DEMONS: Extended Manufacturer Usage Description to Restrain Malicious Smartphone Apps

Ina Berenice Fink*, Martin Serror*, Klaus Wehrle
Chair of Communication and Distributed Systems, RWTH Aachen University, Germany
{fink, serror, wehrle}@comsys.rwth-aachen.de
*Equal Contribution

***Abstract*—The growing popularity of the consumer IoT intensifies the risks for security and privacy breaches. It typically suffices to successfully attack a single IoT device to access the home network illicitly. This observation emphasizes the need for in-network security, complementing each device's security mechanisms with additional network-layer protection. Recently, the IETF proposed Manufacturer Usage Description (MUD) to limit network traffic of IoT devices to their required minimum. However, the tangled communication of IoT devices, e.g., connections to smartphones and smart speakers, is not covered by MUD. We propose Distributed Enforcement of MUD on Smartphones (DEMONS), extending central enforcement of MUD with distributed enforcement at authenticated smartphones to mitigate the threats of malicious apps and IoT devices by filtering malicious traffic close to its origin and preventing further spread. We discuss the security gains and demonstrate that the introduced overhead regarding latency, bandwidth, and power consumption has a negligible performance impact.**

***Index Terms*—security and privacy, smart home, Internet of Things, access control, policy-based network security, MUD**

I. Introduction

We observe an increasing number of locally and Internet-wide connected devices in smart homes [13]. However, this development bears severe security risks since these devices are known for their frequent security flaws [3], [18], [7]. Indeed, many already deployed Internet of Things (IoT) devices suffer from missing or weak encryption of communication [4], the use of well-known standard passwords [12], and complicated or lacking update mechanisms [19]. These vulnerabilities, among others, facilitate the massive infiltration of malware on IoT devices, e. g., to form large botnets. Even worse, a single compromised device in a home, e. g., a legacy device with known vulnerabilities, often suffices for attackers to gain local network access and infect other devices. Therefore, improving IoT and smart home security needs to address the emerging risks of illicit access to the local network, e. g., via a malware-laden smartphone [16] or by targeting legacy devices.

A promising approach in this context is *in-network security* offering additional protection to the security mechanisms implemented on the devices. In particular, policy-based approaches restrict the allowed network traffic of each IoT device to the required connections for completing its functions [17], [14]. Hence, they leverage the fact that many IoT devices have a limited purpose with regular communication patterns, e. g., a sensor periodically reporting its measurements. The main idea thus follows the principles of *least privilege* and *defense in depth*, reducing the attack surface of a device and limiting the damage of infected devices. Recently, the Internet Engineering Task Force (IETF) proposed Manufacturer Usage Description (MUD) [9], which enables IoT devices to signal the network their required connections as communication rules. The standard defines a format for these rules and how a central network entity, e. g., the home router, can securely retrieve the rules of a connected device from a trusted source.

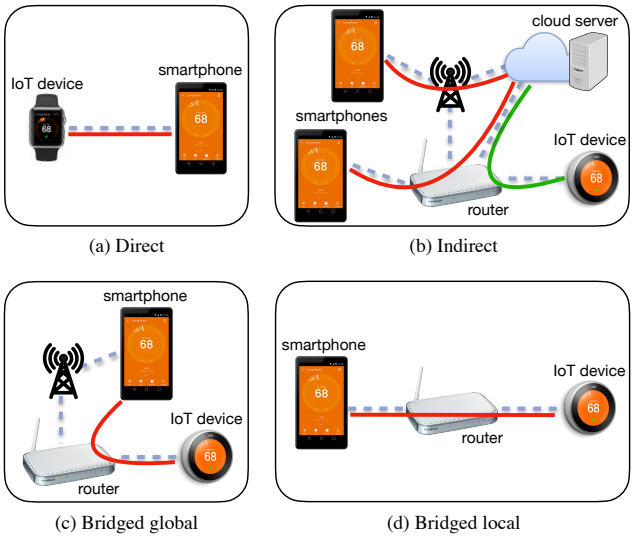

Fig. 1. Different communication scenarios between control apps (smartphones) and IoT devices. Due to dynamic IP addresses of smartphones and central enforcement, current policy-based approaches, such as MUD, only cover (b) indirect communication (green line), where the traffic flows through a cloud server. Other traffic to/from smartphones remains unfiltered (red lines).

Nevertheless, considering that users typically interact with IoT devices using a smartphone [7] questions the effectiveness of enforcing MUD rules only centrally at the router. Indeed, similar to [11] and shown in Fig. 1, we observe different scenarios for the communication between smartphone and IoT devices. Since MUD omits multi-purpose devices, like smartphones or tablet computers, it currently only covers *indirect communication* (cf. Fig. 1(b)), where the traffic flows through a cloud service. Moreover, the standard does not define *how* and *where* traffic enforcement should occur. Therefore, based on our findings in [6], we consider the different IoT communication scenarios and the pivotal role of smartphones, arguing that malicious traffic should be filtered close to its origin before spreading further in the home network.

Hence, we propose *Distributed Enforcement of MUD on Smartphones (DEMONS)* to efficiently enforce MUD rules for IoT devices in smart homes. We mitigate the risks emanating from smartphones and IoT devices by extending MUD to restrict access to the local network. Based on [6], we implement a central *Local MUD Manager (LMM)*, which we complement with *Mobile MUD Enforcement Engines (MMEEs)* running on different smartphones associated with the home network. The MMEEs enforce MUD rules close to the IoT devices, which reduces unwanted network traffic and even extends MUD enforcement to IoT devices directly connected to a smartphone. Our solution neither restricts the devices' intended functionality nor significantly influences latency, bandwidth, or power consumption. Our contributions are as follows:

1) We thoroughly summarize the problem space and the requirements for enforcing MUD in smart homes (Sec. II);
2) We propose a scalable framework for distributed MUD enforcement in complex smart home networks (Sec. III);
3) We provide a detailed evaluation regarding the security benefits and the performance of our prototype (Sec. IV).

II. Problem Analysis and Related Work

We analyze the current state of policy-based security for smart homes by summarizing the MUD standard and its limitations (Sec. II-A). Then, we generalize the considered attacker model (Sec. II-B) and present the related work (Sec. II-C).

A. MUD and Its Current Limitations

Manufacturer Usage Description (MUD) [9] is a recently proposed standard by the IETF specifying access control for smart devices. The key idea is that so-called *MUD files*, which contain Access Control Lists (ACLs), describe the accepted network connections for each IoT device. MUD files thus enable the narrow definition of the expected communication behavior of IoT devices since such devices usually have limited functionality and thus only require a small set of allowed connections. Moreover, MUD files can be transformed into rules that can be enforced at a central networking device to restrict access, prohibit undesired behavior from the outset, and serve as a reasonable basis for intrusion detection.

Today, MUD is well received and even supported by global players like Cisco and Google, acknowledging its potential for improving smart home security. However, the current standard does not appropriately cover all smart home devices, including smartphones and smart TVs, for two distinct reasons: (i) Devices typically use dynamic IP addresses. While MUD allows defining abstract device classes as endpoints, mapping non-IoT devices to such classes, especially given changing IP addresses, is not examined. Thus, currently, MUD files cannot conveniently define connections involving individual devices, whether deployed locally or in an external network. (ii) Even if defining such connections would succeed, the resulting rules would apply to entire devices. However, general-purpose devices provide various services, e. g., multiple apps on a smartphone. Hence, all these apps would gain access to the IoT devices instead of dedicated control apps only.

Consequently, the MUD standard does not consider the different interactions between IoT devices and others inside and outside the local network, which attackers can readily exploit as an entry point to a smart home. To better understand this weakness, we continue with our general attacker model.

B. Attacker Model

A widespread assumption for smart home security is that IoT devices are locally protected against unauthorized access from the Internet. However, this assumption does not reflect the current situation of smart homes, where attackers typically only compromise a single device to gain access to the network.

Therefore, we assume that the attacker successfully accessed the victim's home network, either by compromising one of her IoT devices or by infiltrating a malicious app on her smartphone. Then, following the Dolev-Yao attacker model [1], the attacker can overhear, intercept, and modify any message at the compromised device and perform passive and active attacks within the local network. In particular, the attacker would scan the network for vulnerable IoT devices and subsequently exploit these vulnerabilities to spread its malware further.

According to the Dolev-Yao model, the attacker is not capable of breaking state-of-the-art security methods. However, in contrast to Dolev-Yao, we assume that the attacker focuses on both legacy and state-of-the-art devices with known and unknown vulnerabilities, which is, unfortunately, a reasonable assumption for smart homes [3], [10], [7]. In the following, we give a short overview of related work before explaining how DEMONS counters this powerful attacker model in Sec. III.

C. Related Work

Our previous work [6] provided a comprehensive overview of related work, showing that existing in-network security solutions do not sufficiently consider the risks emanating from complex smart home networks. Consequently, we proposed a general solution concept to extend MUD to smartphones, which we use as a starting point for this paper. With DEMONS, in turn, we provide a concrete design and implementation of MMEE and LMM, including the details regarding their interaction and hedge against interference by third parties. We thoroughly evaluate the performance and security to determine its potential for smart home security.

Furthermore, we acknowledge that the attention for complex smart home communication and challenges for suitable security mechanisms is growing. Sikder et al. [15] extend access control to support multi-user multi-device scenarios in smart homes by restricting the communication of individual apps to specific IoT devices. However, their solution only targets apps running on a smart home platform by Samsung, neglecting smartphone apps. Similarly, Zhang et al. [20] propose a solution for traffic monitoring of smart home apps to detect malicious behavior, again targeting only apps running on Samsung's platform. While their work does not consider the risks emerging from smartphones, it still confirms our hypothesis that device-based access control is insufficient due to individual (malicious) apps running on the same device.

Afek et al. [2] deploy a MUD-based solution for protecting IoT devices on the ISP level. While their system only offers device-based access control, again omitting individual services such as smartphone apps, they recognize the problem of dynamic IP addresses (cf. Sec. II) and tackle it with tracking applications. These run, e. g., on smartphones and report metadata including the current IP address to a central instance, which then maps it to placeholders in MUD files.

III. Design

To close the gap left by existing policy-based approaches, we propose DEMONS, which effectively covers the different IoT communication scenarios. We describe our system's overall architecture before going into its details, identifying and addressing challenges left open by existing related work, and highlighting the security benefits of our architecture.

A. Architecture

Based on [6], we define two main components in DEMONS, a central *Local MUD Manager (LMM)* and *Mobile MUD Enforcement Engines (MMEEs)* running on different smartphones associated with the home network (cf. Fig. 2). To restrict the communication between IoT devices and devices to control them, the central LMM is deployed in the local home network on top of a Software-Defined Networking (SDN) controller ①. The LMM enforces MUD policies as the *MUD manager* described in [9] while benefiting from the SDN paradigm, which allows installing dynamic flow rules at SDN switches in the local network and thus fine-granular steering of the local network traffic. By running the SDN controller on the same device as, e. g., the local DHCP server, the SDN controller and thus the LMM's IP address is well-known to all devices connected to the local network. The IoT devices advertise themselves by transmitting MUD URLs as specified in [9], e. g., using DHCP. Consequently, the LMM is aware of all local IoT devices, including their MAC and IP addresses. Further, the LMM uses the MUD URLs to retrieve the individual MUD files of the present IoT devices.

The LMM cannot reliably distinguish traffic that emerges from different services running on the same device. Furthermore, the original MUD standard only considers connections between IoT devices and cloud services or specific groups of devices. However, defining connections to individual devices, such as smartphones, is not feasible because of their (usually) dynamic IP addresses. The consideration of individual services or apps running on a device is not included at all.

To cover the traffic from individual smartphones and their individual apps, distributed MMEEs complement the LMM ②. These MMEEs run directly on smartphones and filter the app traffic. Therefore, each MMEE maintains a flow table configured based on extended MUD files (cf. Sec. III-C), including the apps' legit communication behavior. The deployment of MMEEs on smartphones further helps the LMM to identify individual smartphones as valid control devices and to determine whether they should be allowed to communicate with IoT devices or not. To this end, the MMEE advertises

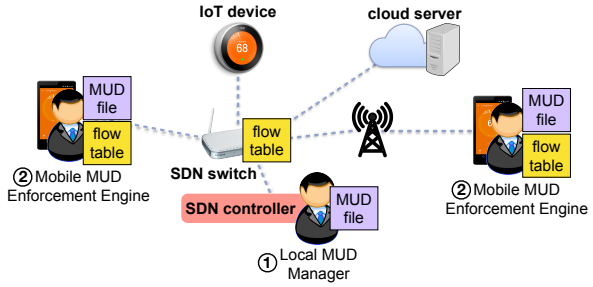

Fig. 2. DEMONS architecture [6]: ① a central LMM filters the traffic of all IoT devices in the network, enabling traffic to/from authorized smartphones and the Internet as defined in the IoT devices' MUD files. ② MMEEs on smartphones authenticate the smartphone to the LMM and filter app traffic.

the smartphone (and its current IP address) as a valid control device permitting only policy-complying communication of control apps running on the smartphone. Consequently, DEMONS only allows those apps to communicate with locally deployed IoT devices if the respective connections are defined in the MUD files on both the LMM and the MMEE. The LMM and the MMEE block all other local communication of smartphones (i. e., without a respective rule or MUD file), where Internet connections remain untouched to ensure non-IoT functionality. In particular, smartphones without MMEE experience unrestricted access to the Internet, while the LMM blocks all their local traffic to protect the smart home network.

Except for the abstract MUD manager, neither [9] nor [6] specify *how* to implement the enforcement within the local network. We thus propose a concrete design in the following.

B. Device-based Enforcement

After an IoT device announces its presence and transmits the URL of its MUD file to the LMM, a parser module checks the file's validity and authenticity as described in [9]. To identify authorized smartphones (running an MMEE), we integrate an *authentication manager* into LMM and MMEE. The authentication manager enables the MMEEs to establish a secure, encrypted connection and authenticate themselves to the LMM, e. g., using certificates and cryptographic protocols such as TLS. Then, they gain general local communication permission, which is revoked as soon as the connection between MMEE and LMM is interrupted to avoid take-over by third parties.

To state a *concrete* authorized smartphone within a MUD file, we leverage the `my-controller` node defined in the MUD standard. This node allows specifying a group of IP addresses as an endpoint within Access Control Entries (ACEs). Thus, we define abstract rules for communication with smartphones in the IoT devices' MUD files and enforce them by mapping the IP addresses of authorized smartphones to `my-controller` endpoints at run-time. Furthermore, we extend the head of the MUD files with a node to include control apps of the respective IoT device, e. g., app package names. This information is passed from the LMM to the MMEE to exclude all other apps from communication.

To enforce ACEs from MUD files, the LMM converts them into flow rules and installs these at appropriate networking

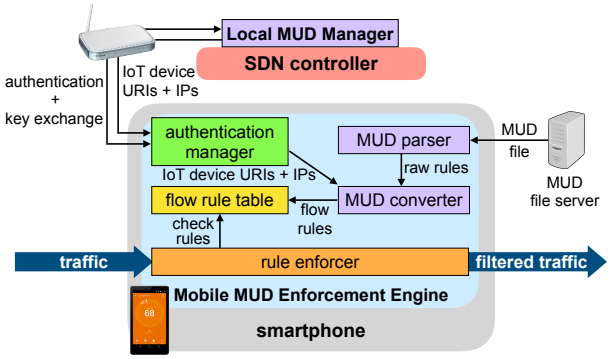

Fig. 3. MMEE architecture: the authentication manager performs authentication to the LMM, enabling local communication. The MUD parser retrieves MUD files of installed apps, extracts rules, and passes them to the MUD converter to create flow rules for filtering-based enforcement.

devices, e. g., SDN switches. Besides, there are multiple modes that we can implement to handle the traffic of IoT devices that have *no* MUD file available. One possibility is to generally drop all traffic associated with the local network and allow connections to and from the Internet only if defined in a global whitelist, e.g., containing official domains of device manufacturers. This policy provides a fair trade-off between security and functionality as undesired (and in particular local) access to the IoT devices is hindered while control via known cloud services is still possible (cf. Fig. 1(b)). Alternatively, in favor of functionality, the user might unlock single IoT devices to enable their full connectivity. Since different modes provide different trade-offs between security and function, we leave the final decision to the user to ensure usability. However, note that whenever a valid MUD file is available, the advertised functionality of the IoT device is not restricted by design.

C. App-based Enforcement

In Fig. 3, we depict the precise architecture of the MMEE. We envision the MMEE to be natively embedded into any smartphone's OS, encouraging use and hindering undetected modification as additionally protected by a chain of security mechanisms[1,2]. As an alternative, however, implementing the MMEE within the smartphone's user-space as a conventional app is also possible, easing deployment and distribution.

By design, the MMEE runs permanently in the background to enable uninterrupted traffic filtering, comparable to a host-based firewall. The rule enforcer module of the MMEE is responsible for monitoring and filtering traffic of all apps installed in the user space of the smartphone. To provide rules for individual apps, we extend the original MUD files [9] with sensible nodes. In the MUD file header, we include the app's package name and signing certificate that the MMEE validates against the respective installed app to prevent spoofing of MUD files. We further define connections to specific IoT devices within ACEs by adding a node `dev-uri` for stating device URIs. These URIs can be matched against broadcasts from IoT devices or device information forwarded from the LMM. To exclude faked or modified MUD files, the MMEE validates their signature according to the MUD standard. Shipping of the MUD URLs can be combined with the download of apps. Moreover, the MMEE regularly checks for updates of MUD files, also considering the `last-update` and `cache-validity` nodes provided by MUD.

If an LMM is available, the MMEE *authentication manager* establishes a secure connection to the LMM. If implemented in the OS, we can ship and anchor certificates with the OS, thus providing high integrity when, e. g., using TLS. In a user-space implementation, Android's Keystore system[3] is an example of the convenient storage of certificates. After successful authentication, the smartphone is enabled for local communication in compliance with MUD files of apps and IoT devices, ensured by both MMEE and LMM.

For apps without MUD file, we can restrict local communication without limiting non-IoT functionality. For instance, we can block all local traffic from apps without MUD files, while Internet traffic is not subject to filtering by default. Another possibility is to allow the user to unlock single apps, thus enabling their full functionality. As this option entails the risk of local attacks, a more viable approach is to handle missing MUD files of apps using category-based policies similar to [5].

D. Security Benefits and Limitations

Compared to existing solutions, our approach covers all communication scenarios identified in Sec. I by design. Besides indirect (cf. Fig. 1(b)) and bridged local (cf. Fig. 1(d)) communication, the MMEE even allows us to control direct communication between IoT devices and smartphone apps (cf. Fig. 1(a)), e. g., via Bluetooth. Our approach does not distinguish between smartphones of residents and visitors, enabling secure multi-user multi-device scenarios without additional effort. To allow for remote control offered by many IoT devices (cf. Fig. 1(c)), a third party on the Internet, e. g., a manufacturer cloud service, can mediate between MMEE and LMM. This cloud service then enables authentication and controlled access to IoT devices, even from remote networks.

To visualize the benefits of our design, we depict two example scenarios of a smart home with several IoT devices. In Fig. 4, we assume a MUD manager deployed at the local router, thus blocking traffic from malicious servers on the Internet. However, the traffic of smartphone apps remains untouched, allowing access from benign *and* malicious apps. In contrast, Fig. 5 shows how DEMONS addresses this vulnerability by applying fine-granular filtering of app traffic in all communication scenarios. Furthermore, having provided a valid MUD file, traffic of benign control apps and their advertised function is not restricted.

Even though our approach primarily aims to improve the security of control apps for IoT devices, we can apply it to all other apps to further reduce the attack surface and avoid connections to untrusted servers. Furthermore, the MMEE can be easily transferred to other complex IoT devices. For

[1]https://source.android.com/security/overview/kernel-security
[2]https://support.apple.com/de-de/guide/security/welcome/web
[3]https://developer.android.com/training/articles/keystore

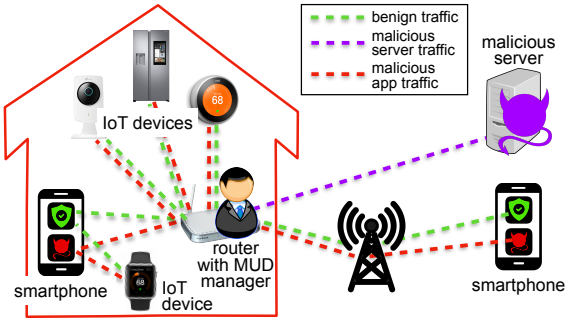

Fig. 4. Smart home with multiple IoT devices and a MUD manager [9] deployed at the router to allow traffic to/from IoT devices only according to their MUD files. While the router blocks traffic from external malicious servers, connections to smartphones are not covered including malicious apps.

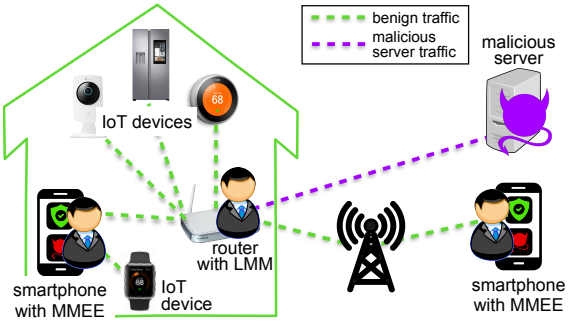

Fig. 5. Smart home with multiple IoT devices and DEMONS, i.e., an LMM deployed at the router and MMEEs deployed at smartphones. Traffic from external malicious servers is blocked as well as to/from malicious apps.

instance, smart TVs or smart fridges usually run the same OS as smartphones, e. g., Android TV or Tizen, allowing us to deploy the same (or slightly modified) implementation.

Despite its benefits, DEMONS also has its limitations. Our solution equally supports apps from IoT manufacturers and third-party apps. However, this still requires corresponding MUD files for apps and devices. Furthermore, original manufacturers might not even provide MUD files. While we see crowd-sourcing as a possible solution for crafting and validating MUD files for apps *and* devices, this idea needs further investigation. Furthermore, our SDN-based approach offers fine-granular traffic filtering, but it does not consider payload, which is also unrealistic for encrypted packets. Thus, elaborate attacks could still exploit authorized connections.

In conclusion, the LMM offers general protection to IoT devices, whereas the MMEE targets security vulnerabilities emerging from unrestricted use of smartphone apps to control IoT devices. Both components individually prevent single attacks, e. g., the MMEE increases security even in SDN-free environments. Finally, their combination leads to a truly comprehensive protection of smart homes. In the following, we thoroughly evaluate DEMONS, showing its plausibility.

IV. Evaluation

To prove its feasibility and effectiveness, we implement and evaluate DEMONS concerning both security and performance. Before diving into the details and results of our evaluation, we shortly describe our prototype and general set-up.

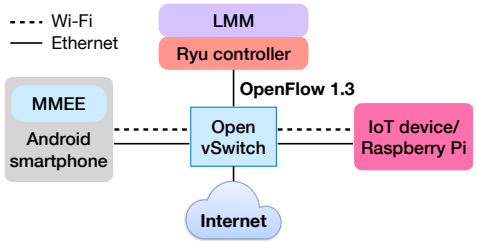

Fig. 6. Evaluation set-up of DEMONS, allowing both Ethernet and Wi-Fi connections between smartphone and IoT device / Raspberry Pi.

A. Implementation and General Set-up

We realize the LMM as an application running on top of the Python-based OpenFlow Controller `Ryu`[4] and connecting it to a virtual Open vSwitch using `Mininet`[5], connected to the Internet. It further uses Ethernet and Wi-Fi interfaces provided by the underlying hardware. We connect physical devices, including smartphones, IoT devices, and a Raspberry Pi, using these interfaces. Thus, our set-up for the evaluation follows the network topology depicted in Fig. 6.

While we generally envision the deployment of the MMEE as part of the smartphone's OS as described in Sec. III-C, we refrain from implementing our prototype within the OS due to the required root access. Instead, we opt for a user-space implementation as an Android app, extending the open-source firewall `NetGuard`[6]. This firewall uses Android's `VpnService`[7] and allows simple app-based filtering of outgoing traffic. In particular, we add components for the *authentication manager* and the *MUD parser* and *converter*, and we customize NetGuard's rule tables and methods for rule look-up (cf. Fig. 3). Android's container-based Keystore system[8] supports the protection of private keys used to authenticate the MMEEs to the LMM. Furthermore, if not embedded in the OS, we assume that MMEEs are only installed from official sources, e. g., Google Play Store, to restrict the deployment of modified implementations. In the following, we first evaluate the security-related functionality of DEMONS. Subsequently, we conduct a thorough performance evaluation.

B. Security Evaluation

A crucial aspect of our evaluation is to show the feasibility of DEMONS, i. e., improving smart home security without impeding the advertised functionality of apps and IoT devices. In the following, we show the effectiveness of DEMONS by analyzing its security improvement for existing smart home attacks. Then, we validate its correct functioning by comparing the network traffic with and without DEMONS.

1) Effectiveness: We evaluate DEMONS' effectiveness using existing smart home attacks identified by [8] and relying on our attacker model (cf. Sec. II-B). In a smart home deployment consisting of a Google Home speaker, a TP-Link smart light

[4]https://github.com/osrg/ryu
[5]https://github.com/mininet/mininet
[6]https://github.com/M66B/NetGuard
[7]https://developer.android.com/reference/android/net/VpnService
[8]https://developer.android.com/training/articles/keystore

TABLE I
SECURITY ANALYSIS OF DEMONS ACCORDING TO THE PRACTICAL SMART HOME ATTACKS IDENTIFIED BY [8]

Attack	Involved Devices	Attack Description	Mitigation
Case 1	Smartphone, Google Home	A malicious webserver pings all local IP addresses using `PingModel`. It sends `http` req. using `WebSocket` to obtain sensitive information from the Google Home.	✓ DEMONS would locally block pings and `http` requests if not explicitly whitelisted.
Case 2	Smartphone, Google Home, Google Home server	A malicious webserver sends `http`-requests with `WebSocket` to obtain the Google Home built-in certificate and `app_device_id`. It uses this information to access the user account on the Google Home server.	✓ DEMONS would block the `http`-request such that the attacker would neither gain the certificate nor the `app_device_id`.
Case 3	Google Home (& server), TP-Link server, TP-Link smart light	As a follow-up attack to attack case 2, the attacker can use the compromised user account to control other devices in the smart home, such as the smart light bulb.	∼ DEMONS cannot prevent the follow-up attack, occurring via Google Home server. However, it would prevent the previous user account attack (see attack case 2).
Case 4	Smartphone, TP-Link server, TP-Link smart light	An attacker steals the TP-LINK authorization token via a backup channel attack to gain full control of the smart light bulb.	✗ DEMONS would not prevent this attack, since the attacker exploits an Android backup vulnerability, outside the scope of DEMONS. The attacker then directly targets the TP-Link server to gain control of the smart light.
Case 5	Smartphone, TP-LINK smart light	A malicious app, which is installed on the user's smartphone uses UDP broadcast to control the smart light bulb.	✓ Since the malicious app would not be able to provide a valid MUD profile, DEMONS would block all local traffic originating from this app by default.
Case 6	Smartphone, TP-LINK server, TP-LINK smart light	A malicious app uses UDP broadcast to bind the smart light bulb to the attacker's account.	✓ Similar to the previous attack case, DEMONS would block all local traffic originating from the malicious app, since it does not have a valid MUD profile. Hence, the attacker could not take over the user's account.

bulb, and a smartphone, the authors identified six distinct attack cases covering a wide range of vulnerabilities. For our work, we are hence interested to what extent DEMONS mitigates the identified attack cases. Therefore, based on the results presented in [8], we analyze DEMONS' effectiveness for each attack case. Table I lists the corresponding results.

The results show that DEMONS effectively mitigates local attacks. By blocking unauthorized local broadcasts and connections, it prevents attack cases 1 and 2. Moreover, apps need a valid MUD profile to access the local network, and this profile limits the possible connections to the necessary minimum, which mitigates attack cases 5 and 6. Finally, DEMONS does not completely mitigate attack cases 3 and 4 since these attacks target the Google Home and TP-Link servers via the Internet, outside the scope of DEMONS. However, for attack case 3, DEMONS would prevent the attacker from gaining the Google Home built-in certificate and `app_device_id`, which are a prerequisite to compromise the user account.

To sum up, this case study shows that DEMONS mitigates existing vulnerabilities concerning typical devices in smart homes. It is hence a valuable extension to existing smart home security measures. However, it does not exempt cloud service providers from implementing adequate security mechanisms for further protection. Next, we validate whether DEMONS ensures the advertised functionality of IoT devices.

2) Validation: We deploy two IoT devices, a smart light bulb by LIFX[9] and a smart plug by TP-Link[10]. Their respective control apps run on a Motorola Moto G4 with Android 6.0. We connect the devices to the virtual switch via Wi-Fi. Then, we record the device traffic both in a traditional wireless network and in combination with the DEMONS prototype described in Sec. IV-A. For both IoT devices, we create simple MUD files, which we deploy at the LMM. To provide example rules, we define all local IPv4-based connections to and from control apps as allowed while all other traffic should be dropped. Moreover, we create MUD files for the control apps, permitting them only to communicate with their own IoT device. We deploy these MUD files at the MMEE.

First, with the IoT devices deployed in a local network without DEMONS, we can turn the devices on and off using their respective control apps using Wi-Fi and via cloud services. This behavior corresponds to their advertised description. Subsequently, we deploy DEMONS but do not yet provide MUD files to the MMEE but only to the LMM. Consequently, we cannot control the IoT devices since only apps with valid MUD files are allowed for communication with IoT devices. Hence the recorded traffic does not contain any packets sent from the smartphone to the IoT devices. Finally, after providing the apps' MUD files to the MMEE, we can control the IoT devices, but (as intended) only from the local network. As we do not define permitted connections to and from the Internet for IoT devices or apps, control via cloud services is still restricted. Accordingly, packets exchanged between external servers and smartphone or IoT devices are not contained in the recorded traffic. Most importantly, by deploying a `ping` app, we note that other apps can still not connect to the IoT devices, which is possible in the absence of DEMONS.

Overall, this validation shows that DEMONS behaves as intended and enables fine-granular traffic filtering between apps and IoT devices. Next, due to its importance for usability, we empirically evaluate the performance of DEMONS.

C. Performance Evaluation

We evaluate the performance of our approach regarding latency, bandwidth, and power consumption. Therefore, we connect a Raspberry Pi, representing an IoT device, and

[9]https://uk.lifx.com/products/lifx

[10]https://www.kasasmart.com/us/products/smart-plugs/kasa-smart-wifi-plug-hs100

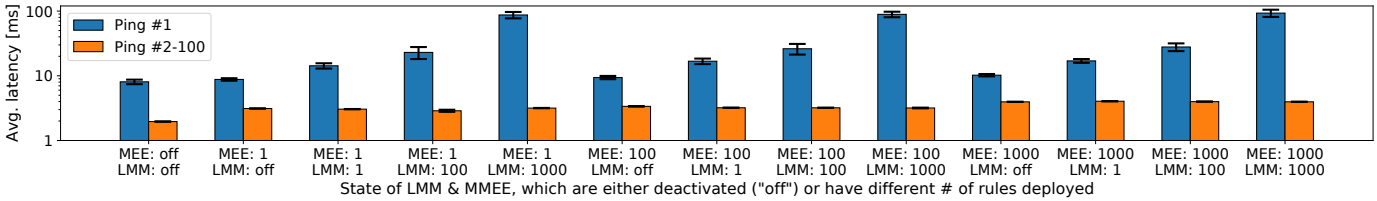

Fig. 7. Average latency with 95% confidential intervals measured for 30 runs with 100 ping messages, depending on the number of deployed rules. The first two bars represent the baseline with no traffic filtering. The latency of the first ping message is higher compared to the rest as this message is forwarded to the SDN controller before direct handling rules are installed at the switch. The latency increases with the number of rules due to higher look-up times.

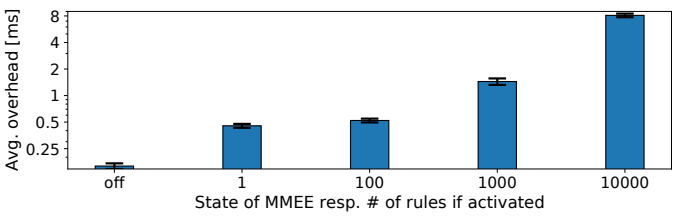

Fig. 8. Average overhead with 95% confidential intervals when matching against different numbers of rules at the MMEE, thus influencing the look-up time. The first bar represents the baseline with no traffic filtering at all.

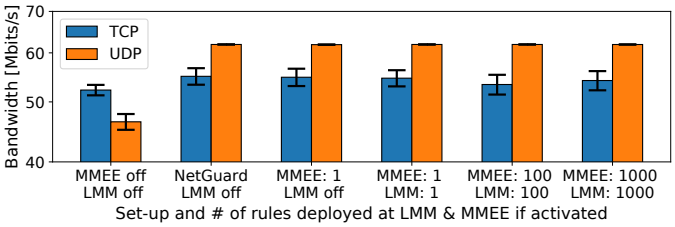

Fig. 9. Average bandwidth with 95% confidential intervals depending on the number of rules to match against and in comparison with NetGuard. The first two bars represent the baseline with no traffic filtering at all.

a Motorola Moto G4, running the MMEE in Android 6.0, to the virtual switch sequentially via Ethernet and Wi-Fi (cf. Fig. 6). Both LMM and MMEE need to match traffic against the rules derived from MUD files to decide how to handle it. The number of rules deployed at the LMM generally increases with the IoT devices in the network. Similarly, the number of rules at the MMEE increases with the installed apps. Concerning scalability, we thus conduct our latency and bandwidth evaluation depending on different, generously chosen numbers of non-matching fake rules installed at both components, simulating the presence of further devices and apps with MUD files. We create real MUD files for all actually deployed devices and apps, defining the evaluated connections as "allowed". All of our result graphs apply logarithmic scales, and the error bars represent the 95% confidential intervals.

1) Latency: Since increased latency can limit service functions, we first evaluate the latency overhead introduced by our implementations of LMM and MMEE when using Ethernet. We conducted 30 runs, during each of which we sent 100 `ping` messages from the smartphone to the IoT device (i. e., the Raspberry Pi) and calculated the average latency over all runs. Furthermore, we conducted this evaluation for different numbers of rules from none to 1000, against which both MMEE and LMM need to compare traffic to apply the filtering. We depict the corresponding results in Fig. 7.

In the figure, we see increased latencies for the first `ping` message compared to the subsequent 99 messages, even when filtering is neither executed at LMM nor MMEE. This salience is due to the use of SDN with OpenFlow, requiring that the first packet of a connection is always forwarded from the switch to the SDN controller to request flow treatment instructions. All subsequent packets are then handled directly by the switch, avoiding the initial overhead. While the latency expectedly increases depending on the number of rules, it does not exceed 4 ms for 1000 rules installed at both LMM and MMEE compared to about 2 ms when filtering is deactivated. Thus, the latency only noticeably increases for the first packet of a connection with a maximum latency of 100 ms for 1000 installed rules at LMM and MMEE. However, for the subsequent course of the connection, the latency is not significantly impaired.

To substantiate the described findings, we additionally measured the overhead introduced by different numbers of rules at the MMEE by comparing the timestamps at the beginning and the end of the filtering function. The results, shown in Fig. 8, confirm that even if traffic needs to be matched against 10000 rules, i. e., when a large number of IoT devices are deployed, this leads only to a latency overhead of about 8 ms compared to 2 ms for 1000 rules. We deem this an acceptable impact considering the application in smart homes.

2) Bandwidth: Lowered bandwidth has an impact both on functionality and user experience. Thus, we evaluate how the bandwidth is influenced by DEMONS using `iPerf3` via Ethernet and again regarding different numbers of installed rules to represent the scalability of our system. We conducted 50 runs, during which 1 GB of data was sent from the smartphone to the Raspberry Pi, respectively, using TCP and UDP. The corresponding results are depicted in Fig. 9.

We noticed that `NetGuard`, the basis for our MMEE, generally enables a higher bandwidth. We explain this by the fact that `NetGuard` applies, e. g., custom socket options like setting `TCP_NODELAY`. Consequently, we evaluate our system against the original implementation of `NetGuard` with deactivated traffic filtering. The results show that the traffic filtering introduced by DEMONS does not significantly decrease the throughput for both TCP and UDP, even with 1000 rules installed at LMM and MMEE.

3) Power: Another essential factor for the user experience is the power consumption of the MMEE, which runs on the user's smartphone. We use the mobile energy measurement

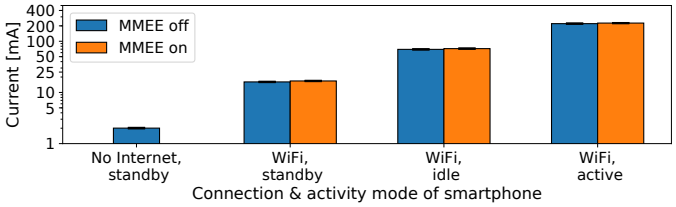

Fig. 10. Average smartphone current for different activity levels using Wi-Fi, each measured over 60 s. The first bar represents the baseline (disabled Internet, standby mode, no MMEE installed).

framework `PowerGraph`[11] to evaluate the energy consumption of the MMEE while considering different activity loads. We use a Wi-Fi connection and the activity loads comprise the following modes.

Standby: The screen is off; no actively running app.
Idle: The screen is on; no actively running app.
Active: The screen is on; `ping` messages are constantly sent with an interval of 200 ms.

We measured the drawn electric current for each activity for 90 s, leading to 713340 measuring points in total. We then calculated the average current after cutting off the first and last 15 s, respectively, to filter out transient phases. The evaluation results are displayed in Fig. 10.

Considering all test cases, we observe the smartphone's average current to be increased only by 5.72 mA at worst. Concerning a typical battery capacity of 3000 mA h (e. g., the Motorola Moto G4), our results indicate a run time of around 13.58 hours of the smartphone if running in the active mode and without having an MMEE installed. Hence, the deployment of the MMEE leads to a reduction of 0.4 hours (or 2.9%) in battery life, leaving a run time of 13.18 hours.

To sum up, our evaluation results show a slight increase in latency and power consumption when using our prototype. Since smart home services are, in general, not time-sensitive, we consider the performance impact as a reasonable trade-off regarding the provided security benefits (cf. Sec. III-D). The different numbers of rules deployed during evaluation indicate good scalability and usability of DEMONS, even in the presence of various IoT devices. However, since it would operate closer to the hardware, we expect even better performance for the implementation of the MMEE in the OS as envisioned in our design (cf. Sec. III-C).

V. Conclusion

In this paper, we aim to improve the security of smart homes by addressing the threats originating from malware-laden smartphones and legacy IoT devices. Therefore, we propose *DEMONS*, a distributed enforcement of MUD-based network policies that extends the centralized LMM with distributed MMEEs running directly on authenticated smartphones. Based on the functionality of each IoT device within the smart home and its necessary connections to other devices, we thus enable fine-granular filtering of local network traffic. Consequently, only authorized smartphone apps are allowed to communicate with IoT devices according to predefined MUD rules. We implemented and evaluated a proof of concept showing only a negligible impact on communication latency, bandwidth, and smartphone power consumption. Furthermore, our security evaluation shows that our approach effectively mitigates existing vulnerabilities of typical IoT devices in smart homes without limiting the intended functionality.

Although this study focuses on smartphones used for interaction with IoT devices, the design of DEMONS is, in general, applicable to all IoT devices not covered by MUD, such as smart TVs and smart speakers. Distributed enforcement thus opens new possibilities for scalable IoT security by blocking malicious network traffic close to its origin. Hence, for future work, we propose to evaluate DEMONS in complex smart home deployments with multiple MMEEs. Further, we want to contribute our extensions to the current IETF MUD standard.

[11] https://www.comsys.rwth-aachen.de/fileadmin/misc/2015/powergraph/

Acknowledgments

This research was supported by the research training group "Human Centered System Security" sponsored by the German federal state of North Rhine-Westphalia.

References

[1] S. Adepu and A. Mathur, "Generalized Attacker and Attack Models for Cyber Physical Systems," in *COMPSAC*. IEEE, Jun. 2016.
[2] Y. Afek, A. Bremler-Barr *et al.*, "Nfv-based iot security for home networks using mud," in *NOMS*. IEEE, April 2020.
[3] J. Bugeja, A. Jacobsson, and P. Davidsson, "On Privacy and Security Challenges in Smart Connected Homes," in *EISIC*. IEEE, 2016.
[4] M. Capellupo, J. Liranzo *et al.*, "Security and Attack Vector Analysis of IoT Devices," in *SpaCCS*. Springer Int'l Pub., 2017.
[5] S. Demetriou *et al.*, "HanGuard: SDN-driven Protection of Smart Home WiFi Devices from Malicious Mobile Apps," in *WiSec*. ACM, 2017.
[6] I. B. Fink, M. Serror, and K. Wehrle, "Extending MUD to Smartphones," in *LCN*. IEEE, Nov. 2020.
[7] D. Geneiatakis, I. Kounelis *et al.*, "Security and Privacy Issues for an IoT based Smart Home," in *MIPRO*. IEEE, 2017.
[8] Y. Jia *et al.*, "A Novel Graph-based Mechanism for Identifying Traffic Vulnerabilities in Smart Home IoT," in *INFOCOM*. IEEE, Apr. 2018.
[9] E. Lear, R. Droms, and D. Romascanu, "Manufacturer Usage Description Specification," RFC 8520, Mar. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8520.txt
[10] F. Loi, A. Sivanathan *et al.*, "Systematically Evaluating Security and Privacy for Consumer IoT Devices," in *IoT S&P*. ACM, Nov. 2017.
[11] S. Notra *et al.*, "An Experimental Study of Security and Privacy Risks with Emerging Household Appliances," in *CNS*. IEEE, Oct. 2014.
[12] M. Patton, E. Gross *et al.*, "Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things," in *JISIC*. IEEE, Sep. 2014.
[13] S. S. I. Samuel, "A Review of Connectivity Challenges in IoT-Smart Home," in *ICBDSC*. IEEE, Mar. 2016.
[14] M. Serror, M. Henze *et al.*, "Towards In-Network Security for Smart Homes," in *ARES*. ACM, Aug. 2018.
[15] A. K. Sikder, L. Babun *et al.*, "Kratos: Multi-User Multi-Device-Aware Access Control System for the Smart Home," in *WiSec*. ACM, 2020.
[16] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-Phones Attacking Smart-Homes," in *WiSec*. ACM, 2016.
[17] V. Sivaraman *et al.*, "Network-Level Security and Privacy Control for Smart-Home IoT Devices," in *WiMob*. IEEE, Oct. 2015.
[18] J. Wurm, K. Hoang *et al.*, "Security Analysis on Consumer and Industrial IoT Devices," in *ASP-DAC*. ACM, Jan. 2016.
[19] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things," in *HotNets*. ACM, Nov. 2015.
[20] W. Zhang, Y. Meng *et al.*, "HoMonit: Monitoring Smart Home Apps from Encrypted Traffic," in *CCS*. ACM, 2018.