the MUD manager will keep track of the IAD internal IP address, thus eliminating the need to perform external DNS requests. This is done by communicating tracking information through dedicated DHCP entries, where additional precautions are taken (namely, using mechanisms that are similar to HOTP [39]) to prevent attacks, where a malicious device is impersonating the IAD.

6 Implementation

In this section we discuss some implementation details of the MUDirect components and the specific challenges in the implementation of each components (see Fig. 7).

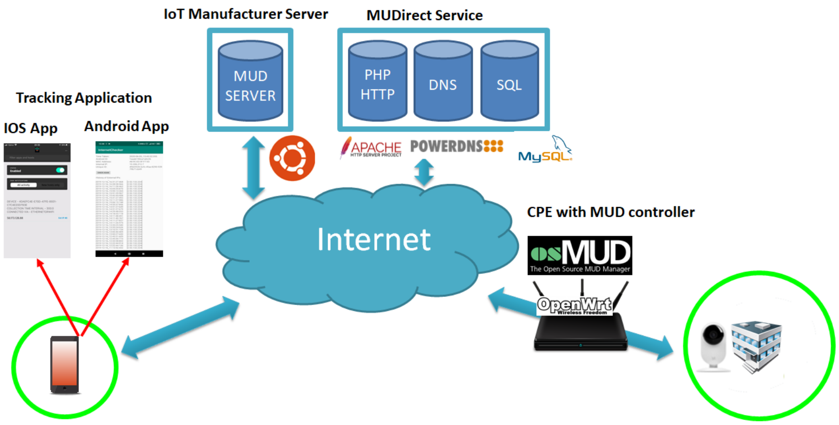

Figure 7: MUDirect Implementation Components

Our MUDirect implementation contains:

- *Tracking application:* We have implemented iOS and Android apps to run on users' mobile devices. We have not implemented an app to run on a desktop. Notice that mobile devices are significantly more challenging than desktops, since their IP addresses change more frequently as they move around.
- *MUDirect service:* The main components of MUDirect Service are: (i) a DNS server; (ii) a database that stores the domain records; and (iii) an HTTP server that receives the information and requests from both the tracking application and the MUD manager (where the latter sends requests only on initial sign-up process of each IAD). We have used open-source implementations for these components: powerDNS for the DNS server, mySQL for the database, and Apache for the HTTP server.
- *MUD manager:* We have extended osMUD [2] manager to support MUDirect. The MUD manager run on a TPLink CPE with OpenWRT. Our TPlink consists of a 64 MB RAM, 580 MHz CPU, and only 8MB flash storage, mostly used by the operating system. Thus, the amount of resource available is limited.

Next we explain the specific challenges we have encountered in the implementation of each one of the components.

6.1 Tracking application

In the implementation phase, we have found out that, in some cases, mobile endpoints use different outgoing IP addresses simultaneously (each one for different outgoing connections). This happens because of either a NAT service with several external IP addresses or because of an intermediate outgoing gateway load balancer (usually in enterprise networks). Therefore, our tracking application obtains a set of external IP addresses of the IAD using a set of “whatismyip” websites (e.g., myexternalip.com, myip.com, ipinfo.io), where queries are done with different port numbers (namely, by opening different sockets). Furthermore, we use HTTPS requests to bypass proxy cache servers that work on HTTP traffic. The tracking application notifies the SVM (namely, the MUDirect service) only upon a change in the IAD’s IP addresses.

6.2 MUDirect service

In our implementation, the SVM owns a parent domain and generates a unique randomly-generated sub-domain under this domain for each IAD. The SVM maintains the authoritative DNS server of that parent domain (and all its sub-domains) to ensure that changes to DNS records, generated by the *MUDirect service*, are reflected in subsequent DNS queries.

The correctness of MUDirect necessitates that the MUD manager (which in charge on generating ACLs) receives the most up to date DNS information. However, caching mechanisms at DNS resolvers yield that the MUD manager may receive stale information before entries are updated in all resolvers.

One solution to this problem is to force the MUD manager to use MUDirect’s authoritative DNS server directly, when retrieving information for MUDirect corresponding domain (thus, bypassing the ISP resolver and its cache, for these specific DNS queries).

However, we have found out that some ISPs perform *DNS interception [40]*, implying that the ISP may redirect DNS traffic aimed at our authoritative server and forward it to its own DNS resolver (which might contain stale information). We have performed DNS requests from different locations and different ISPs over the globe using probes from RIPE ATLAS [41]. Fig. 8 shows that more than 2% of the ISP worldwide perform DNS interception (where we checked 963 ASes by using more than 4000 probes); in addition, it was previous reported that DNS interception is performed in China [40].

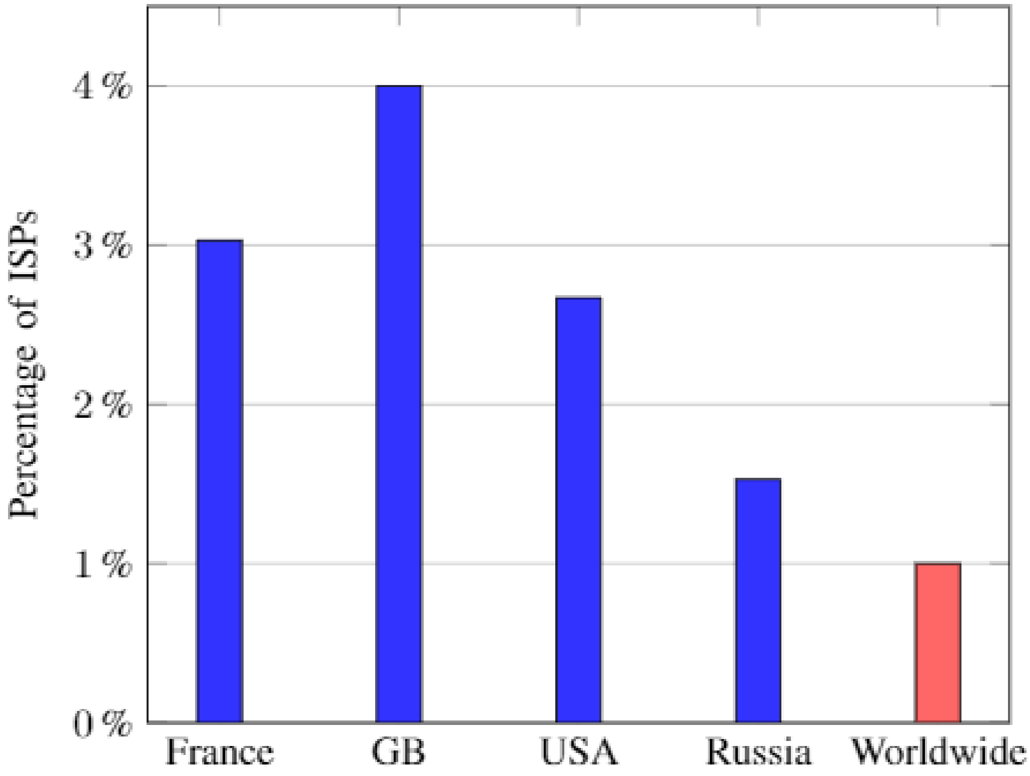

Figure 8: Percentage of ISPs that perform DNS interception for representative countries.

To circumvent this problem, we add a random value as a sub-domain; in such a case, each DNS query bypasses all caching mechanisms and reach our DNS authoritative server, which ignores the random part and returns the up to date information.

6.3 The MUD manger

As a MUD manager, we have used osMUD with several extensions. A first extension is a code that replaces the placeholder `$owner-unique-domain$` in the IoT device's MUD with the domain of the IAD. A more involved extension deals with how domain names, that appear extensively in MUD files, are resolved to obtain IP addresses (required for monitoring actual packets). In general, MUD managers have two options for such resolutions: either to actively issue DNS requests periodically or to sniff the DNS traffic of the IoT devices [32] (namely, the MUD manager needs to be updated only when the IoT device is updated, which happens only after a corresponding DNS request and response).

However, sniffing DNS traffic is not applicable to our case as, in direct communication, the IoT device acts as a server and does not issue any DNS request. Thus, we have changed osMUD to perform active DNS requests, for SVM domains only, in 5 seconds intervals.

We note that we did not observe any delays in the IADs due to outdated DNS resolution information. We speculate that this is because IADs require anyhow to create new connections upon IP address changes; the delays due to reconnections mask the delays that may have been caused by outdated information.

6.4 Validation experiment

We have validated our solution by a lab experiment, with a Yi camera [42] and two types of IADs, trying to connect the the Yi camera using P2P communication (more specifically, using STUN protocols). The first type is of authorized IADs, that were registered to MUDirect beforehand; in all the tests, the authorized endpoints succeeded to enter the camera without a noticeable delay. The second type included unauthorized IADs, that play the role of an attacker; all of these IADs were blocked successfully by our osMUD implementation.

The experiment was done with IPv4 since nowadays IPv4 is still being used by the vast majority of internet users [43]. We note that MUDirect's design and implementation fully support IPv6.

6.5 MUD implementation complexity

An important factor in the implementation complexity is the frequent changes of IADs' IP addresses, as it affects the number of updates the SVM (or more specifically, the MUDirect service) receives. We have distributed our tracking applications to 51 Android & iOS users from Israel, Australia, and Singapore, recording a total of 2008 one-hour samples. Fig. 9 shows the number of IP address changes per hour. In the majority of the cases 83% a device uses maximum two external IP Addresses per a hour, while there are rare cases that the IP has changed 7 times in an hour; additionally, as mentioned before we have observed that some devices use several IP addresses simultaneously. We note that our experiment was done during the COVID-19 pandemic, and therefore, it might underestimate the number of IP addresses used per IAD, as travel was sometimes restricted.

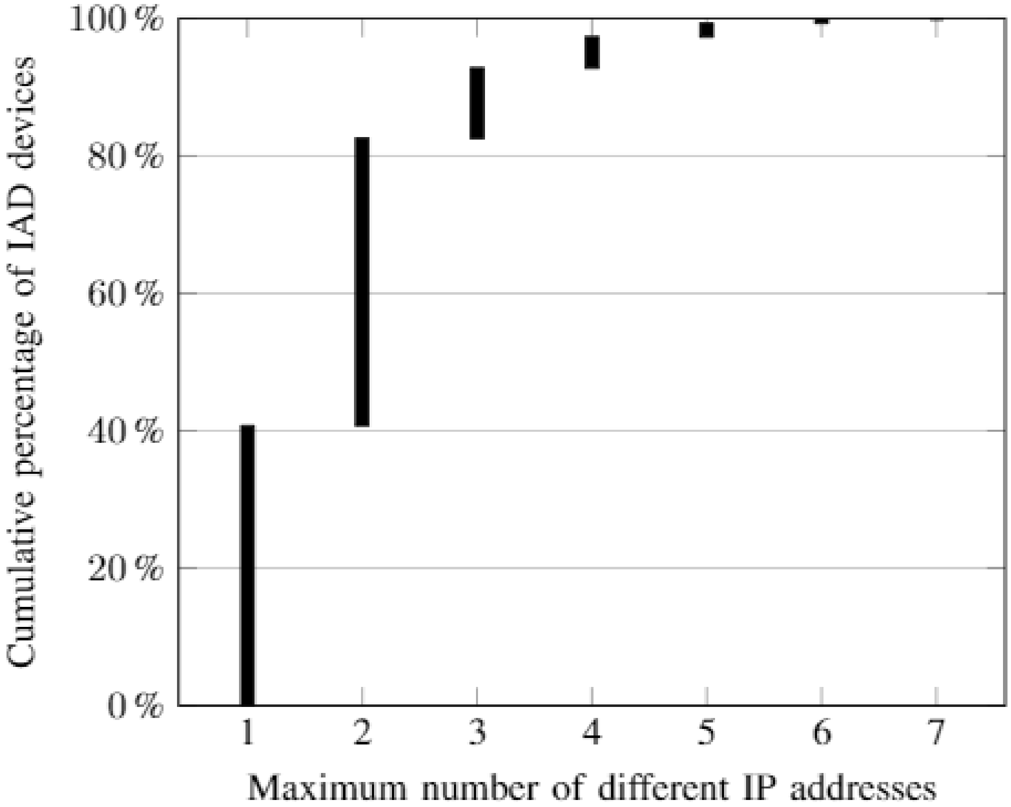

Figure 9: Cumulative Distribution Function (CDF) of the number of different IP addresses used by a single IAD in one hour.

Finally, we note that the number of database records required for our MUDirect service equals the number of users. As the frequency of DNS queries is 5 seconds, the number of DNS queries per second is one-fifth the number of users.

7 Accuracy

In rare cases, MUDirect may either cause a false alert (a.k.a. false positive) or miss a malicious activity (a.k.a. false negative).

A false negative may occur when an attacker succeeds to obtain the same IP address of the legitimate user. This situation is feasible if the legitimate user is in a network behind NAT, Carrier-Grade NAT (CGN), or VPN, and the attacker is able to join the same network and to receive the same IP address. However, the more customers are behind the same service (NAT,CGN or VPN) the more likely is that there are multiple outgoing IPs and the chance of receiving the same IP is small.

For example, in CGN [44], used by more than 90% of cellular networks [45], the maximal number of customers that share the same IP Address (sometimes referred as the CGN compression ratio, or CCR) is usually lower than 1 to 20 (1:16 at [46], 1:8 at [47], 1:14 at [48], 1:18 at [49], etc.). This implies that even in the case of a small-size ISP, with only a million customers, it is very unlikely (1 to 50,000 = 0.002%) that an attacker would share the same address of a specific legitimate user.

False positives may occur when a legitimate user has just updated its IP address, but the MUD manager has not updated its record yet. As we set the MUD manager query interval to 5 seconds, we did not observe any false positives in our experiments.

8 Future Work

MUD is a promising solution to reducing the attack surface of IoT devices. However, as we show, for most IoT devices, the current solution is insufficient as it does not deal with direct communication. Thus, we have presented MUDirect, a fully-functional open-source MUD framework, supporting both P2P and intra-LAN communication.

MUDirect service can be deployed by IoT Manufacturers or as a Software as a Service (SaaS) solution. SaaS has a clear economic incentive, since MUDirect requires a DNS record per legitimate endpoint, and the same record can be used for multiple IoT devices of the same user (even across multiple LANs).

We note that MUD is not an adequate solution also when the IoT device is a client, but a broker is used to dynamically select servers that host its legitimate endpoints. In such a case, the IoT vendor can use a similar mechanism as MUDirect: the MUD file may contain a correspond domain name for the broker decision (e.g. iotX.broker.com) and the IoT vendor X would update dynamically this record. In case the client location influences the broker decision, then a solution per client, with a placeholder in the firmware, should be used. Note that the solution is more complex in this case, since the record is per an IoT vendor, thus requiring continual cooperation and updates from the vendor itself.

9 References

[1] Y. Afek, A. Bremler-Barr, D. Hay, R. Goldschmidt, L. Shafir, G. Avraham, and A. Shalev, "Nfv-based iot security for home networks using MUD," in *NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, April 20-24, 2020*. IEEE, 2020, pp. 1–9. [Online]. Available: https://doi.org/10.1109/NOMS47738.2020.9110329

[2] "Open source mud manager." [Online]. Available: "https://osmud.org/

[3] 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

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

[5] 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/

[6] IAB and G. Camarillo, "Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability," RFC 5694, Nov. 2009. [Online]. Available: https://rfc-editor.org/rfc/rfc5694.txt

[7] "Port Forwarding - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Port_forwarding

[8] 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

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

[10] 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

[11] A. Keränen, C. Holmberg, and J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal,” RFC 8445, Jul. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8445.txt

[12] 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

[13] “Shadon: The search engine for IoT devices ,” https://www.shodan.io/.

[14] 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/

[15] 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.

[16] A. Hamza, D. Ranathunga, H. H. Gharakheili, T. Benson, M. Roughan, and V. Sivaraman, “Verifying and monitoring iots network behavior using MUD profiles,” *CoRR*, vol. abs/1902.02484, 2019. [Online]. Available: http://arxiv.org/abs/1902.02484

[17] “28 MUDgee created MUD Files.” [Online]. Available: https://iotanalytics.unsw.edu.au/mudprofiles

[18] “MUDirect implementation.” [Online]. Available: https://github.com/deepness/MUDirect

[19] A. G. Malis, D. A. Y. Lin, D. J. Heinanen, B. Gleeson, and D. G. Armitage, “A Framework for IP Based Virtual Private Networks,” RFC 2764, Feb. 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc2764.txt

[20] 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

[21] P. Srisuresh, B. Ford, and D. Kegel, “RFC 5128: State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs),” Internet Engineering Task Force, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5128

[22] P. Matthews, J. Rosenberg, and R. Mahy, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),” RFC 5766, Apr. 2010. [Online]. Available: https://rfc-editor.org/rfc/rfc5766.txt

[23] N. Mazhar, R. Salleh, M. Zeeshan, and M. M. Hameed, “Role of device identification and manufacturer usage description in iot security: A survey,” *IEEE Access*, vol. 9, pp. 41 757–41 786, 2021.

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

[25] A. Feraudo, P. Yadav, R. Mortier, P. Bellavista, and J. Crowcroft, “Sok: Beyond iot MUD deployments - challenges and future directions,” *CoRR*, vol. abs/2004.08003, 2020. [Online]. Available: https://arxiv.org/abs/2004.08003

[26] S. M. Sajjad, M. Yousaf, H. Afzal, and M. R. Mufti, “emud: Enhanced manufacturer usage description for iot botnets prevention on home wifi routers,” *IEEE Access*, vol. 8, pp. 164 200–164 213, 2020.

[27] S. Singh, A. Atrey, M. Sichitiu, and Y. Viniotis, “Clearer than mud: Extending manufacturer usage description (mud) for securing iot systems,” in *ICIOT*, 2019.

[28] P. Yadav, V. Safronov, and R. Mortier, “Enforcing accountability in smart built-in iot environment using mud,” in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings,*

Cities, and Transportation, ser. BuildSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 368–369. [Online]. Available: https://doi.org/10.1145/3360322.3361004

[29] 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.

[30] V. Andalibi, E. Lear, D. Kim, and L. J. Camp, "On the analysis of mud-files' interactions, conflicts, and configuration requirements before deployment," 2021.

[31] I. B. Fink, M. Serror, and K. Wehrle, "Extending MUD to smartphones," in *45th IEEE Conference on Local Computer Networks, LCN 2020, Sydney, Australia, November 16-19, 2020*, H. Tan, L. Khoukhi, and S. Oteafy, Eds. IEEE, 2020, pp. 353–356. [Online]. Available: https://doi.org/10.1109/LCN48667.2020.9314782

[32] "NCCoE NIST MUD Current Suggested Builds." [Online]. Available: https://www.nccoe.nist.gov/projects/building-blocks/mitigating-iot-based-ddos/securing-home-iot-devices

[33] "iHome Smart Plug Spec." [Online]. Available: https://cdn.ihomeaudio.com/media/product/files/iSP6X_2017_QSG_v10.pdf

[34] "Pixstar Knowledgebase article for Pixstar Photo Frame." [Online]. Available: https://pixstar.uservoice.com/knowledgebase/articles/442554

[35] "Withings BabyMonitor Article." [Online]. Available: https://support.withings.com/hc/en-us/articles/211667678

[36] "Ring Doorbell unofficial support answer." [Online]. Available: https://www.adscon.com/sites/blog/Lists/Posts/Post.aspx?ID=52

[37] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1362–1380.

[38] "August Doorbell unofficial github code." [Online]. Available: https://github.com/Samfox2/homebridge-videodoorbell/issues/21

[39] M. View, D. M'Raihi, F. Hoornaert, D. Naccache, M. Bellare, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm," RFC 4226, Dec. 2005. [Online]. Available: https://rfc-editor.org/rfc/rfc4226.txt

[40] B. Liu, C. Lu, H. Duan, Y. Liu, Z. Li, S. Hao, and M. Yang, "Who is answering my queries: Understanding and characterizing interception of the DNS resolution path," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1113–1128. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/liu-baojun

[41] "RIPE Atlas service." [Online]. Available: https://atlas.ripe.net

[42] "YiHomeCamera1080p." [Online]. Available: https://www2.yitechnology.com/yi-1080p-home-camera

[43] "Google Statistics IPv6 Users Percentage." [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html

[44] S. Perreault, I. Yamagata, S. Miyakawa, A. Nakagawa, and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)," RFC 6888, Apr. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6888.txt

[45] 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*, 2018, pp. 2249–2257.

[46] "RIPE Labs: Blockers to IPv6 Adoption." [Online]. Available: https://labs.ripe.net/Members/david_holder/blockers-to-ipv6-adoption

[47] "Carrier Grade NAT - Observations and Recommendations." [Online]. Available: https://www.rmv6tf.org/wp-content/uploads/2012/11/CGN_Observations_Recomendations-NAv6S_20121.pdf

[48] L. Rouleau, D. Serrano, B. Lecointe, F. Ravet, G. Coma, and P. Christou, "CNG direct injection spark-ignition engine with high turbulence and high compression ratio: numerical and experimental investigations," in *12th Conference of Gaseous-Fuel Powered Vehicles*. Stuttgart,

Germany: FKFS, Oct. 2017. [Online]. Available: https://hal-ifp.archives-ouvertes.fr/hal-02194896

[49] R. Hosmath, N. Banapurmath, S. Khandal, V. Gaitonde, Y. Basavarajappa, and V. Yaliwal, “Effect of compression ratio, cng flow rate and injection timing on the performance of dual fuel engine operated on honge oil methyl ester (home) and compressed natural gas (cng),” *Renewable Energy*, vol. 93, pp. 579–590, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096014811630194X

תקציר

Manufacturer Usage Description (MUD) הוא סטנדרט הגנה למכשירי IoT, מבוסס whitelist, שהוצע לאחרונה על-ידי ה IETF. MUD מאפשר לכל יצרן מכשיר IoT לפרסם את כללי התקשורת המצופים ממנו ולכן מקל על אכיפת הכללים ברשת בה המכשיר נמצא. במצב כזה, ניתן בקלות לזהות תקשורות לא צפויות שהמכשיר מבצע ולטפל בהם בהתאם. כללי התקשורת מוגדרים על-ידי רכיבי-קצה איתם המכשיר אמור לתקשר (בין אם על-ידי כתובת IP ובין אם על-ידי הדומיין שלהם), פורטים ופרוטוקולים מורשים לשימוש. בעוד ששיטה זו מתאימה להגנה על מכשירי IoT המשמשים כקליינטים שמתחברים לשרתים אחרים (שנמצאים לדוגמא בענן), היא איננה מתאימה למקרים בהם מכשיר ה IoT משמש כשרת בעצמו, אליו מתחברים רכיבי קצה אחרים (לדוגמא, סלולרי של משתמש) שאותם לא ניתן להגדיר בצורה קבועה (על-ידי דומיין או כתובת IP ייחודיים להם). במקרה כזה, MUD מוריד את רמת האבטחה ומאפשר לכל רכיב-קצה לגשת למכשיר ה IoT (על-מנת לא לפגוע בתפקודו), ולכן הגנת MUD כבר לא אפקטיבית. בעבודה זו אנו מראים שהתופעה הזאת רווחת – ולמעשה, קיימת ב 78% ממכשירי ה IoT אותם בדקנו. לאחר-מכן, אנו מציגים את MUDirect, המהווה פתרון לבעיה בצורה בה האפיון של MUD נשמר. לסיום, אנו מציגים את מימוש MUDirect (שהוא הרחבה למימוש osMUD), ומראים שהוא מאפשר הגנה למכשירי IoT בעלי תעבורת P2P.

עבודה זו בוצעה בהדרכתה של פרופ' ענת ברמלר-בר מבי"ס אפי ארזי למדעי המחשב, המרכז הבינתחומי, הרצליה.

הבינתחומי הרצליה
בית ספר אפי ארזי
למדעי המחשב

המרכז הבינתחומי בהרצליה

בית-ספר אפי ארזי למדעי המחשב

התכנית לתואר שני (M.Sc.) - מסלול מחקרי

MUDirect:
הרחבת MUD להגנה על מכשירי IoT
בעלי תעבורת P2P

מאת

אברהם שליו

עבודת תזה המוגשת כחלק מהדרישות לשם קבלת תואר מוסמך M.Sc.
במסלול המחקרי בבית ספר אפי ארזי למדעי המחשב, המרכז הבינתחומי הרצליה

ספטמבר 2021