Final Dissertation

Submitted to obtain the degree of Master of Science in Cybersecurity

---

Stepping out of the MUD:

contextual network threat information for IoT devices with manufacturer-provided behavioural profiles

---

Student
Luca Morgese
s2576120
l.morgese@student.utwente.nl
lucamorgese@outlook.com

University of Twente Supervisors
Dr. Ir. Andrea Continella
Ir. Thijs van Ede

External Supervisor
Ir. Tim Booij, TNO

Additional Committee Members
Prof. Dr. Ir. Roland van Rijswijk, University of Twente

UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente
The Netherlands

December 13, 2021

Contents

Abstract

Besides the unprecedented benefits that the Internet of Things (IoT) brings, it comes with a lack of adequate security measures, leading to attacks multiplying every year. Enhancing collection and sharing of actionable IoT threats intelligence is a crucial step to counter these trends. Currently, IoT threat intelligence is costly to obtain, especially at scale. Thus, in this work, we propose a novel approach to produce near real-time and fine-grained information on IoT network threats, from real-world vantage points. We use the Manufacturer Usage Description specification to collect necessarily-anomalous IoT traffic from multiple deployments, and obtain an open window on malicious traffic targeting IoT at scale. We implement and validate our approach on two IoT deployments. We show that we can detect when devices from different deployments are being synchronously targeted by similar or different attack patterns. We demonstrate that we can obtain a heatmap view of IoT network anomalies, related to specific devices and deployments.

1 Introduction

The Internet of Things (IoT) paradigm refers to a cyber-physical ecosystem of interconnected devices (things), which exchange and process data to enable intelligent decision-making [1]. IoT increases control, efficiency, and automation of tasks and is already present in many different domains, such as healthcare, logistics, industry, and smart public environments [2,3]. In effect, some latest figures on the rate of adoption of IoT devices underpin its imposing growth: it is expected that the number of Internet-connected devices will quadruple from 9 billion in 2020 to 38 billion in 2030 [4].

Unfortunately, while it comes with benefits, the continuous integration of IoT corresponds to an increasing attack surface. Recent works reveal an average 240% growth of IoT-targeting ransomware from 2015 to 2016 [5], and discuss its catastrophic potential as they may target critical sectors, such as smart energy grids [6,7]. More recently, IBM X-force reports that, in 2020 only, the number of IoT-directed attacks increased by 800% compared to the previous year [8] — mostly driven by a proliferation of post-Mirai botnets (e.g., `Mozi`, `OKIRU`, `BrickerBot`, and `Persirai`).[1] In 2021, research showed that newly deployed IoT devices are attacked within the first five minutes [9], and 1.5 billion attacks on smart devices were recorded in the first half of the year, more than twice with respect to the same period in 2020 [10].

Against this thriving threat landscape, both research and industry stress the need to enhance collection and sharing of actionable IoT security information [11–15]. More information about attackers' intents, interests, and trends in IoT allows faster, more precise, and better-informed security interventions on both network administrators and device vendors' sides [12, 16–18], making IoT integration and use safer.

1.1 Limitations in the state-of-art

IoT honeypots, network-telescopes, and threat intelligence (TI) feeds are three main state-of-art approaches to collect and share security information. Each, though, has limitations in providing accurate, timely, and at-scale security information.

First, IoT-specific honeypots capture how attackers' behaviour evolves over time [19–21], and produce deployment-specific security information. Though, scaling them geographically imposes relevant challenges. The choice of the deployment space (e.g., universities, cloud providers, or private addresses) and geographical position introduces concerns of costs, attractiveness to attackers, and risks of being fingerprinted as decoy systems [19, 22, 23]. Indeed, a 2021 survey from Franco et al. [22] exposes a widespread lack of anti-detection measures in IoT honeypots.

Second, there is an emerging trend in analysing network telescopes traffic to collect IoT-related information. Network telescopes (or, 'darknets') are portions of routable IP addresses that do not host any service: all traffic that they receive is thus necessarily anomalous [24]. This approach produces previously unachieved insights on at-scale IoT malicious activities [12,25–27]. Though, network telescope inferences are biased towards internet-wide activities, due to their undiscriminating and passive nature: they cannot capture attacks targeted to specific deployments, and may thus underestimate target-specific exploitation attempts [18].

Third, platforms such as AlienVaultOTX, Censys, and Shodan[2] provide open-access and crowd-sourced

[1] Mozi: https://securityintelligence.com/posts/botnet-attack-mozi-mozied-into-town/; OKIRU: https://research.checkpoint.com/2017/good-zero-day-skiddie/; Brickerbot: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/brickerbot-malware-permanently-bricks-iot-devices; Persirai: https://www.trendmicro.com/en_us/research/17/e/persirai-new-internet-things-iot-botnet-targets-ip-cameras.html.

[2] AlienVault: https://cybersecurity.att.com/products/ossim; Censys: https://censys.io; Shodan: https://www.shodan.io.

IoT TI at scale [28]. They arguably provide defenders with valuable resources. Though, they may suffer issues in balancing timeliness and accuracy (with respect to identifying attacks taking place), and coverage of captured threats (as dependent on the varying contributors' capability to capture them), of the security information they publish. Integrating threat intelligence feeds in security solutions may thus even risk generating harm [27, 29, 30].

1.2 Problem Statement

1.2.1 Motivating example

The 2017 OKIRU/SATORI botnet [31] targeted Huawei HG532 home routers with a 0-day remote code execution vulnerability (CVE-2017-17215).[3] The infected router would then be used to perform distributed DoS via TCP and UDP flooding. Vulnerability and attack were identified by CheckPoint Research, thanks to their arsenal of distributed sensors and honeypots, at service of their clients — in this case, Huawei. By the information in their reports, the attack would blindly target any host it reached, disregarding its OS, which thus facilitated its detection.[4]

It is not hard to think of possible similar attack campaigns that could be motivated by more specific interests (rather than largely distribute a botnet), and better designed, such as a threat agent disrupting financial advisors' home offices, or eavesdropping cafe networks near institutional buildings. A honeypot may produce actionable information in similar scenarios only if it simulates the respective network environments, hosts the interested devices, and is not known as a decoy. Because these attacks would be target-specific, a network telescope could not detect these scenarios altogether (as it only senses 'noise' internet-wide traffic). A threat intelligence feed can be leveraged only if attacks were previously detected, adequately reported, and their feed is consulted by the specific deployments.

1.2.2 Challenge

With the above considerations, we aim at proposing an approach to collect IoT-related anomalous network events information with the following characteristics.

1. **Specific**. Current state-of-art solutions struggle in identifying directly what the target of anomalous IoT network events is, on a large-scale picture. We want to directly identify the specific targets that such events are reaching: specific device models or vendors, geographical locations, deployment types (e.g., office, home, restoration).
2. **Timely**. With the above considerations on the state-of-art, we understand that providing timely availability of specific security information, as soon as a security event takes place, represents a central point of concern. This is especially true from an at-scale perspective.
3. **Consistent at scale**. Due to varying implementation details and maintenance requirements, any current state-of-art approach struggles in producing consistent IoT network anomalies information at-scale. A light-weight, consistent, and easily deployable solution, to this end, is missing.

We argue that no state-of-art approach thoroughly satisfies all these requirements at the same time. The challenge we identify is proposing an approach that does so. We describe this endeavour as aimed to achieve *contextual threat awareness*, for occurring anomalous IoT network events: knowing in potential real-time if anomalous activities are targeting IoT, at-scale, and what IoT devices these activities specifically reach.

1.3 Goal

This work presents a novel approach to monitor the distribution of malicious IoT-related traffic for different deployments over time. The core idea is that comparing variations in anomalous traffic throughout IoT deployments allows us to infer, for example, whether an IoT threat is targeting: all airports in a specific country; all IP cameras indiscriminately; or home offices of government personnel.

We base our proposal on a newly introduced standard: Manufacturer Usage Description (MUD) profiles [32]. These profiles are manufacturer-provided configuration files, uniquely associated with an IoT device model. A MUD profile specifies the strictly necessary network communication rules for a device to function, thus offering a white-list in this respect. This white-list can then be enforced at local IoT deployments. All traffic related to a device that is not explicitly allowed by a MUD profile is necessarily

[3] https://nvd.nist.gov/vuln/detail/CVE-2017-17215

[4] No host reconnaissance preceded the attack vector being sent, and the attack payload included the `HUAWEIUPNP` keyword. Besides, the research group commented on the low sophistication level of the threat. Reported at https://threatpost.com/huawei-router-vulnerability-used-to-spread-mirai-variant/129238/.

anomalous, which we classify as MUD-rejected traffic (MRT).

We propose to monitor feeds of rejected traffic yielded by multiple MUD profiles (for related devices) at different deployments, associating to each feed the respective deployment metadata. This way, we can investigate rejected traffic across these feeds, which may indicate coordinated or targeted malicious phenomena — based on geographical location, deployment sector, device model, and vendor. After detecting correlated or emerging events, we can generate alerts, and trace back the source of the anomalies with local network logs. Our approach is capable of yielding a near-real-time monitor on how anomalous activities are targeting specific IoT deployments — contributing at large to the benefit of contextual IoT threat awareness.

In this work, we propose and implement a pipeline to achieve this. We test our approach in a laboratory setting where we target IoT devices at two separate environments. We subject multiple devices to synchronous anomalous traffic, filter the traffic of all devices according to their MUD rules, and produce a description of how their rejected traffic varies. Analysing these descriptions, we show that we can detect when any two identified devices are being targeted by same or different anomalous network events, through a given time span.

For the sake of open science, and to promote further research on this approach, we publish the source code of our implementation at https://github.com/lucamrgs/MUDscope.

1.4 Contributions

Our work brings the following contributions:

- We provide security operators with a new lightweight and capillary vantage point for monitoring IoT-related anomalous traffic. This vantage point senses an up-front picture of necessarily-anomalous traffic related to IoT devices.
- We present a novel approach for monitoring IoT-related malicious events at scale using MUD profiles. We propose to cluster the traffic rejected by each profile in a specific deployment. By comparing MUD-rejected clusters of multiple deployments, we can contextualize what targets are reached by network threats.
- We demonstrate that the approach we propose is able to detect when multiple devices, from virtually any arbitrary deployment, are subject to synchronous similar or different anomalous activities.
- Our method advances the case for specification-based anomaly detection in IoT security, showcasing possibilities of coordination and security information sharing across defenders.

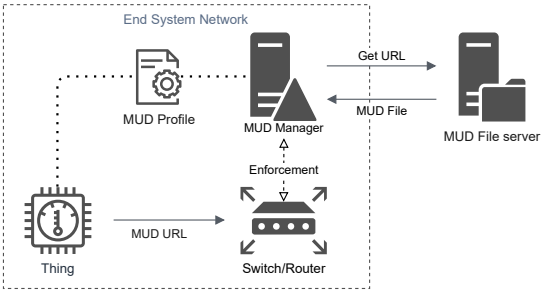

Figure 1: MUD Architecture, taken and adapted from RFC 8520 [32].

2 Background

2.1 Manufacturer Usage Description (MUD)

The Manufacturer Usage Description (MUD) specification [32] instructs IoT manufacturers to define *MUD profiles*. A MUD profile white-lists the intended and minimal connection requirements — such as the set of network ends (IP addresses and ports) and related modes of communication (protocols, connection parameters, initiating side) — that a device needs. A local network can use these information to author a "context-specific access policy, so the device functions only within" the connection rules that the MUD profile includes [33]. The specification is a work in progress, and it is expected to be progressively enriched beyond its initial network access control list–based nature.

The core objectives of MUD are to (i) reduce the threat surface on a device to those communications intended by the manufacturer; (ii) provide a flexible way to scale management of network policies for an increasing number of types of devices in a network; (iii) keep implementation costs of such measures at a bare minimum. There are two caveats: MUD is designed for IoT devices for which it is possible to precisely describe the expected network behaviour (which maximises the limitation of the attack surface); and it primarily aims at defending the devices from threats, rather than protecting a local network from IoT devices that have been compromised.

The MUD framework (Figure 1) has three architectural elements: an *URL*, used to locate a profile; the *profile* itself; and protocols for local network management systems to retrieve the profile. A *manager* module receives the URL from a device, and fetches the up-to-date profile from a vendor-side *file server*, verifying the profile authenticity. An *enforcer* module then implements the profile rules as actionable policies, configuring the routing capability that the device interfaces with (e.g., a SDN switch,[5] or a router). In its current definition, the *profile* is a JSON (JavaScript Object Notation)[6] file serializing network configuration parameters[7] that include (refer to §2–4, 8, in RFC 8520 [32]): meta-information on the device and the profile itself, network access control lists divided from-device and to-device, implementing (at least) matches on IPv4, IPv6, TCP, UDP and ICMP; DNS matches to resolve domain names, specified in the rules, to IP addresses. The sample of a MUD profile provided by the RFC is reported in Appendix A.1.

2.2 Specification-based intrusion detection in IoT

A Network Intrusion Detection System (NIDS) monitors network activity to detect malicious traffic. Traditionally, three main methods of intrusion detection exist: signature-, anomaly-, or specification-based. In signature-based detection, traffic and programs behaviours are compared against known malicious patterns or strings to detect anomalous events [34]. Anomaly-based detection generates a behaviour profile by observing a system under normal (benign) circumstances. Once such profile is obtained, anomalies are detected as deviations from that profile. Specification-based detection works similarly to anomaly-based detection, but uses predefined profiles describing known *benign* behaviour.

Recent works investigate NIDS approaches in the IoT, and find that specification-based anomaly detection is particularly suited for detecting malicious activities its domain. The foremost reason being that most IoT devices perform very limited actions, which can be easily described and modelled [35–38]. In this work, we rely on MUD as a specification-based anomaly detection tool.

[5] A network switch implementing the Software-Defined Networking framework, https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html.

[6] https://www.json.org/json-en.html

[7] Modelled in YANG, a standard language to describe network configurations, defined in RFC 6020, https://datatracker.ietf.org/doc/html/rfc6020.

3 Threat Model

3.1 Assumptions and scope

MUD Framework. As the adoption of MUD is still at early stages, we assume that a manufacturer-provided MUD profile exists for deployed devices. In this work, we autonomously generate the profiles using the *MUDgee* open-source tool by Hamza et al. [39]. Besides, we assume that local deployments have means to retrieve and validate updated MUD profiles, thus preventing attackers from injecting fake profiles in a network (this is already partly possible thanks to GlobalPlatform's MUD file server).[8]

Network environment. In this work, we scope the use-case of our approach to Small Office, Home Office (SOHO) -like IoT network environments. We do this because of the similarly limited security capabilities that they enforce, which in turn make these environments easier to reach by network threats [40] (such as spreading botnets).

Devices. While, in theory, our approach should work for any network protocol included in a MUD profile, we limit this research to IoT devices communicating over the UDP-TCP/IP stack as the majority of devices communicate this way.[9] Moreover, we scope our approach to single-purpose IoT devices, as they represent the primary use-case of the MUD specification.

MUD scope. We clarify that, with our proposal, we *do not* aim to block MUD-disallowed traffic. In the scope of this work, we use MUD as a tool to passively monitor IoT-related anomalous activity.

3.2 Threat agents

MIRAI-derived botnets represent a majour threat to IoT [8], and most of them spread by identifying victims through scanning techniques [41]. These malware, or the threat actors behind them, can implement different levels of specialization in the chosen techniques and targets, to cause specific types of disruption [6, 7]. With this framing, the threat agents that we assume in this work correspond to internet-originated and IoT-targeting scanning activities, and Denial of Service (DoS) attacks, as attempts to disrupt IoT capabilities in a local deployment. On the MITRE ATT&CK framework,[10] these map to *active scanning* in the reconnaissance phase, and *network denial of service* in the impact phase. As these attacks are observed as highly common among IoT de-

[8] https://mudfileservice.globalplatform.org/mud-files-database.php

[9] Reference from W3.org: https://www.w3.org/People/Frystyk/thesis/TcpIp.html.

[10] https://attack.mitre.org

Traffic scenario	(→ = executes) Target device	Attack category	# packets c. Total	Attack
benign	both	None	137k	-
dos-synflood	EZVIZ	SYN Flooding	106k	48k
dos-synflood	NUGU	SYN Flooding	35k	17k
scan-hostport	EZVIZ	Port Scanning	80k	5k
scan-hostport	NUGU	Port Scanning	19k	6k
scan-portos	EZVIZ	OS Detection	186k	4k
scan-portos	NUGU	OS Detection	24k	8k
mirai-udpflood	EZVIZ→	UDP Flooding	592k	475k
mirai-udpflood	NUGU→	UDP Flooding	592k	475k
mirai-ackflood	EZVIZ→	ACK Flooding	156k	38k
mirai-ackflood	NUGU→	ACK Flooding	156k	38k
mirai-httpflood	EZVIZ→	HTTP Flooding	124k	5k
mirai-httpflood	NUGU→	HTTP Flooding	124k	5k
mirai-bruteforce	EZVIZ→NUGU	Telnet Bruteforce	273k	1.5k
mirai-bruteforce	NUGU→EZVIZ	Telnet Bruteforce	180k	1k

Table 1: Description of the used portion of the *IoT Network Intrusion Dataset*, by Kang et al. [42]. Each traffic scenario is a traffic capture file containing related activities. The target device is either victim or executor (→) of the attack.

Ezviz Attack scenario	Role	# of packets All	MRT	Ground truth	Ground truth in MRT	% MUD detected
mirai-ackflood	atk	156k	96377	37816	37816	100
mirai-httpflood	atk	124k	63793	5232	5232	100
scan-portos	vict	186k	9944	4981	4463	89.6
mirai-brutefrc-atk	atk	273k	36435	1636	1187	72.56
mirai-brutefrc-vict	vict	179k	13087	961	737	76.69
mirai-udpflood	atk	593k	533203	474642	474642	100
scan-hostport	vict	80k	8200	5101	4659	91.34
dos-synflood	vict	106k	99320	48103	48095	99.98
				Average		91.27

Table 2: Effectiveness of MUD rules in detecting malicious packets. Tested for the *Ezviz* scenarios, in Kang et al. dataset. For all specified attack scenarios, MUD filters in average 91.27% of malicious packets.

vices (§VIII.K in franco et al. [22]), we choose them to validate our work, noting that our approach naturally extends to any attack coming from any source that is not white-listed by a MUD profile.

4 Preliminary analysis

4.1 IoT network intrusion dataset

We study the feasibility of our approach on the *IoT Network Intrusion Dataset* by Kang et al. [42]. This dataset allows us to test the effectiveness of MUD rules in various attack scenarios, which fit our threat model.

The dataset contains traffic captures (pcaps) from a real network test-bed hosting two smart-home IoT devices (an *SKT Nugu (NU 100)* smart speaker and an *EZVIZ (C2C Mini O Plus 1080P)* wi-fi camera). The devices are subject to the same attack scenarios, at separate times. Among the included attacks are DoS, reconnaissance scanning, and MIRAI DoS and bruteforcing. The dataset also includes a *benign* traffic capture. We note that the dataset also contains man-in-the-middle ARP spoofing attacks, which are out of our scope according to our defined threat model, and we thus do not consider them. A description of the dataset is shown in Table 1.

4.2 MUD filtering

The proposal of using MUD as a vantage point to monitor IoT-related anomalous activity relies necessarily on its effectiveness in filtering anomalous traffic. We perform a preliminary analysis to verify this: we use Kang's dataset and generate an MUD profile for one device, using MUDgee [39]. We filter the attack traffic to and from the device through its MUD rules. We finally verify what percentage and types of malicious traffic are blocked.

4.2.1 MUD Profiles acquisition

To generate an MUD profile, MUDgee takes as input a *pcap* file (a network traffic capture) containing *only benign* traffic, and outputs the profile related to the user-specified device. The tool also outputs an OpenFlow table [43], listing forwarding rules for the allowed UDP/TCP flows associated to the MUD rules.

OpenFlow is a widely adopted protocol to control packets forwarding in switches in a software-defined (i.e., programmable) network;[11] a framework often coupled with IoT networks [44]. The OpenFlow rules output by MUDgee thus synthesize and enforce the related MUD profile.

4.2.2 Validation of MUD attacks prevention

We thus use Kang's dataset to test the effectiveness of a MUD profile to intercept malicious traffic interesting a device. We generate a MUD profile for the *Ezviz* device with *MUDgee* [39], and related OpenFlow rules, using the benign capture in the dataset. For each Ezviz-related attack scenario pcap, we filter MUD-rejected traffic by matching each packet against the MUD OpenFlow rules. We output the non-abiding packets to a scenario-specific MUD-rejected traffic (MRT) pcap file. Then, we use the dataset-provided ground truth to filter malicious packets from these MRT pcaps. We are thus able to assess what percentage of the malicious activities a MUD profile can successfully capture.

Table 2 shows the results of this process. We see large differences in MUD-rejected and actual ground truth `mirai-httpflood` and `mirai-butefrc` scenarios. We

[11] https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html

verified this is due to the local traffic that is not included in MUD rules, and thus does not undermine the approach. Where the MRT–ground truth difference is about double, we verified this simply happens because we filter bi-directional flows, whereas the dataset's ground truth only covers flows originated from the malign source (which are mono-directional). Notably, over the represented scenarios, MUD captures in average 91.27% of malicious packets.

Overall, these preliminary results show that a MUD profile can effectively capture a range of anomalous traffic. This notion provides a baseline to support the steps we take in developing our approach.

4.3 Attack-discriminating flow features

From MUD-rejected traffic, we want to identify, for instance, if an IoT device was targeted by DoS, scan, or brute-forcing attacks. This would contribute to an improved understanding of what anomalous activities are targeting IoT, to the benefit of contextual awareness, as we argue in §1.2.2. We conduct a preliminary analysis on what features in the rejected traffic allow us to discriminate anomalous events.

4.3.1 MUD-rejected flows custom dataset

In computer networking, a *flow* describes an exchange of packets between two hosts, to carry a logical connection [45], underlying an operation. Expressing MUD-rejected traffic in terms of flows is, therefore, a first abstraction to better understand the anomalous operations (i.e., attacks, 'events') an IoT device engages with.

We derive a flow-based dataset from Kang's, consisting of the CSV (comma-separated values) MRT flows file obtained from *all* attack scenarios concerning the Ezviz camera (we explain how we obtain MUD-rejected flows, and related features, in §5.2 and §5.3.1). We label each flow with the attack scenario they belong to. A flow is labelled *unknown* if it was filtered, but does not belong to a specific attack according to Kang's ground truth.[12] An overview of the obtained dataset is shown in Table 3.

4.3.2 Selection of flow features

We want to select the flow features that are most representative of the attack types in the dataset. To this end, we compute the Adjusted Mutual Information (AMI) index[13] of each feature with respect to the 'attack type' label of our derived dataset. We choose AMI because it is a robust metric in the presence of large classes, as is the case in our dataset (for *DoS* and *unknown* classes).

Among two variables, AMI measures how the entropy of one variable is reduced once the value of other variable is known. It first computes the mutual information $MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log(\frac{p(x,y)}{p(x)p(y)})$, where here x and y are the values of the feature and the class respectively. Then, it adjusts the value by normalizing it on the expected value for MI, and the average over the entropy of X and Y: $AMI(X,Y) = \frac{MI(X,Y) - E(MI(X,Y))}{(avg(H(X),H(Y)) - E(MI(X,Y))}$.

Attack label	# Entries	%
dos-synflood	44547	63.186
unknown	10413	14.762
mirai-ackflood	7659	10.851
scan	5815	8.243
mirai-httpflood	1737	2.466
mirai-udpflood	223	0.003
mirai-brutefrc-atk	94	0.001
mirai-brutefrc-vict	59	~0.001
Total	**70547**	-

Table 3: Layout of custom MRT flows dataset derived from Ezviz attack scenarios in Kang et al. [42]

Feature	Description	AMI
bpp	bytes per packet	0.630
flgs_int	int value of flags bits array	0.585
da	destination address	0.518
obyt	output bytes	0.468
ibyt	input bytes	0.456
dp	destination port	0.414
opkt	output packets	0.456
sa	source address	0.387
ipkt	input packets	0.307

Table 4: Selected features and their AMI scores with respect to the attack label.

4.3.3 Results of features selection

Our MRT flows dataset contains a selection of attack scenarios that would not be reasonably found in the MUD-rejected traffic from a short time window. Therefore, to obtain an estimate on what flow features would be most discriminating in more representative attack scenarios, we test the AMI performance of each feature on several *subsets* of our dataset. We take all combinations of three different labels, and

[12] As we observed in §4.2.2, most of the *unknown* traffic corresponds to local ARP messages.

[13] https://en.wikipedia.org/wiki/Adjusted_mutual_information

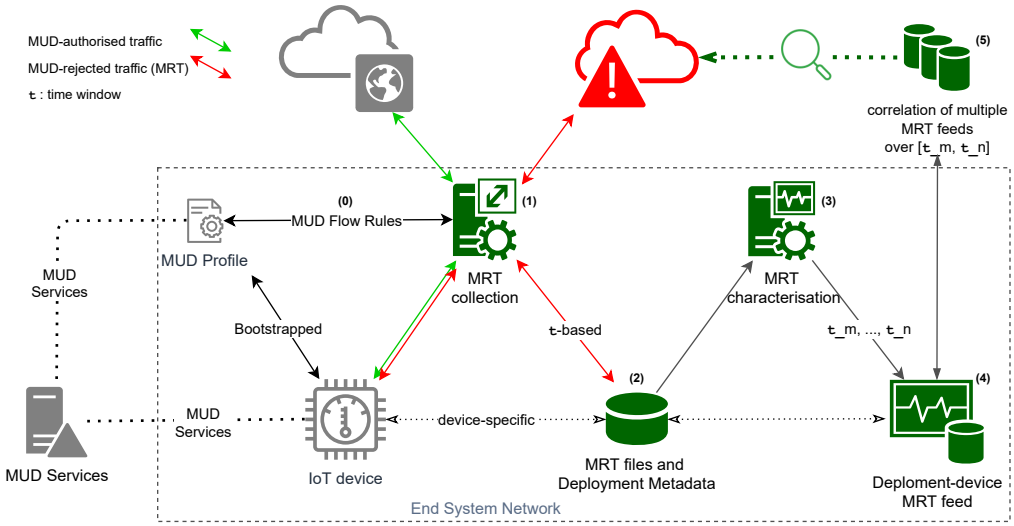

Figure 2: High-level architecture for the proposed MUD usage. (0) The local environment retrieves MUD profiles and flow rules; (1) we enforce MUD rules on device-related traffic; (2) we save the rejected traffic with associated metadata; (3) we describe how rejected traffic evolves through time; (4) we save the description in an MRT feed; (5) we compare multiple MRT feeds to observe how rejected traffic fluctuates for different IoT devices. Similar or different fluctuations allow us to infer contextual IoT threat information.

compute AMI scores over the related subsets of the dataset. We select the features that perform best on average.

Table 4 presents a list of top-scoring features and their description. From these features, we discard: `opkt` and `ipkt`, because they are necessarily correlated with the higher-scoring `bpp`, `ibyt`, and `obyt`. Besides, we choose `sa` instead of `da`: we do this because we know that the dataset is biased in recognising specific targets in all scenarios where devices are attackers. Instead, discerning attack scenarios based on the source is more in line with MUD's intention (§2.1).

Overall, these AMI scores suggest that we can expect these flow features to effectively help in discerning the attack scenarios of our threat model (we demonstrate this in §6.1).

5 Methodology

We overview the high-level functioning and steps of our architecture, in §5.1. In §5.2 and the following subsections, we go through the steps in greater detail.

As the pipeline for our proposal is device-centric, all processes described in this section are referred to network activities and the MUD profile specific to one device in one deployment, unless stated otherwise.

The code we developed to implement our pipeline is open source, available at https://github.com/lucamrgs/MUDscope.

5.1 High-level architecture

The MUD profile of an IoT device white-lists all network traffic that the device is expected to engage with. Collecting device-specific traffic that is not allowed by its MUD rules, thus, directly yields all anomalous network activities relative to the device in its deployment. For the sake of conciseness, we refer to a particular device in its particular deployment as a 'device-deployment pair', DDP.

We can thus link MUD-rejected traffic to its DDP metadata: device model, device functionality (e.g., camera, motion sensor), device vendor, deployment type (e.g., industrial IoT, office, home-office, cafe), deployment location. This conceptual setup allows using a device's MUD as a *highly specialised IoT network telescope*: it captures necessarily anomalous traffic for a specific IoT setting.

By monitoring and describing the evolution of the MUD-rejected traffic of a DDP, we can observe how its captured anomalous activities fluctuate. With these notions, we can *compare multiple MUD-rejected traffic fluctuations* from several DDPs, over a given time window. This allows us to assert whether and which IoT devices are being targeted by *similar* or *different* anomalous network events at a given time (or targeted by anomalous activities at all).

In other words, this approach produces a heatmap of anomalous activities, displaying what specific devices and deployments they target. For instance, we can capture, over a time window, a common increase

in anomalous activities targeting all Huawei HG532 routers; a brand of motion sensors deployed in smart airports; or all home offices of government personnel.

The high-level logical components of our proposal are represented in Figure 2. Once a MUD profile for a device is retrieved by a local deployment **(0)**, the main steps of our proposal consist in the following. Collection of MUD-rejected traffic **(1)**, where we listen to device-specific traffic, filter, and save traffic that does not abide by MUD rules. We couple this rejected traffic with device and deployment metadata **(2)**. At **(3)**, we cluster and describe (i.e., *characterise*) MRT, discriminating different types of anomalous traffic MUD filters. We log these characterisations and how they change over time in an MRT feed **(4)**. We finally compare MRT feeds from multiple deployments at **(5)**, to capture a view of how IoT-targeting anomalous activity spreads at scale.

5.1.1 MUD acquisition

At step **(0)** in Figure 2, a local environment has obtained and validated a MUD profile, for a given device. The communication rules by which the device shall abide are thus known in the local network.

5.1.2 Collection of MUD-rejected traffic

An MRT collector module (**(1)** in Figure 2) listens to the traffic that the device is engaging with. The traffic that does not abide by the device's MUD is saved at regular timeouts, for a given time-value t. This returns consecutive 'snapshots' of rejected activities, making it easier to capture partial or full anomalous packets exchanges between a device and an unknown host, which we can later match against rejected traffic from other profiles (thus DDPs).

The module produces a series of consecutive MRT files, each spanning t time. The MRT is coupled with information specific to the IoT device and the deployment (at **(2)**, Figure 2) — a *context* for the MUD-rejected traffic.

5.1.3 MRT characterisation

For each t-time MRT collected, we generate a higher-level characterisation of the anomalous activities that it captures (at **(3)** in Figure 2). In particular, we discriminate network events by clustering packet exchanges with similar features. From this, we produce a descriptive snapshot of the anomalous communications targeting an IoT device at a given time t.

This characterisation abstracts technical details of the rejected traffic, and allows us to extract clusters' features to more easily describe how it evolves. Moreover, by adding a layer of abstraction we can anonymise IP addresses and discard any potential private information in the MRT.

5.1.4 MRT Feeds

For each pair of MRT characterisations over successive time windows (t_i, t_{i+1}), we describe their difference, and record it in an 'MRT transition entry', expressing how MUD-rejected traffic 'changes' through the two time windows. For a series of $(t_m, ..., t_n)$ successive MRT characterisations, we thus obtain a series of entries describing the differences between successive pairs (**(4)** in Figure 2).

This series of transition entries is an *MRT feed*, finally representing how the anomalous activity fluctuates for the IoT device-deployment pair that the MUD covers.

5.1.5 MRT feed comparison

Each DDP thus yields an MRT feed showing the fluctuations of anomalous activities for an IoT device. We finally compare multiple such feeds over given time-spans (**(5)**, Figure 2). If two or more MRT feeds show similar fluctuation, we infer that the related devices-deployments are experiencing the same anomalous network activities. If two or more MRT feeds are highly dissimilar, then the related device-deployments are experiencing different anomalous activities. Generally, sensing fluctuations altogether suggest that an anomalous event is taking place.

Overall, by observing both intensity and similarity of fluctuation across multiple DDP MRT feeds, and investigating equivalences in their associated metadata, we produce fine-grained contextual awareness for network threats targeting IoT devices, and associated deployments.

5.2 Collection of MUD-rejected traffic

Figure 3 presents a more detailed overview of our approach implementation, which we discuss next.

In step 1 in Figure 3, we collect the network traffic flowing in a local environment which hosts IoT devices. We save the traffic in *packet capture* files (pcaps). We then split these pcaps on a time-window basis (obtaining `t-`pcaps). In step 2, we filter device-specific traffic from a `t-`pcap. We match each packet in this traffic against the MUD rules of the device. We convert all device-specific traffic that does not abide

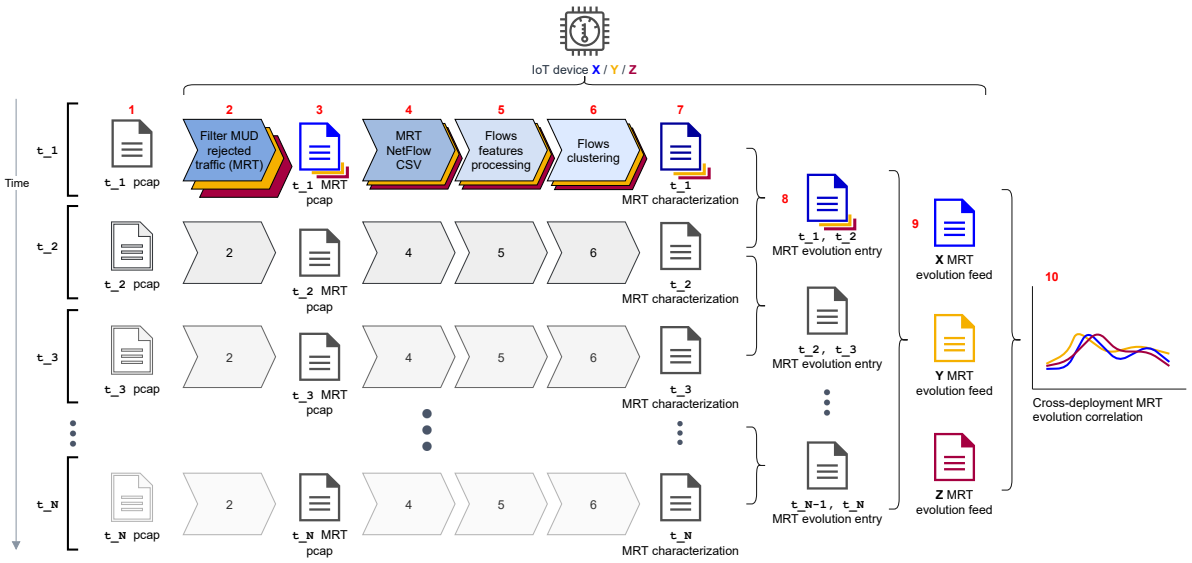

Figure 3: Pipeline for our MUD usage proposal. The sequence of steps of our pipeline is detailed for the time window t_1. One colour codifies one device. At 1, we collect a traffic capture from the local network. In 2, we filter device-specific MUD-rejected traffic and output an MRT pcap (3). In steps 4 and 5 we generate the corresponding MRT flow file, and in 6 we cluster similar flows together, to obtain an MRT characterisation. This characterisation offers a snapshot of anomalous network events happening at time t_1 (7). Then, for each pair of successive $(t_i – t_i + 1)$ characterisations, we describe how the snapshot changes in an MRT evolution entry (8). All such entries over a $t_1 – t_N$ sequence constitute an MRT evolution feed (9). In 9, we finally compare MRT feeds from different devices, to capture context-specific fluctuations of unsolicited IoT-related traffic.

by the device's MUD rules to *bidirectional flows*. We save these rejected flows to a `t-MRT` pcap file (step 4).

We use wireshark[14] to collect network traffic. We implement an MRT collector module with `python3.8` and `scapy`[15]. We extract flows from pcap files using `nfdump`.[16]. Each flow is described by features such as start and end time, duration, source and destination addresses and ports, packets and bytes per second (`pps`, `bps`), bytes per packet (`bpp`), and TCP flags (`flg`). A table presenting all flow features can be consulted in Appendix A.2.

5.3 MRT characterisation

We now want to be able to differentiate the flows in an MRT CSV in the different types of activities that they underlie. For instance, we want to group MUD-rejected flows that concern a third-party tracking the usage of an IoT device. Or, group together all flows that originate from scanning activity by a remote host; or DoS performed by bots from a same botnet.

In other words, we want to group traffic belonging to common such 'network events', captured by a MUD filter. This way, we achieve an event-based description of what anomalous traffic targets IoT, contributing to a better understanding thereof. Our intuition is that all flows of a specific network event can be recognised by similar characteristics of their features. Because network 'events' cannot be enumerated, nor precisely identified, we choose to perform this task with a clustering algorithm.

5.3.1 Clustering MUD-rejected flows

We choose `HDBSCAN` (hierarchical density-based spatial clustering of applications with noise) [46] as clustering algorithm, because it is particularly suited for its generalization, and time efficiency [47]. HDBSCAN yields obtained clusters, plus a noise group for the data-points that could not be collocated to any cluster.

To cluster the MUD-rejected flows, we need to represent numerically each of their features. To this end, we process non-numeric features of the rejected flows (step 5, Figure 3). We resolve IP addresses to 'private', 'public', 'broadcast', 'reserved' categories, encoded as ordinal numbers. We then represent in one-hot encoding the set flags for each flow, and generate a respective integer corresponding to the decimal value of the flags bits array.

[14] https://www.wireshark.org
[15] https://scapy.readthedocs.io/en/latest/
[16] https://manpages.ubuntu.com/manpages/xenial/man1/nfdump.1.html

Following best practices, we scale flow features through standardization. To produce a consistent standardization of the features at each t-time MRT CSV of different devices, we fix mean and standard deviation from the empirical observations of a *reference scaling dataset* — specific to an IoT environment. We obtain this reference dataset simply from collecting a *large* amount of observed traffic (e.g., over one day), and transforming the captures to flow format. We note that an MRT scaling reference for a device can be produced at any given time.

In step 6 (Figure 3), we thus cluster the MRT flows, based on the most events-discriminating features, as found in our preliminary analysis (§4.3): bytes per second (`bpp`), TCP flags value (`flgs_int`), output and input bytes (`obyt`, `ibyt`), destination port (`dp`), and (category of) source address (`sa`).

In our implementation, we pre-process non-numeric features as described with `python` libraries `IP.iptype()`,[17] and `sklearn`.[18]. We tune standardization parameters on a reference scaling dataset with `sklearn`'s `StandardScaler`[19]. We cluster flows at each t-MRT CSV with `python3`'s `hdbscan` [46].

5.3.2 Characterisation file

For each MRT flows file for time t, we obtain a file listing a set of clusters. These clusters represent the MUD-rejected network events taking place in the capture. We report the results of clustering, and the related rejected flows, in an *MRT characterisation* file (step 7, Figure 3).

This characterisation file contains, for each cluster, statistical descriptions of the flows belonging to each cluster, and a 'spatial' description of the obtained cluster. Among the information in the spatial description is the number of points (flows) of the cluster, the within-cluster distances average, standard deviation, minimum and maximum coordinates among the points, the centroid, and a *meta-centroid*. The meta-centroid consists of the centroid (given by the average of all clusters' points), plus two additional dimensions: average and standard deviation of between-cluster points distances. The meta-centroids are thus $n+2$ dimensional, with n = number of flow features selected for clustering. We do this to have a more robust notion of *similarity* of two clusters over different characterisations.

[17] Outputs categories such as 'private', 'public', 'broadcast', 'reserved'. Open source at https://pypi.org/project/IPy/.

[18] https://scikit-learn.org/stable/

[19] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

As we explain next, the flow descriptors for each cluster allow look-ups for captured anomalous events, while we use the spatial cluster descriptors to describe how MUD-rejected traffic evolves through time.

5.3.3 Description of MRT evolution

We record how MUD-rejected traffic varies over time, by describing the differences between two consecutive MRT characterisations. This way, we keep track of fluctuations for the MRT of a device in a sequence of time windows. We later compare these fluctuations against those in MRT from other devices, to infer if and how anomalous activities target multiple device-deployments over a time span.

Each cluster in each characterisation is linked to the respective set of captured flows, for traffic rejected by a MUD profile of a device at a given time. Describing MRT evolution based on the flows comes with several challenges: for each two successive characterisation, it is not granted that clusters identified by a label consistently map to the same type of network event, let alone if the number of clusters varies across characterisations (for instance, cluster *2* at time t may split and map to clusters *8* and *10* at time $t+1$).

To account for this, we describe MRT evolution by tracking the *spatial* evolution of clusters (such as splits and merges, appearances and disappearances, and shifts) between characterizations. We map each network event to the spatial descriptor of its related cluster — its *meta-centroid* (§5.3.2).

5.3.4 Cluster evolution

We describe how clusters evolve, through the changes in the distances between clusters' centroids, from a characterisation at time t_a, to the successive characterisation at time t_b.

We refer to *two sets* of clusters $C_a : \{c_i^a, i = 0, ..., |C_a|\}$, over t_a, and $C_b : \{c_j^b, j = 0, ..., |C_b|\}$, over t_b. The time window t_a *precedes* t_b, i.e., start-time of t_a precedes start-time of t_b, and end-time of t_a precedes end-time of t_b. A c_i^a is the meta-centroid of the *non-noise* cluster number i, for the clustering performed in the time window a.

We compute the distance matrix between pairwise clusters $M[i,j] = dist(c_i^a, c_j^b)$, where $dist(x,y)$ is the Euclidean distance.[20] Note that row indices correspond to clusters in C_a, and column indices to clusters in C_b. From analysing the distance matrix, we

[20] We use `scipy`'s `distance_matrix`, at https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance_matrix.html.

define three 'match' cases that describe shifts, splits, and merges of clusters.

For a t_a preceding t_b, the following cases are defined. The cluster $c_i^a \in C_a$ **forward-matches** with the cluster $c_j^b \in C_b$. This means that c_i^a is geometrically closest (i.e., is at the minimal distance) to c_j^b from all clusters in the *future* time window t_b. We say $m_{fwd}(c_i^a) = c_j^b \iff$

$$c_j^b = argmin_{j=0,\dots,|C_b|}(dist(c_i^a, c_j^b)).$$

We also account for $\overline{v}_{fwd(i,j)} = \overline{c}_j^b - \overline{c}_i^a$, the relative forward-match vector: as it describes the direction of the shift, it may reveal further information of cluters' evolution. Similarly, c_j^b **backward-matches** with c_i^a when the cluster c_j^b in the new time window is closest to the cluster c_i^a from the previous time window t_a. We say $m_{bwd}(c_j^b) = c_i^a \iff$

$$c_i^a = argmin_{i=0,\dots,|C_a|}(dist(c_i^a, c_j^b))$$

and $\overline{v}_{bwd(j,i)} = \overline{c}_j^b - \overline{c}_i^a$ is a backward-match vector. A **mutual match** between c_i^a and c_j^b, for fixed i and j, happens $\iff$

$$m_{fwd}(c_i^a) = c_j^b \wedge m_{bwd}(c_j^b) = c_i^a$$

The corresponding $\overline{v}_{mut(i,j)} = \overline{c}_j^b - \overline{c}_i^a$ is the mutual match vector. A cluster in C_a has no backward matches in C_b, and a cluster in C_b has no forward matches in C_a.

Through these metrics, clusters C_a forward-matching on one cluster in C_b express a merge of clusters. Clusters in C_b backward-matching one cluster in C_a expresses a split of clusters. Mutual-matching clusters correspond to cluster shifts.

Table 5 shows two examples of clusters evolution.[21] In the table above, we see a relatively stable evolution: just one new cluster (3^b) appears at a very short distance from the existing ones. Below, the high agglomeration of forward matches highlights a *merge* of clusters to 3^a, 5^a, and 6^a, to 3^b.

Ca \ Cb	0	1	2	3	4	5
0	0.020858	2.419684	2.405222	2.403964	2.403957	2.403956
1	2.416821	0.001128	0.281784	0.277283	0.277189	0.277154
2	2.402623	0.281764	0.001103	0.052307	0.052008	0.051955
3	2.400945	0.277189	0.051946	0.003871	0.001101	0.002798
4	2.400944	0.277154	0.051893	0.006378	0.002798	0.001101

Ca \ Cb	0	2	3
0	3.019654	3.596038	3.825001
1	0.354194	2.373034	2.351585
2	2.495640	0.162561	0.352583
3	2.464904	0.412329	0.247349
4	2.399984	0.330156	0.099545
5	2.564678	0.501293	0.412108
6	2.476171	0.341645	0.176153

Table 5: Two scenarios exemplifying cases of **mutual**, **forward**, and **backward** matches between clusters over consecutive time windows. In the case above, MUD-rejected traffic is relatively stable from C_a to C_b: only one backward match at a very close distance is recorded. In the case below, we observe a merge of clusters: three forward matches agglomerate over cluster c_3^b.

5.4 MRT feeds

We build a log that describes how clusters of MUD-rejected anomalous activities vary over time. First, we define metrics that represent how clusters evolve through two consecutive time windows. We record these metrics in an *evolution entry*. Then, we append consecutive evolution entries, yielding an MRT feed. An MRT feed lists all changes of MRT through time, therefore describing MUD-rejected traffic fluctuations.

5.4.1 Evolution entry

Using the notions of mutual, forward, and backward matches we described in §5.3.4, we derive a set of metrics (features)[22] that describe how the MUD-rejected traffic varies, in consideration of how clusters 'move' through time (step 8, 3). We report these features in an *evolution entry*, which we overview next. A complete description of all MRT Evolution features can be found in Appendix C.2.

All features refer to the comparison between two consecutive characterisations. We first include metadata features such as the dimension of the centroids, start and end time of the two characterisations, number of clusters at each characterisation, and difference in percentage of noise points. We do so to provide proxy information to verify if comparing two MRT feeds is possible (e.g., given the dimension of the centroids), or that that may offer early explanation of the evolution of rejected traffic, and prevent false alarms (for instance, one circumscribed high fluctuation in an MRT feed may be due to very distanced time windows between the two successive charactersiations, or a sub-optimal tuning of a clustering algorithm, constantly yielding a high number of noise points).

Next, we include *overview* metrics: balance of gained-to-lost clusters (`clusters_balance`); average of all euclidean distances over the distance ma-

[21] We generated both distance matrices with our pipeline, on two consecutive characterizations of `synflood` events — one for the Ezviz, and one for the Nugu devices in Kang's dataset [42].

[22] We use the terms *metric* and *feature* interchangeably.

trix of meta-centers (`all_dists_avg`); `mut`, `fwd`, and `bwd_matches`, the number of respective mutual, forward, and backward matches; respective `mut`, `fwd`, `bwd_matches_percentage`, the percentage of match events of each type over all match events; finally, `fwd` and `bwd_agglomeration_average`, indicating over how many clusters, in average, forward matches and backward matches agglomerate. We select these metrics as *marker* features, because the remaining metrics derive from them.

For the remaining features, for each average (`*_avg`) metric, we include a respective standard deviation metric (`*_std`). We then include the *deciles* over all distances (`all_dists_deciles`), to have a representation of their distribution. We also include ten (10) *decile* vectors (`mut`, `fwd`, `bwd_vects_decile_#`), to represent the distribution of shift's dimensions vectors for each specific match case. All vector features are meta-centroid–dimensional. For forward and backward agglomeration, we account for `std`,`max`, and `max_percentage`).

Note that all vector and decile features are multi-dimensional, whilst the others are mono-dimensional.

5.4.2 Expected features behaviour

We give some examples to clarify how we expect the metrics we described for an evolution entry to represent the behaviour of anomalous activities.

If a device is subject to (MUD-rejected) *harmless noise* (such as background Internet radiation: a constant noise of anomalous packets spreading through Internet [48]) over time, we expect the values of the features (i.e., columns) in its MRT feed to show little variation. The number of clusters will remain stable, mutual matches percentages will stay close to 100%, and mutual matches vectors will be of constant size, averaging on a fixed value.

A *new* network event would generate the following reactions in the feed values. The clusters balance will increase, coupled with an increase of backward matches, agglomerated on the new clusters — because previous clusters will still mutually match. Backward matches vectors will be longer than previously observed, because the new clusters will collocate further from the existing ones. If a network activity *ceases*, then the number of clusters will reduce, and forward matches will increase — corresponding to the previously existing clusters pointing ahead towards the remaining clusters. Again, forward vectors would be relatively longer than previously observed.

One same network event that changes over time may yield merges and splits of its related clusters. The consequences are similar to appearances and disappearances of events, but forward and backward vectors will be comparatively shorter (because clusters underlie the same events). In the case of generally dynamic changes of events, though there might be no specific patterns that may describe them, values in an MRT feed will still yield overall turbulence, allowing to assert that anomalous activities are, in fact, taking place.

5.4.3 MRT feed

In summary, by comparing two (t_a, t_b) consecutive characterisation we obtain an evolution entry describing how clusters (thus MUD-rejected traffic) 'change' from t_a to t_b. We do the same for successive pairs of characterisations ((t_b, t_c), (t_c, t_d)...). Therefore, we produce a list of entries describing how MUD-rejected traffic changes through time.

The consecutive evolution entries are appended together, and generate an *evolution feed* (9 in Figure 3), describing the evolution of the clusters mapping MUD-rejected network events. An MRT feed is thus a dataset where rows list the descriptions of consecutive transitions from characterisation to characterisation of MUD-rejected traffic. Each column lists the values of an evolution entry feature (e.g., the clusters balance) over time. We provide in Appendix C.3 a link to an example of an MRT feed.

5.5 MRT feeds comparison

So far, *for a given device in a specific deployment*, we filter the MUD-rejected traffic to consecutive time-window–based pcap files. Each pcap file is converted to a flows file, and the flows are clustered together in network events, described by an MRT characterisation file. Consecutive MRT characterisations are processed in consecutive pairs. Each pair outputs an evolution entry. In turn, consecutive evolution entries form an MRT (evolution) feed.

Our ultimate goal is to compare MRT feeds *of devices in different deployments*, to capture both common and unique fluctuations MUD-rejected activities across a selection of devices, through a given time span (10 in Figure 3). Matched with feeds metadata, this information highlights compromise attempts that are specific, for instance, to a single deployment, to a device model, a geographical area, or to an industry type.

To this end, we build a module that processes an arbitrary number of MRT feeds, spanning on the same

sequence of time windows. In particular, we compute the correlation for fluctuations of the same metrics across them, to capture common or different events.

5.5.1 MRT features correlation

For a feature column f (as per MRT evolution entry definition, §5.4.1) across MRT feeds, the module computes a correlation matrix $M_f[i,j] = \widehat{cor}(F_i[f], F_j[f])$, where $F_i, i = 0, ..., n$ is one of n MRT feeds.

For mono-dimensional features (recall §5.4.1 for mono and multi-dimensional features), $\widehat{cor}(F_i[f], F_j[f])$ simply computes the Pearson correlation coefficient[23] between the two columns of feature f in the feed i and j. For a multi-dimensional feature f' (of dimension D), their correlation, across two MRT feeds, is the average over the correlation of their dimension columns two-by-two: $\widehat{cor}(F_i[f'], F_j[f']) = avg(cor(F_i[f'[k]], F_j[f'[k]]), k = 0, ..., D)$, where cor computes the Pearson correlation coefficient.

5.5.2 Expected MRT comparison behaviour

According to the expected behaviour (§5.4.2), if the MUD-rejected traffic for two devices evolves similarly, we should be able to record high (positive)[24] correlation across their MRT feeds. If, instead, their MUD-rejected traffic behaves in different ways, we should observe low correlation.

We note that we may expect high correlation in case the devices are subject to the same baseline non-harmful noise. Therefore, for each M_f, we also plot a graph to visualize the trend of the MRT feed feature. This also allows to set thresholds to automatically detect spikes, or anomalous deviations altogether. Such anomaly detection routines to systematically detect deviations from baseline noise will be addressed in future work.

Moreover, because MUD profiles are white-lists, we expect any profile to reject anomalous traffic equally. This should make our pipeline agnostic to the diverse complexities of MUD profiles.

6 Evaluation

In this section, we first evaluate how effective is our approach in the step of clustering different types of anomalous network activities filtered by MUD (§6.1). Then, in §6.2, we use Kang's dataset [42] to evaluate how our MRT feed correlation system responds when two devices are subject synchronously to the same and different anomalous activities. Finally, we carry experiments to evaluate our whole approach, on two real IoT deployments. We demonstrate that we can assert when devices from different deployments are targeted at the same time, by same or different anomalies. We describe related process and results in sections §6.3, §6.4, and §6.5.

6.1 Clustering of MRT flows

We evaluate the clustering procedure for MUD-rejected flows, with the flow-based dataset of MRT obtained from Kang's dataset, as described in §4.3.1.

6.1.1 MRT flow features

In our preliminary analysis (in §4.3) we find the following flow features as best representative of an attack scenario: bytes per second (`bpp`), TCP flags value (`flgs_int`), output and input bytes (`obyt`, `ibyt`), destination port (`dp`), and (category of) source address (`sa`). We thus apply `hdbscan` on these features.

The `hdbscan` clustering computes distances between data-points. Though we include categorical features (`sa` and `dp`) within the data-point (flows) dimensions, we assume this does not affect the clustering performance significantly. The source addresses are parsed to a limited number of categories (§5.3.1). Therefore, the dimension contributes with 0 distance if anomalous flows are from the same type of source address (e.g., local, broadcast, public). Otherwise, we assume they do not strongly affect proximity, as flows of the same attack match closely on other features.

We assume a similar outcome for the destination port features: attacks targeting the same port contribute with 0 distance on the metric. Otherwise, network anomalies of the same type, targeting multiple ports, are geometrically closest with respect to the other features. Though, we are aware that if very 'distant' ports are targeted, this renders the clustering less effective.

In any case, while the clustering algorithm may result less precise in terms of isolating anomalous traffic events, it still gives its description of 'what is happening' *consistently*. Finally, refer to note at §6.1.4, on improving the characterisation procedure.

[23] https://pandas.pydata.org/docs/reference/api/pandas.Series.corr.html

[24] We correlate the rejected traffic to observe if it behaves *similarly*. Therefore, in this work, we always refer to *positive* correlation, unless otherwise stated.

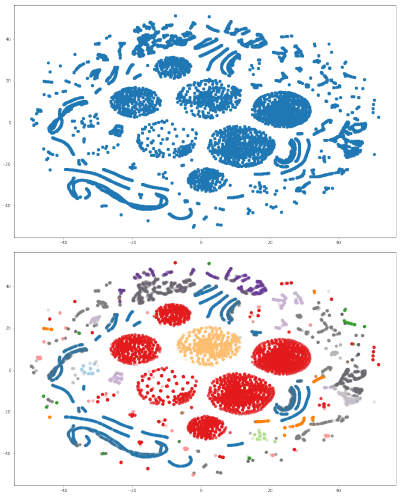

Figure 4: TSNE visualization of clustering performance. Note: the TSNE was generated on the selected features *plus* `pps` (packets per second), to improve the visualization. We verified that all *red* clusters are in fact sub-clusters of *dos-synflood*. We do not include `pps` as we observed that the additional fragmentation reduces the characterisation performance in the case of MRT files of single scenarios.

Cluster	% flows	Most represented	% purity
0	2.27	dos-synflood	94.90
1	11	unkown	100
2	1.22	unknown	100
3	1.66	dos-synflood	99.91
4	2.43	dos-synflood	100
5	56.16	dos-synflood	99.99
6	10.87	mirai-ackflood	99.85
7	1.26	mirai-httpflood	97.42
8	2.43	scan	78.08
9	5.85	scan	96.87
-1	4.79	-	-
Average	**9.09**	-	**87.91**

Table 6: Clustering results over produced clusters. Non clustered points map to '-1'.

6.1.2 Choice of clustering hyperparameters

We use the AMI and v-measure[25] between clustering labels and original attack labels to grade the performance of the clustering. Furthermore, we are interested in minimizing the number of yielded clusters and the number of non-clustered points (noise), to approximate a precise discrimination of the types of network anomalies. We thus want to consider values that produce *high* AMI and v-measure, *low* number of clusters, and low number of noise points.

Referring to the utilized `hdbscan` implementation [46], we choose to tune `hdbscan`'s `min_cluster_size` (the minimum number of points required to form a cluster) and `min_samples` (the number of points that must be in a neighbourhood of a point, for it to be considered a core point — i.e., a point part of the body of a cluster).[26] We perform a grid search over 30 combinations of `min_cluster_size` as a percentage of the dataset size, and `min_samples` as, in turn, a percentage of the `min_cluster_size` integer value. For values of 1.2% and 0.2% respectively, we obtain the best performances with AMI = 0.751, v-measure = 0.752 (homogeneity = 0.866, completeness = 0.664), 11 clusters, and 0.04% noise points. We report in Appendix B.1 the complete results of our grid search.

We specify `min_cluster_size` and `min_samples` as percentages of the dataset size, because choosing a fixed value would make the clustering less robust in case of a highly varying number of flows.[27] By making these parameters relative to the sample size, we ensure that a characterisation is produced agnostic of the number of flows reaching a device. Besides, any such output is equally valid as representation of the evolution for the MUD-rejected traffic.

6.1.3 Characterisation performance

A T-distributed stochastic neighbour embedding (TSNE)[28] visual representation of the clustering results is shown in Figure 4.

We want to have an explicit indication of how effectively the clustering algorithm isolates the attack scenarios in our custom dataset, and what attack scenarios are mostly captured. We thus define the metrics of cluster purity, label match, and label mismatch. The purity p for cluster i is $p_{C_i} = \frac{|\{f : f \in L_{C_i}\}|}{|C_i|}$, where f is a flow, and L_{C_i} is the label that is most represented in the cluster. We use this metric as a measure for the homogeneity of a single cluster. The *label match* for label i represents, in what percentage, in average, l_i is the label of the flows over the clusters where l_i is the most represented label: $ma_{l_i} = avg(\{\frac{|\{f \in l_i\}|}{|C_j|}, \forall f \in C_j, j : L_{C_j} = l_i\})$. Inversely, the label *mismatch* for label i is $mi_{l_i} = avg(\{\frac{|\{f \in l_i\}|}{|C_j|}, \forall f \in C_j, j : L_{C_j} \neq l_i\})$.

[25] https://towardsdatascience.com/v-measure-an-homogeneous-and-complete-clustering-ab5b1823d0ad

[26] https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

[27] For instance, values too small would generate a very high number of clusters, making the characterisation less informative. Values too high would make the algorithm insignificant in case a low number of flows is registered at a given time window (as could be the case for data exfiltration).

[28] dimensionality reduction, https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

Attack label	% avg over clusters		Correctly isolated	
	ma_l	mi_l	#	%
dos-synflood	99.70	15.05	44038	98.85
unknown	100	9.42	8632	82.90
mirai-ackflood	99.85	0	7659	100
scan	87.48	4.93	5342	91.86
mirai-httpflood	97.42	12.48	871	50.14
mirai-udpflood	0	6.59	-	-
mirai-brutefrc-atk	0	1.11	-	-
mirai-brutefrc-vict	0	1.11	-	-
Average	**60.43**	**6.33**	-	-

Table 7: Attack labels matches (ma_l) and mismatches (mi_l) over the produced clusters.

Table 7 lists matches and mismatches values for each dataset label. Table 6 lists the clustering results with respect to the produced clusters.

6.1.4 Characterisation discussion

All but *mirai bruteforce* and *UDP flooding* scenarios are reasonably identified (Table 7). This happens because `min_cluster_size`'s value was selected greater than the size of bruteforce and udp-flooding classes, which are therefore undetected. We run the clustering on a subset of the dataset containing only bruteforce and udp-flooding scenarios, and we observe that the clustering produces usable results in this case well. We report these in Appendix B.2.

We note that optimizing the characterisation further is beyond the scope of this work. Different parameter sets, clustering techniques, or more deterministic approaches to partition the MRT traffic, can be investigated. Overall, the three types of DoS, scanning, and miscellaneous ('unknown')[29] traffic were effectively isolated in distinct clusters. This indicates that the characterisation procedure is suitable to capture network events aligned with our threat model (§3).

6.2 MRT feeds correlation

We evaluate our MRT correlation method by applying our pipeline to Kang's dataset [42]. We recall that the dataset consists of two sets of traffic captures, one per device (Ezviz camera and Nugu speaker). Each set contains separate captures for network attack scenarios for a device (§4.1).

Per our threat model (§3), and objectives (§1.3), we want to test two scenarios of anomalous traffic patterns, reaching the two devices over the *same time span*: one where devices are subject to the same pattern, and one where they are not. With the first scenario, we mimic a 'wide-scale' anomalous network event, where a network threat rapidly targets multiple hosts. This could be the spreading of a computer worm (§9.2.1, [49]), or the outbreak of a new botnet, generating targeted scanning activities, then bruteforcing devices, over wide internet areas.

On a side note, matching time-spans of multiple devices can be extended to asynchronous cases. For example, when performing forensic analysis, to compare the network traffic that was rejected by MUD profiles of devices, before the outage of a service. In this work, we primarily focus on analysing MUD-rejected traffic on simultaneous time windows.

6.2.1 Data setup

In Kang's dataset, the two devices are subject to the same attack scenarios, though in separate periods. Hence, we simulate the scenario where the two IoTs are *targeted simultaneously by the same network attacks*, by simply processing a fixed sequence of attack scenarios, and ignoring time constraints. We apply our pipeline to both devices, and compare their MRT feeds.

To reproduce the scenario where devices are targeted simultaneously, but by different anomalous activities, we simply shuffle the order of the set of captures for each device, and proceed with the pipeline.

6.2.2 Expectations

In reference to §5.5, when the two devices are subject to the same anomalous activities, we expect to observe high feature-to-feature correlation between their MRT feeds. A high correlation means that we can sense when the MUD-rejected traffic from both devices yields the same anomalous patterns. In turn, it means that we can infer when two devices are being targeted by the same sequence of anomalous activities in given time windows. Contrarily, when the devices are subject to different anomalous activities, we expect to observe low correlations in their MRT feeds.

In both cases, we expect to observe noticeable fluctuations of the evolution of MUD-rejected traffic, because we are testing our pipeline on a succession of actual network attacks. We note that, at this stage, we do not yet have a reference for how the MUD rejected traffic evolves in the scenario where no anomalies occur.

[29] As we identified in 4.2.2, 'unknown' traffic corresponds at large to local traffic. Therefore, it is reasonable that it was isolated to a limited number of clusters.

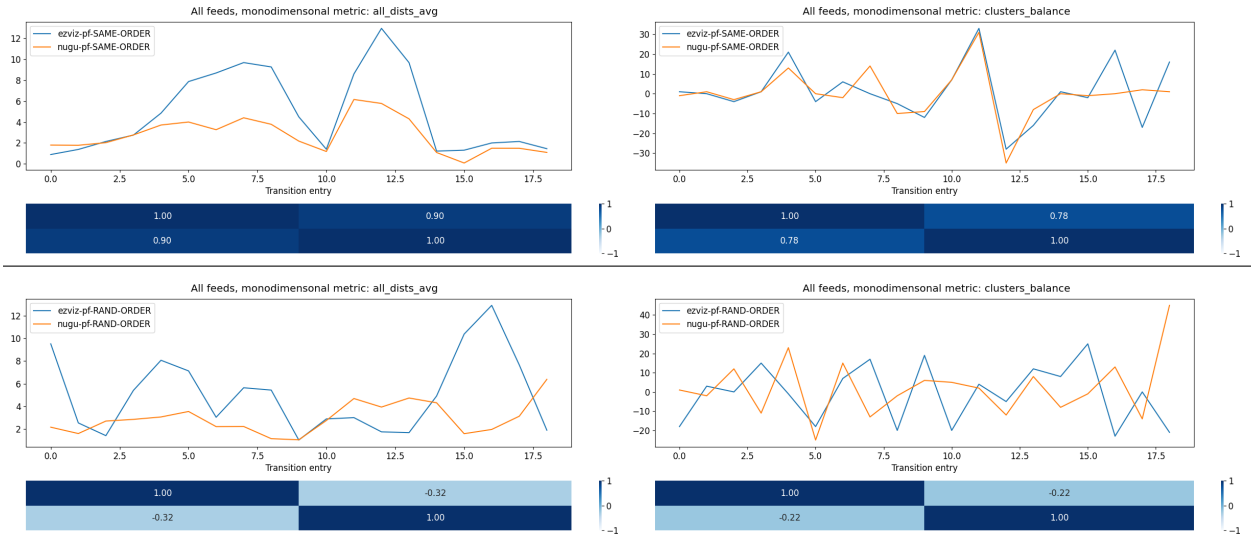

Figure 5: A sample of the preliminary results on Kang's dataset for *synchronised (above)* and *non-synchronised (below)* anomalous activities over two IoT devices. We plot the graphs of features fluctuation, and the matrix of pairwise Pearson's coefficient. We select two high-level features of the MRT feeds, expressing, for each transition between characterisations: `all_dists_avg`, the average clusters' distance between clusters of successive characterisations; and `clusters_balance`, the variation in the number of clusters.

6.2.3 Dataset-based results

Figure 5 shows a selection of two features from the results of our preliminary validation. As expected, we observe positive correlation across Ezviz and Nugu MRT feeds, in the scenario where the same pattern of anomalous network events is targeting the devices: the average correlation over all mono-dimensional features is 0.755.

Though, we observe that multi-dimensional *vector* features show instead generally low correlation, yielding a total average of 0.22 over all multi-dimensional features. We believe this is due to the relatively high dimensionality of the clusters' space, making the 'direction' of a cluster shift less likely to be consistent. Indeed, while we do not observe correlation in the multi-dimensional features, the correlation of the fluctuations for the average value of the distances between successive-characterisations (recall `all_dists_avg`, §5.4.1) clusters is 0.898.

When the devices are exposed to the second scenario (different patterns of anomalous traffic) we observe low correlation in their MRT feeds. The average Pearson correlation coefficient over all mono-dimensional features amounts to -0.086, and 0.039 for multi-dimensional features. Again as expected, we observe in both cases fluctuations in the evolution of the MUD-rejected traffic.

Besides providing an early validation of the approach, this test gives us indications on how to expect our MRT feed features to act. The complete results for these tests can be found in Appendix D.1.

6.3 Experiments planning

We finally test our approach on two real-world IoT deployments. We make use of the IoT Federated Laboratory from the Technical University of Eindhoven (TU/e), developed within the European INTERSECT. project,[30] to set up two replicated IoT network environmnts, one at TU/e, and one at University of Twente (UT). We deploy a third node at TNO facilities in Den Haag from which we send malicious traffic to the devices, to reproduce an external network threat.

We subject two pairs of devices across deployments to anomalous traffic patterns, and compare the MUD-rejected traffic fluctuations between them. Our results show that we can successfully assert when multiple devices are attacked at the same time, and if they're reached by the same attack patterns, thus producing a heatmap-like view of threats reaching IoT at multiple deployments.

6.3.1 Design of experiments

Recall from our threat model (§3) that we wish to test our approach by collecting MRT feeds from IoT devices in different 'small office, home office' (SOHO)

[30] https://intersct.nl

network environments. We want to subject IoT devices in such environments to patterns of malicious events, and study if these patterns are reflected when correlating their MRT feeds.

We scope our tests to the case of *two* SOHO environments, each hosting the same selection of IoT devices. In general terms, for at least two pairs of devices in these two environments, we can compare their MRT feeds in varying configurations for *same* or *different*: (i) device model (thus MUD profile), (ii) network environment, and (iii) network events taking place (meaning, presence or not of malicious traffic).

6.3.2 Procedure

We design our experiments with reference to two equal pairs of IoT devices **A** and **B**, with each pair being deployed in one of two environments: **1** or **2**.[31] We define a routine $r = (e_t, e_{t+1}, ..., e_{t+n})$ of network events e taking place over n consecutive time windows. Each experiment consists of choosing two of the four devices, and subjecting them *synchronously* to the routine of network events. We generate the MRT feeds for all devices at each experiment, we correlate their features (as in §5.5), and discuss the results for attacked and non-attacked devices respectively.

We choose to execute the following four experiments to test a selection of scenarios for our approach:

- **Experiment 1**: we attack two *identical* IoT devices deployed in two different environments. This reproduces a scenario where a network threat targets one specific IoT model.
- **Experiment 2**: we attack two *different* IoT devices deployed in two different environments. This is to test whether we're able to capture the same anomalous network threat reaching multiple IoT devices, agnostic of their type.
- **Experiment 3**: We attack two *similar-purpose* IoT devices in two different deployments. This reproduces the case where anomalous activity targets one *type* of IoT device.
- **Experiment 4**: We attack two *different* IoT devices at the *same* deployment. This reproduces a scenario (combined with that of experiment 2) where a network threat aims at compromising IoT devices in one specific deployment (or deployment type).

For these experiments, our approach is successful if (i) MRT feeds of devices attacked synchronously show similar fluctuations, thus yielding high correlation throughout their metrics; (ii) MRT feeds of non-attacked devices do not correlate through multiple features, and show a relatively 'flat' trend for MRT events (and do not correlate with MRT feeds of attacked devices). This indicates that we can sense when devices from different deployments are reached by the same or different anomalies, or reached by anomalies at all.

[31] We note that for a selection of more than two devices, replicated in two environments, the experiments ca be extended by choosing multiple 'A-B' devices pairs.

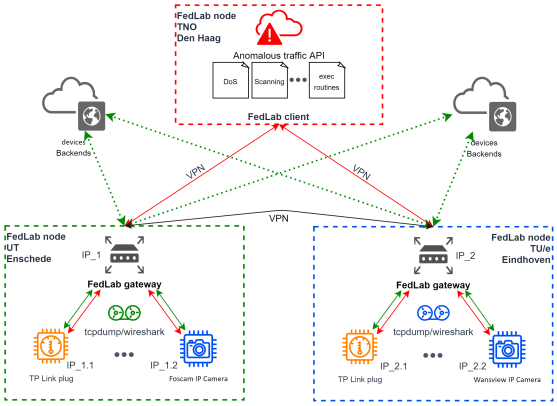

Figure 6: Configuration of the IoT Federated Lab for our experiments. Subnets at University of Twente and of Eindhoven host the same selection of IoT devices, operating normally, representing SOHO environments. A third node at TNO, in Den Haag, launches malicious traffic against the devices, acting as a network threat agent.

Device	IoT type	TU/e	UT
Wansview 1080p Q5	IP camera	✗	✓
Foscam C1-V3	IP camera	✓	✗
TP-link HS100/HS110	Power plug	✓	✓
Ring Video Doorbell	Doorbell	✓	✓
Calex Motion Sensor	Motion sensor	✓	✓

Table 8: IoT devices deployed at TU/e and UT laboratory environments. For our experiments, we use the non-greyed out devices: The Wansview and Foscam as two IP cameras, and two identical TP-Link plugs.

6.4 Experiments Setup and execution

6.4.1 Network environments

We deploy two replicated SOHO-like network environments, one at the University of Eindhoven (TU/e), and one at the University of Twente (UT). Both host a quasi-identical selection of commercial IoT devices, listed in Table 8. We experiment with two replicated pairs of IoT devices: an IP camera (Wansview 1080p Q5 at UT, Foscam IP camera at TU/e),[32] and a smart plug (A TP-link HS100 at both

[32] Though we had planned to use two identical IP cameras, we had to use different models for reasons outside our control.

environments).

Then, we set up a threat agent at a third location in Den Haag, at TNO facilities, from which we can launch malicious traffic to all devices. The three environments are interfaced on the same VPN, with internet-gateway at TU/e. From the TNO threat-agent node, we are thus able to directly connect to both UT and TU/e environments. We collect all traffic flowing in each IoT environment into capture files.

This setup allows us to perform our tests on deployments that closely mimic real-world ones, without overhead network configuration issues. Figure 6 presents the overall architecture of the Federated Laboratory for our tests.

6.4.2 Device's data setup

We generate pcap with 30 minutes of network activities where we normally interact with all devices. We create one such pcap per environment. We generate the four devices' MUD profiles with these pcaps.

To verify the profiles' consistency, we capture again 30 minutes of normal traffic the day following that of the original capture. We then process this second capture through each device's MUD, and observed if any traffic is rejected. Indeed, some benign IP addresses of the devices' backends are not be resolved with the domain names in the generated profiles. We thus manually insert the related forwarding rules in the MUDgee-generated OpenFlow table. Because our setup uses experimental tools, this is necessary. By design, similar issues would not arise with use of official MUD profiles.

We create environment-specific MRT scaling reference dataset (explained in §5.3.1) for both TU and TU/e deployments, by aggregating all traffic collected throughout our experiments.

Finally, we create relative deployment-device metadata pairs for each of the four devices. We specify the metadata in a JSON file. An example of metadata information for one device can be consulted in Appendix C.1. We note that this is a test version for the metadata specification, which can be enriched and defined further.

6.4.3 Network events

The routines of network events we implement include normal operations, scan, and DoS traffic. Normal operations simply correspond to *not sending any traffic* from our threat agent to a device. To perform scan events, we make use of the `nmap` tool to invoke

Exp.	UTwente		TU Eindhoven	
	TP-link plug	Wansview cam	TP-link plug	Foscam cam
1	R	-	R	-
2	-	R	R	-
3	-	R	-	R
4	R	R	-	-

R

Time window	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Event	n	n	s	s	n	n	d	d	n	n
Duration	25 minutes (each t = 150s)									

Table 9: The execution schedule of the experiments we run on our setup (above). (R) indicates that the device is subject to the attack routine, whereas the dashes (-) indicate that the device is unharmed. The routine of malicious traffic R is described below: n codifies a 'normal operations' event, s a scanning event, and d a DoS event.

'stealth' TCP SYN scans to target IP addresses.[33] Using `scapy`,[34] we implement DoS as 'syn flooding', generating random TCP packets with SYN flag to true, and sending them at relatively high frequency (one to three thousand per second) towards designated IP address and ports.

For our experiments, we thus define an anomalous routine $R = (n, n, s, s, n, n, d, d, n, n)$, where n is 'normal operations' event (the device is not attacked), s is a scan event, and d is a DoS event. We set each event to last at most for 150 seconds (2 minutes and 30 seconds), to better control the experiments. In line with our experiment design (§6.3.1), we implement a script that launches the routine *synchronously* towards two devices. This attack scenario replicates botnet spreading activities, where infected hosts first scan for victims, then launch volumetric attacks against them [50].

6.4.4 Execution

Replicating then the experiments procedure we designed (at §6.3.2), we execute the following four experiments runs, where we synchronously attack, respectively:

1. the TP-link WiFi plugs at both deployments;
2. the Wansview camera at UT, and the TP-link plug at TU/e;
3. the IP cameras models at both deployments;
4. the Wansview camera and TP-link plug, both deployed at one environment (UT, in this case)

Table 9 presents a synthetic view of the experiments. For each, we gather two 25 minutes-long packet capture files (pcaps), one per deployment. To achieve a

[33] https://nmap.org/book/man-port-scanning-techniques.html

[34] https://scapy.readthedocs.io/en/latest/

Exp	attacked devices		clusters		# matches			% matches			agglomeration avg		...	All features avg	
			balance	dists. avg	mut	fwd	bwd	mut	fwd	bwd	fwd	bwd		1-dim	+dim
1	TP-link plug UT	TP-link plug TU/e	.93	.20	.96	.90	.99	.96	.94	.98	.89	1		.80	-.07
2	Wansview cam - UT	TP-link plug TU/e	.88	.77	1	.68	.94	.33	.58	.55	.76	.99		.58	.01
3	Wansview cam - UT	Foscam cam - TU/e	.99	.76	.99	.98	1	.99	.98	.99	.99	1		.90	.42
4	Wansview cam - UT	TP-link plug UT	.98	.95	.98	.90	.98	.82	.81	.95	.94	1		.87	.35

Table 10: We report an overview of the results of our four experiments. We show the correlation coefficients over the marker features of MRT feeds, and the average correlation over all mono- and multi-dimensional features.

time-window view, we then split each such capture in ten 150 seconds-long captures.

Finally, we process each such experiment-environment sequence of pcaps through our MUD filter pipeline, for each device. We obtain four MRT feeds per experiment, of which we then correlate the features (as per §5.5).

6.5 Experiments results

For our experiments and our setup, our approach produces the expected results at large. The anomalous routines are captured equally in the MUD-rejected traffic of all attacked IoT devices, and thus reflected similarly in their MRT feeds. MRT feeds across targeted devices therefore show positive correlation on multiple features.

For non-attacked IoT devices, their MRT traffic simply captures benign broadcast and DHCP messages not explicitly included in their MUD rules (a limitation of the tool we use to generate the profiles). It also captures a minimal amount of ICMP traffic, and communication with legit backend addresses that were not correctly matched against the MUD-allowed domains. In any case, MUD-rejected traffic across unharmed devices shows — as expected — a largely 'flat' trend, and inconsistent correlation values across MRT features.

Table 10 summarises the results of the four experiments for the *marker* high-level features (§5.4.1 of MRT feeds. Appendix D.2 links to the complete results we obtained for all experiments.

6.5.1 Attacking the same device across deployments

We attacked the two TP-link plugs, and left the two IP cameras unharmed. The average correlation of mono-dimensional features of the smart plugs MRT feeds is 0.801 (std. 0.34). We observe a low correlation value (0.2) in the *all_dists_avg* metric, which we attribute to underlying differences in the MRT scaling references in this case. Otherwise, the related multi-dimensional feature of `all_dists_deciles` show a high correlation (0.81). All marker features correlate

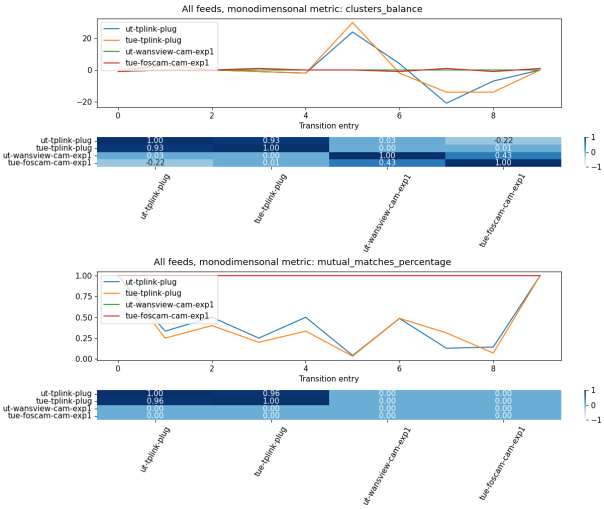

Figure 7: Visualization of the fluctuations over `clusters_balance` and `mutual_matches_percentage` metrics over devices in **experiment 1**. The two TP-Link plugs were subject to the same anomalous routines synchronously, whereas the two IP cameras were left unharmed. As expected, the clusters' balance values for the IP cameras remain around 0, and only mutual matches between successive clusters are observed. The MRTs for the attacked devices show instead higher fluctuations and correlation.

with coefficient >= .89 (recall 5.4.1 for the description of these features).

The two IP cameras show instead an average correlation of 0.04 for mono-dimensional features, with visibly 'flat' MRT. Figure 7 shows the fluctuations and correlations of two representative features for the four devices in this experiment.

6.5.2 Attacking different devices across deployments

We subject the Wansview IP camera at UT, and the TP-link plug, to the anomalous traffic routine. The plug at UT and the camera at TU/e are unharmed. Mono-dimensional features in the MRT feeds of the targeted devices show an average correlation of 0.579 (std. 0.358). Interestingly, it is the values of percentages of mutual, forward, and backward matches, and respective agglomerations, that decrease the overall correlation value. We understand that this happens because the Wansview camera responds differently to the scan and DoS traffic with respect to the plug,

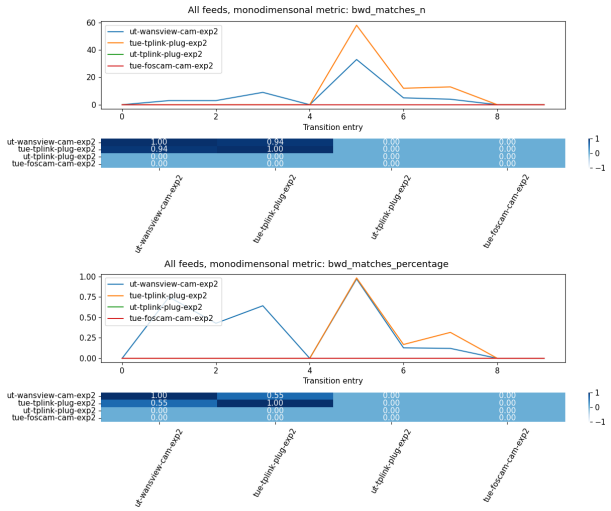

Figure 8: Fluctuations over `bwd_matches` and `bwd_matches_percentage` metrics over devices in **experiment 2**. The attacked devices (UT cam, and TU/e plug) created different response traffic to the anomalous routines. While the trend of numbers of matches events correlate, their relative percentage does not.

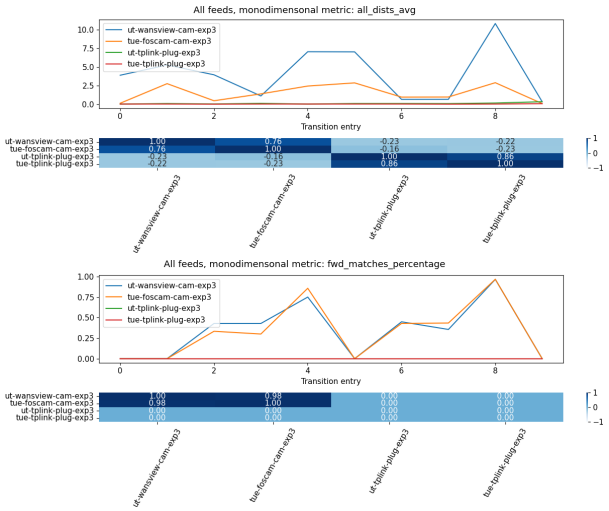

Figure 9: **Experiment 3** results sample of `all_dists_avg` and `fwd_matches_percentage` metrics. The particularly high correlation suggests that the two IP cameras (targeted in this experiment) captured and reacted to anomalous traffic very similarly.

thereby generating different volumes and flows in their MUD-rejected traffic. Indeed, the other marker metrics show generally high correlation (refer to Table 10), suggesting that the network events are in fact still clustered consistently.

The MRT feeds features for non-attacked devices show, again as expected, inconsistent correlation, and flatter fluctuations. We report again the results plots for two exemplifying features in Figure 8

6.5.3 Attacking similar devices across deployments

We attacked the two different IP cameras at UT and TU/e respectively, and left the two plugs unharmed. All marker features in the MRT feeds of both cameras show high correlation (average 0.902, variance .27). Such values suggest that, besides having captured the rejected traffic equally, the devices responded in very similar ways. The MRT feeds of the two plugs show low correlation and flat behaviour. Again, we report in Figure 9 a sample of the results for two features.

6.5.4 Attacking different devices in one deployment

We attacked both camera and plug IoT devices at UT. To compare the two symmetrical scenarios for this experiment, we later recorded 25 more minutes (the duration of the routine) of 'normal operations' traffic for the same devices at UT.

Once again, the mono-dimensional features correlate in average with 0.869, std. 0.14 (for multi-dimensional features, average 0.352, std. 0.1), when devices are synchronously attacked. When the devices are not attacked, correlations over the MRT features approaches 0, and their trends are flat. Figure 10 shows two exemplifying features results for this final experiment.

7 Discussion

We first discuss in §7.1 on the results our approach achieved, and how they compare with the state-of-art — according to the requirements we defined in §1.2.2. We then bring some considerations on scalability and use-cases for our approach, in §7.2. Finally, we discuss limitations and future work in §7.3 and §7.4.

7.1 Contextual threat awareness

7.1.1 On the results

As the MRT feeds are associated with deployment metadata (in our tests, of the universities of Twente and Eindhoven), the results of our experiments can be so interpreted: we detected one network threat simultaneously targeting: (1) smart plug devices (from TP-Link) across academic environments; (2) a range of IoT devices at academic environments; (3) IP camera IoT devices at academic environments; (4) IoT devices at the University of Twente. We thus achieved a 'heatmap' view of targeted IoT devices across multiple deployments.

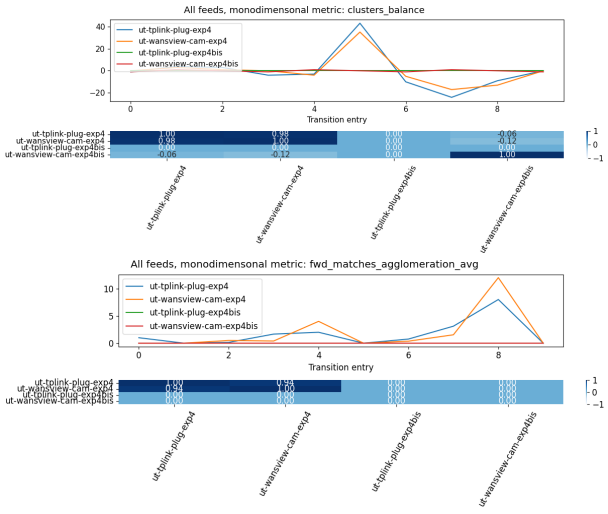

Figure 10: Sample of results for **experiment 4**: plots for `clusters_balance` and `fwd_matches_agglomeration_avg`. The represented fluctuations are for the MRT feeds from the same pair of devices, at times where they are synchronously attacked, and where they are left unharmed.

These results demonstrate that we can use MUD profiles to track the anomalous activities that IoT devices engage with, at virtually any arbitrary deployment. By observing correlations on the fluctuations of MUD-rejected traffic, we are so able to assert whether multiple IoT devices are being targeted by similar patterns of malicious traffic. By accessing MRT feeds device-deployment metadata, we can use that information to verify exactly what deployments are being targeted.

Overall, MUD profiles represent a viable and novel tool to the use-case of gathering contextual IoT threat awareness. Notably, experiments 2, 3, and 4, show that MUD 'filters' can discard the same anomalous traffic agnostic of the complexity and extension of the MUD profiles' rules. This early result shows that the *view* over the anomalous activities targeting IoT is consistent across multiple devices' models, suggesting that our approach can be used ubiquitously.

7.1.2 Challenge requirements satisfaction

In §1.2.2, we identified three requirements that we wanted to achieve with our approach: having a *specific*, *timely*, and *consistent at-scale* method to identify occurring network threats targeting IoT devices. We argue that our approach is poised to satisfy all of them, to benefit at large contextual knowledge of IoT network threats.

Our method is **specific**, as it allows coupling detailed information about a device and its deployment, together with the feed of anomalous traffic that its MUD captures. Therefore, it prompts a specific view on what are the victims of emerging network anomalies. Although we did not validate it empirically, such specific view is, in principle, geographically **scalable**, as it poses minimal requirements on local IoT network environments. Moreover, because of MUD's design (§2.1) the produced view is **consistent**, as it is agnostic of the device models, and, more importantly, anomalous traffic is discriminated with manufacturer-provided ground truth in MUD profiles. Finally, our method achieves quasi-real-time **timeliness**. It observes and characterises anomalous traffic on a continuous basis. The time required by the characterisation algorithm is dependent on the number of input data-points, i.e., the size of MUD-rejected traffic at a given time window. The width of such time windows can be flexibly changed. Besides, we can assume that, in the average case, an IoT device is not reached by anomalous traffic. Emerging anomalous activities can therefore be promptly detected.

7.1.3 Comparative note with existing approaches

We report some observations in consideration of achieving similar results with other state-of-art solutions.

Our approach differentiates from using a network intrusion detection and prevention system (NIDPS) in some respects. First, the filtering rules are manufacturer-defined and provide ground truth by default. This is different for NIDPSs, which can return false-positive reports on anomalies [51]. Second, a MUD-based approach provides a device-specific detection system, allowing direct probing of IoT anomalous traffic. Again, a NIDPS is normally configured to protect multiple hosts, and thus operates on more general filtering rules or policies. Consequently, fetching IoT-specific anomalous traffic from NIDS is less direct. Third, varying NIDPSs solutions are differently specialised and configured. Therefore, they do not operate consistently across deployments. We argue that collecting rejected traffic via a generic NIDPS-based approach is less suited to perform the fine-grained IoT anomalies sensing that we pursue.

Contributors to threat-intelligence (TI) feeds, as well, would not be able to reproduce our results with similar benefits and directness, unless all following three conditions are met: (i) attack-involved parties utilise similar NIDPS capabilities to extract the anomalous traffic; (ii) the information is published to the same threat-intelligence platforms; (iii) other deployments,

or vendors, actively consult (and enforce) such TI platforms. We argue that this would be particularly hard to realise, especially for 'consumer' IoT deployments. Instead, we argue that a MUD-based approach is comparatively easier to deploy and manage — especially if vendor-backed (as we discuss in §7.2.3).

With respect instead to network telescopes, the anomalous traffic we sent was targeted to the two environments, and would not reach any darknet. This supports the case for a deployment-centered vantage point to assert contextual information on network threats.

Finally, in regards to our experiments, an IoT honeypot — if distributed through academic environments — would have sensed the same traffic, and produced similar results. Though, denial of service and scanning are IoT-targeting anomalous activities well-known to honeypots [22]. Therefore, we argue that a honeypot would not have contributed to contextual threat awareness as directly as a MUD-based vantage point.

7.2 Considerations on the approach

7.2.1 Asynchronous and noisy anomalous events

With our experiments, we notice that the rejected anomalous patterns may produce specific fluctuations. We thus argue it is possible investigating *fluctuation-fingerprints* of (malware-specific) anomalous events — at least, for threats unaware of this approach. It would then be possible to match such fingerprints across MUD-rejected traffic feeds of IoT devices, *asynchronously*. This extends the viability of our proposed approach to asserting contextual threats also over varying time spans: from non-synchronised, day-long outbreaks of botnets, to post-breach forensic investigation.

On a related note, for a MUD-aware attacker, a natural attempt at lessening the effectiveness of our approach would be that of introducing noise and padding in performing targeted attacks. Along with targeting specific victims, a threat may mix attack vectors with noise traffic, whilst also sending noise to other random devices or deployments. Our approach would still be able to detect a synchronous increase of anomalous traffic across multiple (random or sector-specific) IoT deployments. Similar observations apply if random noise is sent at random time intervals, for instance over a day, to arbitrary IoT deployments. In such asynchronous context, our approach would record many chaotic fluctuations of anomalous traffic through a selection of deployments. Thus, it would still produce an informative view on contextual anomalies.

7.2.2 Realistic MUD enforcement

We further note that in this work, we primarily focused on highlighting anomalies *inbound* IoT devices. Our passive MUD-filtering approach would also be able to detect anomalous traffic *outgoing* compromised IoT devices. In case a device is compromised, it is then possible to review the MRT feed logs to detect or rule out internet-originated attack vectors, from MUD-disallowed sources.

While a passive MUD usage eases the deployment of our solution, in the more realistic scenario where MUD profiles are actively enforced, our approach can collect MUD-rejected packets the same way, and operate on the same working basis of a passive standpoint. We also note that the working basis of our approach stands for any network communication protocol rules included in a MUD profile. This ensures that our approach is in *ready* state as MUD profiles (or similar solutions altogether) are progressively rolled out.

7.2.3 Approach use-cases

Our approach is not dependent on MUD enforcement capabilities, nor widespread MUD adoption. In this respect, we list two use-cases that could uptake our proposed solution within the near future.

Suppose one IoT manufacturer decides to abide by the MUD standard. All its products are thus associated with MUD profiles, containing related networking rules to be translated into local forwarding tables. The manufacturer can uptake the task of collecting MRT feeds for all its deployed devices, and analyse it similarly to how we propose. With this vantage point, the manufacturer can offer product-related security intelligence to its clients, reporting on singularly or commonly shared anomalous activities. In turn, the manufacturer also has a direct view on anomalous events interesting its devices, and can further investigate whether its devices may present 0-day vulnerabilities, or overall weak security accordingly.

As a second use case, assuming more than one manufacturer implements MUD profiles, business consortia of any size could cooperate to monitor their IoT assets. For instance, emerging start-ups throughout Europe could share and co-monitor the MUD-

rejected traffic of their IoT assets, to infer contextual threats and IoT-related security posture. In the same way, universities could cooperate to leverage such vantage points to conduct further IoT anomalies research.

7.3 Limitations

7.3.1 MUD-filtered activities

Necessarily, all observations and methods offered by this work refer to anomalous IoT traffic that is directly blocked by MUD rules. The approach we propose cannot capture MUD-elusive attacks, such as packets spoofing any MUD-allowed source, man-in-the-middle, or traffic coming from compromised vendors' capabilities. Though, there exist some works that extend MUD's security grants, by introducing parties' authentication and integrity checks on MUD profiles [52–54], which would strengthen our approach in these regards.

7.3.2 MRT fluctuations

In relation to more advanced attacks, if some devices present vulnerabilities that can be exploited by network attackers, for instance via a single packet, such inbound attacks would be hardly detected by our approach. Similarly, if an attack spreads widely over time, it would hardly be linked to any fluctuation in devices' MRT feeds. These considerations call for investigating different and more advanced methods of characterising MUD-rejected traffic, based on different clustering algorithms, or finer anomalous events discrimination methods overall.

7.3.3 Extensions to corporate and industrial deployments

A third natural limitation of our approach concerns the varying security capabilities of IoT deployments. Differently from SOHO environments in our threat model (§3), corporate or industrial environments protected by multiple layered firewalls, for instance, will likely block IoT targeting malicious traffic well before it reaches a MUD 'filter' implementing our proposal. Therefore, the MRT feed for such device-deployment pairs would yield a view of network threats inconsistent with that of a same device, deployed for example in an unprotected cafe network.

We envision two ways to address this. One concerns integrating security-posture information in the MRT feed metadata of a device. This would allow a more informed comparison of MRT feeds across devices. A second solution could regard deploying such MUD filters (also) at 'edge' locations of protected networks. This way, the view of MUD-rejected traffic would result more consistent throughout deployments. In any case, capturing MUD-rejected traffic at deeply nested network environments still represents a vantage-point of interest for corporate and industrial deployments — they gain a viewpoint on what anomalous activity possibly reaches those devices.

7.4 Future work

In this work, we implemented a minimal working solution for our approach. The potential that this methodology underlies opens for much future work to pursue. And, naturally, when vendors will publish official MUD profiles for devices models, we will test our pipeline on real (non-experimental) IoT deployments.

In the first place, our experiments showed that understanding 'how much' the features of MRT feeds fluctuate can yield relevant information on whether a device is attacked or not. Follow-up work will thus concern generating suitable metrics that express the magnitude of fluctuations in MRT feeds, to integrate with correlation observations. Automatic generation of alerts will also be addressed.

In the second place, our approach is based on analysing capture files offline. One immediate development of our pipeline will regard adapting it to online usage. Further work will then be required to engineer a distributed infrastructure that can serve the aim of this approach, automatically collecting MRT feeds from multiple deployments, and sharing related information.

Finally, as we discussed, more advanced anomalous events characterisation techniques can be investigated. For instance, we observed that our `hdbscan` implementation generates a relatively high number of clusters in MRT. We will investigate new and diverse ways of characterising rejected traffic to more consistent and events-faithful views.

8 Related Work

We first report related literature referred to IoT security information collection and sharing, in §8.1. We then report the state-of-art research on MUD profiles, in §8.2.

8.1 IoT threat landscape monitoring

8.1.1 Honeypot-based works

One widely appreciated contribution to IoT honeypots is IoTPot, from Pa et al. [55] in 2015. Thanks to an adaptable backend, IoTPot studies Telnet-based IoT attacks directed to 8 different CPU architectures. The authors contribute with an analysis on the 'scope and variety' of IoT Telnet attacks. A 2018 work from Kamoen [20] builds upon the IoTPot custom-backend approach to further propose a persistent-state IoT honeypot ('Honeytrack'). Therein, they maintain the status of the attack progress from a threat agent, and they re-presented it to the agent upon their successive interactions with the honeypot. Kamoen's Honeytrack focuses specifically on studying the adversary behaviour, and how it dynamically evolves when a target is compromised and weaponised. A 2020 work from Tabari and Ou [19] similarly addresses the "largely unknown nature of attackers' activities towards IoT" (§1, second challenge, in Tabari and Ou), by proposing a honeypot whose interaction is incrementally designed and integrated. They do so to progressively understand attackers' specific interests, and thus interface simulated devices with higher-chance of compromise. They deploy their honeypot in 12 worldwide-spanning locations. Another 2020 contribution, by Wu et al., [21] proposes a controller architecture ('ThingGate') to broker configuration and communication data for bare-metal IoT honeypots. Their work is motivated by the need of studying IoT attacks in greater detail through physical honeypots.

8.1.2 Network telescope–based works

IoTPot and Honeytrack's authors make use of darknets to gain preliminary results suggesting what to account for in honeypot architectures. Indeed, darknet-based approaches are able to produce at-scale insights on IoT threats. A 2018 work from Shaik et al. [26] achieves internet-scale monitoring of compromised IoT devices, by correlating network telescope captures with threat intelligence feeds. Pour et al. [17] integrate the same setup with geolocation databases and ISPs' feedbacks, and infer at-scale IoT-probing campaign characterized both by affected industry sector and vendors. The same research group expands the approach to achieve at-scale and locality-specific IoT-botnets evolution [25] and consumer-IoT compromises [27]. Furthermore, they implement 'ex-IoT', an IoT threat intelligence feed that streams findings from such network-telescope internet-scale IoT monitoring capability [56]. Notably, a 2019 work from Griffioen and Doerr [50] studies Mirai-like botnets evolution and behaviour by leveraging on 7,500 Honeytrack [20] deployments, Delft's University network telescope, and flow probing of infected devices.

8.1.3 Considerations

Differently from the above approaches, our method proposes using real IoT deployments as a vantage point to collect malicious traffic. By design, this offers greater scalability options, and an upfront position to intercept malicious phenomena proactively. Finally, our work allows differentiating malicious traffic according to deployment characteristics, achieving a highly specific view on what are the targets of emerging anomalies.

8.2 MUD research

8.2.1 MUD in threats prevention

One first contribution to MUD research is a 2018 work by Hamza et al. [39]. The authors create a tool to generate MUD profiles from network capture files of benign traffic (*MUDgee*). Studying the MUD profiles for 28 devices, they show how the specification can be integrated with Supervisory Control and Data Acquisition (SCADA) policies. A 2018 work from Schutijser [57] also proposes a MUD-profile generation tool, and shows how the specification is effective in blocking DoS attempts. Hamza et al. build upon MUDgee and study the intrusion-detection effectiveness of the specification [58]. They show that MUD is able to block internet-side threats, for devices with limited functionality, and specifically against volumetric attacks [59]. The researchers extend their MUD-based intrusion detection method in a work from 2019 [54], where they infer different anomalous status cases of devices, by matching their dynamically observed behaviour against their MUD-expected behaviour.

8.2.2 Extensions to MUD standard

Other works are focused on MUD integration and enhancing profiles' expressiveness. Matheu et al. propose a way to extend the profiles to integrate security-testing results [60]. In related work, the authors design an SDN-backed authentication messages exchange to bootstrap MUD profiles in industrial environments [61]. Sajjad et al. [62] extend MUD profiles with firmware integrity information, fetched from vulnerabilities repositories, and distributed through a blockchain framework. Furthermore, they design a MUD bootstrapping routine that also accounts for gateway authentication, to prevent attackers to by-

pass MUD through router vulnerabilities. Feraudo et al. [52] propose a federated-learning framework[35] where only MUD-compliant devices are allowed to publish training data. Finally, Afek et al. [63] implement an ISP-level service to enforce and manage MUD on behalf of SOHOs, thus unburdening local deployments from the task.

8.2.3 Considerations

Notably, the work of Afek concludes by stating that such ISP-level MUD services are at a perfect vantage point for detecting 'global phenomena' of malicious events affecting SOHO IoT (§VI, Afek et al. [63]). In fact, to the best of our knowledge, no other MUD work has yet explored this aspect. Our work moves from Afek's idea, but favours a distributed and deployment-centric vantage point.

9 Conclusion

As the integration of internet-connected devices across societal contexts constantly increases, threats targeting IoT menace catastrophic outcomes. Enhancing collection and sharing of IoT threats information is one crucial step to counter this adverse trend.

In this work, we presented an approach to achieve a detailed monitor on how anomalous network events target IoT devices at scale. We proposed the use of the networking white-lists defined in the Manufacturer Usage Description profiles, to collect an edge-view of IoT network anomalies. Collecting the traffic rejected by devices' MUD profiles at multiple deployments yields an open window on what anomalous network events target IoT— an IoT-specialised network telescope.

We tested our vantage point in an experimental setting with two IoT deployments, and an external attacker, replicating a real-world scenario. We demonstrated that through the MUD vantage point, we can detect when similar anomalous events target IoT devices synchronously, at multiple locations. We discussed how this work can be built upon to reveal emerging attackers' interests, spotlighting vulnerable devices, or preferred targets. This knowledge can inform specific and timely remedy actions, improving the security posture of IoT manufacturers, and IoT-employing environments.

Most of all, we hope that this work has effectively presented reasons to further explore IoT security approaches based on manufacturer-provided devices specifications. These can unlock novel opportunities for stakeholders' cooperation, whilst simplifying security solutions and infrastructures, in the IoT domain.

[35] Definition of federated learning from Google at https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

References

[1] ENISA. Internet of things (IoT) — ENISA. [Online]. Available: https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures/iot

[2] N. N. Dlamini and K. Johnston, "The use, benefits and challenges of using the internet of things (IoT) in retail businesses: A literature review," in *2016 International Conference on Advances in Computing and Communication Engineering (ICACCE)*. IEEE, 2016-11, pp. 430–436. [Online]. Available: http://ieeexplore.ieee.org/document/8073787/

[3] M. b. M. Noor and W. HaslinaHassan, "Current research on internet of things (IoT) security: A survey," *Computer Networks*, vol. 148, pp. 283–294, 2019-01-15. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1389128618307035

[4] A. Holst. IoT connected devices worldwide 2019-2030. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[5] A. Zahra and M. A. Shah, "IoT based ransomware growth rate evaluation and detection using command and control blacklisting," in *2017 23rd International Conference on Automation and Computing (ICAC)*. IEEE, 2017-09, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8082013/

[6] S. R. Zahra and M. Ahsan Chishti, "RansomWare and internet of things: A new security nightmare," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2019-01, pp. 551–555. [Online]. Available: https://ieeexplore.ieee.org/document/8776926/

[7] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: Iot botnet of high wattage devices can disrupt the power grid," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 15–32. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/soltan

[8] D. McMillen. Internet of threats: IoT botnets drive surge in network attacks. [Online]. Available: https://securityintelligence.com/posts/internet-of-threats-iot-botnets-network-attacks/

[9] NETSCOUT. (2021) Cyber security & threat intelligence report | NETSCOUT. [Online]. Available: https://www.netscout.com/threatreport/

[10] T. Seals. (2021-09-06) IoT attacks skyrocket, doubling in 6 months. [Online]. Available: https://threatpost.com/iot-attacks-doubling/169224/

[11] M. van Staalduinen and Y. Joshi, "The IoT security landscape: adoption and harmonisation of security solutions for the internet of things," TNO, Tech. Rep., 2019. [Online]. Available: https://repository.tno.nl/islandora/object/uuid%3A989e7450-206f-4f7c-93aa-5587e4674781

[12] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019. [Online]. Available: https://doi.org/10.1109/COMST.2019.2910750

[13] J. Saleem, M. Hammoudeh, U. Raza, B. Adebisi, and R. Ande, "IoT standardisation: challenges, perspectives and solution," in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, ser. ICFNDS '18. Association for Computing Machinery, 2018-06-26, pp. 1–9. [Online]. Available: https://doi.org/10.1145/3231053.3231103

[14] O. Garcia-Morchon, S. Kumar, and M. Sethi. (2019-04) RFC 8576 - internet of things (IoT) security: State of the art and challenges. [Online]. Available: https://tools.ietf.org/html/rfc8576

[15] A. Costin and J. Zaddach, "Iot malware : Comprehensive survey , analysis framework and case studies," in *BlackHat USA*, 2018. [Online]. Available: https://i.blackhat.com/us-18/Thu-August-9/us-18-Costin-Zaddach-IoT-Malware-Comprehensive-Survey-Analysis-Framework-and-Case-Studies-wp.pdf

[16] M. Husák, N. Neshenko, M. S. Pour, E. Bou-Harb, and P. Čeleda, "Assessing internet-wide cyber situational awareness of critical sectors," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ACM, 2018-08-27, pp. 1–6. [Online]. Available: https://dl.acm.org/doi/10.1145/3230833.3230837

[17] M. S. Pour, E. Bou-Harb, Kavita Varma, N. Neshenko, D. A.Pados, and K.-K. R. Chooc, "Comprehending the IoT cyber threat landscape: A data dimensionality reduction technique to infer and characterize internet-scale IoT probing campaigns," *Digital Investigation*, vol. 28, pp. S40–S49, 2019-04-01. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287619300246

[18] P. Richter and A. Berger, "Scanning the scanners: Sensing the internet from a massively distributed network telescope," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. Association for Computing Machinery, 2019-10-21, pp. 144–157. [Online]. Available: https://doi.org/10.1145/3355369.3355595

[19] A. Z. Tabari and X. Ou, "A first step towards understanding real-world attacks on iot devices," *CoRR*, vol. abs/2003.01218, 2020. [Online]. Available: https://arxiv.org/abs/2003.01218

[20] S. Kamoen, "Honeytrack: Persistent honeypot for the internet of things," 2018. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3A344bd7aa-0a17-47dc-92fd-bd6f7e7b08c8

[21] C.-J. Wu, K. Yoshioka, and T. Matsumoto, "ThingGate: A gateway for managing traffic of bare-metal IoT honeypot," *Journal of Information Processing*, vol. 28, no. 0, pp. 481–492, 2020. [Online]. Available: https://www.jstage.jst.go.jp/article/ipsjjip/28/0/28_481/_article

[22] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems," *arXiv:2108.02287 [cs]*, 2021-08-04. [Online]. Available: http://arxiv.org/abs/2108.02287

[23] A. Vetterl and R. Clayton, "Bitter harvest: Systematically fingerprinting low- and medium-interaction honeypots at internet scale," in *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. Baltimore, MD: USENIX Association, Aug. 2018. [Online]. Available: https://www.usenix.org/conference/woot18/presentation/vetterl

[24] D. Moore, "Network telescopes: Tracking denial-of-service attacks and internet worms around the globe," in *Proceedings of the 17th Conference on Systems Administration (LISA 2003), San Diego, California, USA, October 26-31, 2003*, Æ. Frisch, Ed. USENIX, 2003.

[25] M. Safaei Pour, A. Mangino, K. Friday, M. Rathbun, E. Bou-Harb, F. Iqbal, S. Samtani, J. Crichigno, and N. Ghani, "On data-driven curation, learning, and analysis for inferring evolving internet-of-things (IoT) botnets in the wild," *Computers & Security*, vol. 91, p. 101707, 2020-04-01. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404819302445

[26] F. Shaikh, E. Bou-Harb, N. Neshenko, A. P. Wright, and N. Ghani, "Internet of malicious things: Correlating active and passive measurements for inferring and characterizing internet-scale unsolicited IoT devices," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 170–177, 2018-09. [Online]. Available: https://ieeexplore.ieee.org/document/8466375/

[27] A. Mangino, M. S. Pour, and E. Bou-Harb, "Internet-scale insecurity of consumer internet of things: An empirical measurements perspective," *ACM Transactions on Management Information Systems*, vol. 11, no. 4,

pp. 21:1–21:24, 2020-10-12. [Online]. Available: https://doi.org/10.1145/3394504

[28] S. Torabi, E. Bou-Harb, C. Assi, M. Galluscio, A. Boukhtouta, and M. Debbabi, "Inferring, characterizing, and investigating internet-scale malicious iot device activities: A network telescope perspective," in *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018*. IEEE Computer Society, 2018, pp. 562–573. [Online]. Available: https://doi.org/10.1109/DSN.2018.00064

[29] H. Griffioen, T. M. Booij, and C. Doerr, "Quality evaluation of cyber threat intelligence feeds," in *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, Eds., vol. 12147. Springer, 2020, pp. 277–296. [Online]. Available: https://doi.org/10.1007/978-3-030-57878-7_14

[30] X. Bouwman, H. Griffioen, J. Egbers, C. Doerr, B. Klievink, and M. van Eeten, "A different cup of TI? the added value of commercial threat intelligence," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 433–450. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/bouwman

[31] Check Point Research. (2017-12-21) Huawei home routers in botnet recruitment. [Online]. Available: https://research.checkpoint.com/2017/good-zero-day-skiddie/

[32] E. Lear, D. Romascanu, and R. Droms. (2019-03) IETF RFC 8520 - manufacturer usage description (MUD) specification. [Online]. Available: https://tools.ietf.org/html/rfc8520

[33] CISCO. (2019) Cisco DevNet - Why MUD? [Online]. Available: https://developer.cisco.com/docs/mud/

[34] I. Ghafir, M. Husák, and V. Přenosil, "A survey on intrusion detection and prevention systems," in *Proceedings of student conference Zvůle 2014, IEEE/UREL*. Zvůle, Czech Republic: Faculty of Electrical Engineering and Communication, Brno University of Technology, 2014, pp. 10–14. [Online]. Available: http://www.radio.feec.vutbr.cz/ieee/userfiles/downloads/archive/2014-Zvule/proceedings.pdf

[35] A. Giaretta, N. Dragoni, and F. Massacci, "Iot security configurability with security-by-contract," *Sensors*, vol. 19, no. 19, p. 4121, 2019. [Online]. Available: https://doi.org/10.3390/s19194121

[36] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, "Intrusion detection systems for IoT-based smart environments: a survey," *Journal of Cloud Computing*, vol. 7, no. 1, p. 21, 2018-12-04. [Online]. Available: https://doi.org/10.1186/s13677-018-0123-6

[37] S. Hajiheidaria, K. Wakilb, M. Badric, and N. J. Navimipour, "Intrusion detection systems in the internet of things: A comprehensive investigation," *Computer Networks*, vol. 160, pp. 165–191, 2019-09-04. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1389128619306267

[38] S. A. Salloum, M. Alshurideh, A. Elnagar, and K. Shaalan, "Machine learning and deep learning techniques for cybersecurity: A review," in *Proceedings of the International Conference on Artificial Intelligence and Computer Vision, AICV 2020, Cairo, Egypt, 8-10 April, 2020*, ser. Advances in Intelligent Systems and Computing, A. E. Hassanien, A. T. Azar, T. Gaber, D. Oliva, and M. F. Tolba, Eds., vol. 1153. Springer, 2020, pp. 50–57. [Online]. Available: https://doi.org/10.1007/978-3-030-44289-7_5

[39] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, and V. Sivaraman, "Clear as MUD: Generating, validating and applying IoT behavioral profiles," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, ser. IoT S&P '18. Association for Computing Machinery, 2018-08-07, pp. 8–14. [Online]. Available: https://doi.org/10.1145/3229565.3229566

[40] Trend Micro. (2017-01-17) Securing your routers against mirai and other home network attacks - security news. [Online]. Available: https://www.trendmicro.com/vinfo/gb/security/news/internet-of-things/securing-routers-against-mirai-home-network-attacks

[41] D. Dodson, D. Montgomery, T. Polk, M. Ranganathan, M. Souppaya, S. Johnson, A. Kadam, C. Pratt, D. Thakore, M. Walker, E. Lear, B. Weis, W. C. Barker, D. Coclin, A. Hojjati, C. Wilson, T. Jones, A. Baykal, D. Cohen, K. Yeich, Y. Fashina, P. Grayeli, J. Harrington, J. Klosterman, B. Mulugeta, S. Symington, and J. Singh, "Securing small-business and home internet of things (IoT) devices: Mitigating network-based attacks using manufacturer usage description (MUD)," in *NIST SPECIAL PUBLICATION 1800-15*, 2021-05-25. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-15.pdf

[42] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, "Iot network intrusion dataset." IEEE Dataport, 2019. [Online]. Available: https://dx.doi.org/10.21227/q70p-q449

[43] Open Networking Foundation. (2012-09-06) OpenFlow switch specification (1.3.1). [Online]. Available: https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf

[44] S. K. Tayyaba, M. A. Shah, O. A. Khan, and A. W. Ahmed, "Software defined network (SDN) based internet of things (IoT): A road ahead," in *Proceedings of the International Conference on Future Networks and Distributed Systems*. ACM, 2017-07-19, pp. 1–8. [Online]. Available: https://dl.acm.org/doi/10.1145/3102304.3102319

[45] D. G. R. Ruth, N. Brownlee, and C. G. Mills, "Traffic Flow Measurement: Architecture," RFC 2722, Oct. 1999. [Online]. Available: https://rfc-editor.org/rfc/rfc2722.txt

[46] L. McInnes and S. Horn. (2017-11-10) How HDBSCAN works — hdbscan 0.8.1 documentation. [Online]. Available: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

[47] L. McInnes and M. Suppa. (2018-12-31) Benchmarking performance and scaling of python clustering algorithms — hdbscan 0.8.1 documentation. [Online]. Available: https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html

[48] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement - IMC '04*. ACM Press, 2004, p. 27. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1028788.1028794

[49] C. O. E. Dictionary. (2005-07-01) Strategies of computer worms. [Online]. Available: https://cdn.ttgtmedia.com/searchSecurity/downloads/Szor_Ch9.pdf

[50] H. Griffioen and C. Doerr, "Examining mirai's battle over the internet of things," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. Association for Computing Machinery, 2020-10-30, pp. 743–756. [Online]. Available: https://doi.org/10.1145/3372297.3417277

[51] N. Hubballi, S. Biswas, and S. Nandi, "Towards reducing false alarms in network intrusion detection systems with data summarization technique," *Secur. Commun. Networks*, vol. 6, no. 3, pp. 275–285, 2013. [Online]. Available: https://doi.org/10.1002/sec.562

[52] A. Feraudo, P. Yadav, V. Safronov, D. A. Popescu, R. Mortier, S. Wang, P. Bellavista, and J. Crowcroft, "CoLearn: enabling federated learning in MUD-compliant IoT edge networks," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, ser. EdgeSys '20. Association for Computing Machinery, 2020-04-27, pp. 25–30. [Online]. Available: https://doi.org/10.1145/3378679.3394528

[53] 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

[54] 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

[55] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: Analysing the rise of iot compromises," in *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. Washington, D.C.: USENIX Association, Aug. 2015. [Online]. Available: https://www.usenix.org/conference/woot15/workshop-program/presentation/pa

[56] M. S. Pour, D. Watson, and E. Bou-Harb, "Sanitizing the IoT cyber security posture: An operational CTI feed backed up by internet measurements," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2021-06, pp. 497–506. [Online]. Available: https://ieeexplore.ieee.org/document/9505129/

[57] C. J. T. M. Schutijser, "Towards automated DDoS abuse protection using MUD device profiles," 2018-08-30. [Online]. Available: http://essay.utwente.nl/76207/

[58] A. Hamza, H. H. Gharakheili, and V. Sivaraman, "Combining MUD policies with SDN for IoT intrusion detection," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, ser. IoT S&P '18. Association for Computing Machinery, 2018-08-07, pp. 1–7. [Online]. Available: https://doi.org/10.1145/3229565.3229571

[59] 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. Association for Computing Machinery, 2019-04-03, pp. 36–48. [Online]. Available: https://doi.org/10.1145/3314148.3314352

[60] S. N. Matheu, J. L. H. Ramos, S. Pérez, and A. F. Skarmeta, "Extending MUD profiles through an automated iot security testing methodology," *IEEE Access*, vol. 7, pp. 149 444–149 463, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2947157

[61] S. N. Matheu, A. Molina Zarca, J. L. Hernández-Ramos, J. B. Bernabé, and A. S. Gómez, "Enforcing behavioral profiles through software-defined networks in the industrial internet of things," *Applied Sciences*, vol. 9, no. 21, p. 4576, 2019-10-28. [Online]. Available: https://www.mdpi.com/2076-3417/9/21/4576

[62] 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. [Online]. Available: https://ieeexplore.ieee.org/document/9187209/

[63] 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

A External material

A.1 MUD profile example from RFC 8520

Sample of a MUD profile for a generic smart lightbulb, listed in RFC 8520 [32].

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url":
        ↪ "https://lighting.example.com/lightbulb2000",
    "last-update": "2019-01-28T11:20:51+01:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "The␣BMS␣Example␣Light␣Bulb",
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6fr"
          }
        ]
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-76100-v6to"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-76100-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-todev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:src-dnsname":
                      ↪ "test.example.com",
                  "protocol": 6
                },
                "tcp": {
                  "ietf-mud:direction-initiated":
                      ↪ "from-device",
                  "source-port": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "mud-76100-v6fr",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-frdev",
              "matches": {
                "ipv6": {
                  "ietf-acldns:dst-dnsname":
                      ↪ "test.example.com",
                  "protocol": 6
                },
                "tcp": {
                  "ietf-mud:direction-initiated":
                      ↪ "from-device",
                  "destination-port": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

A.2 Netflow features

Selected non-protocol–specific features from the `nfdump` tool, at https://manpages.ubuntu.com/manpages/xenial/man1/nfdump.1.html.

- ts. Start Time - first seen;
- te. End Time - last seen;
- td. Duration;
- pr. Protocol;
- sa. Source address;
- da. Destination address;
- sp. Source port;
- dp. Destination port;
- sas. Source autonomous system;
- pas. Previous autonomous system;
- ipkt. Input packets;
- opkt. Output packets;
- ibyt. Input bytes;
- obyt. Output bytes;
- flg. TCP flags;
- dir. Direction: ingress, egress;
- bps. Bytes per second;
- pps. Packets per second;
- bpp. Bytes per packet.

B HDBSCAN clustering evaluation

B.1 Grid search on HDBSCAN parameters

As explained in the main body of this document, we performed the grid search on thirty (30) combinations of percentage value of `min_clusters_size` for the entries over all entries of the dataset, and relative percentage of `min_samples`, for the entries over the number of entries in `min_clusters_size`. Below the listing of the output.

```
[({'min_c_size': 0.3, 'min_s_core': 0.1, 'n_c_size':
     211, 'n_s_core': 21},
  {'ami': 0.7176737121497185,
   'compl': 0.599110472342462,
   'hmg': 0.8955883999048692,
   'n_clusters': 26,
   'noise': 2073,
   'v_score': 0.7179458006611124}),
 ({'min_c_size': 0.3, 'min_s_core': 0.2, 'n_c_size':
     211, 'n_s_core': 42},
  {'ami': 0.7184539366004842,
   'compl': 0.6046915439469844,
   'hmg': 0.8857045109199511,
   'n_clusters': 24,
   'noise': 2621,
   'v_score': 0.7187056439663094}),
 ({'min_c_size': 0.3, 'min_s_core': 0.3, 'n_c_size':
     211, 'n_s_core': 63},
  {'ami': 0.7217725018069703,
   'compl': 0.6089686076366687,
   'hmg': 0.8865897762418038,
   'n_clusters': 23,
   'noise': 2755,
   'v_score': 0.7220117213782395}),
 ({'min_c_size': 0.3, 'min_s_core': 0.5, 'n_c_size':
     211, 'n_s_core': 105},
  {'ami': 0.7216447821321361,
   'compl': 0.6066574801974591,
```

```
  'hmg': 0.8911742207570977,
  'n_clusters': 24,
  'noise': 2325,
  'v_score': 0.7218935302769877}),
({'min_c_size': 0.3, 'min_s_core': 1, 'n_c_size': 211,
    'n_s_core': 211},
 {'ami': 0.7167467748048453,
  'compl': 0.6060183251049304,
  'hmg': 0.8776859086268659,
  'n_clusters': 22,
  'noise': 2909,
  'v_score': 0.7169808270701487}),
({'min_c_size': 0.3, 'min_s_core': 1.5, 'n_c_size':
    211, 'n_s_core': 316},
 {'ami': 0.7141688463983411,
  'compl': 0.6125885376906761,
  'hmg': 0.8566678398366254,
  'n_clusters': 17,
  'noise': 3568,
  'v_score': 0.7143544276123411}),
({'min_c_size': 0.5, 'min_s_core': 0.1, 'n_c_size':
    352, 'n_s_core': 35},
 {'ami': 0.7189683890159466,
  'compl': 0.6118995297296262,
  'hmg': 0.8720518130494886,
  'n_clusters': 19,
  'noise': 3651,
  'v_score': 0.7191719552010638}),
({'min_c_size': 0.5, 'min_s_core': 0.2, 'n_c_size':
    352, 'n_s_core': 70},
 {'ami': 0.7312086957357227,
  'compl': 0.6229426637505312,
  'hmg': 0.8855637469446406,
  'n_clusters': 18,
  'noise': 3135,
  'v_score': 0.7313929003302995}),
({'min_c_size': 0.5, 'min_s_core': 0.3, 'n_c_size':
    352, 'n_s_core': 105},
 {'ami': 0.7265651397147384,
  'compl': 0.621988691648539,
  'hmg': 0.8739280320278963,
  'n_clusters': 17,
  'noise': 3628,
  'v_score': 0.7267427987570086}),
({'min_c_size': 0.5, 'min_s_core': 0.5, 'n_c_size':
    352, 'n_s_core': 176},
 {'ami': 0.7190442220170609,
  'compl': 0.61387876092512,
  'hmg': 0.8682213949598651,
  'n_clusters': 17,
  'noise': 2784,
  'v_score': 0.7192262574567897}),
({'min_c_size': 0.5, 'min_s_core': 1, 'n_c_size': 352,
    'n_s_core': 352},
 {'ami': 0.7148466716380937,
  'compl': 0.6155954625658637,
  'hmg': 0.8527511312121718,
  'n_clusters': 16,
  'noise': 3913,
  'v_score': 0.7150215477688168}),
({'min_c_size': 0.5, 'min_s_core': 1.5, 'n_c_size':
    352, 'n_s_core': 528},
 {'ami': 0.722436291401929,
  'compl': 0.6281257473051989,
  'hmg': 0.8504852034454239,
  'n_clusters': 14,
  'noise': 3977,
  'v_score': 0.7225858211248554}),
({'min_c_size': 1.2, 'min_s_core': 0.1, 'n_c_size':
    846, 'n_s_core': 84},
 {'ami': 0.7519655391288909,
  'compl': 0.6646554933311745,
  'hmg': 0.8659613899984694,
  'n_clusters': 11,
  'noise': 3400,
  'v_score': 0.7520706208638162}),
({'min_c_size': 1.2, 'min_s_core': 0.2, 'n_c_size':
    846, 'n_s_core': 169},
 {'ami': 0.751972356797838,
  'compl': 0.6645793919670121,
  'hmg': 0.8661086400343864,
  'n_clusters': 11,
  'noise': 3381,
  'v_score': 0.7520774205294138}),
({'min_c_size': 1.2, 'min_s_core': 0.3, 'n_c_size':
    846, 'n_s_core': 253},
 {'ami': 0.7464418117860208,
  'compl': 0.6601777828646404,
  'hmg': 0.8589227621167591,
  'n_clusters': 11,
  'noise': 3693,
  'v_score': 0.7465493006628577}),
({'min_c_size': 1.2, 'min_s_core': 0.5, 'n_c_size':
    846, 'n_s_core': 423},
 {'ami': 0.7369709503307816,
  'compl': 0.6584545384024955,
  'hmg': 0.8370093417555682,
  'n_clusters': 10,
  'noise': 5031,
  'v_score': 0.7370724322756455}),
({'min_c_size': 1.2, 'min_s_core': 1, 'n_c_size': 846,
    'n_s_core': 846},
 {'ami': 0.7153361390657719,
  'compl': 0.6379095170862578,
  'hmg': 0.814438375669607,
  'n_clusters': 10,
  'noise': 4871,
  'v_score': 0.7154456497803421}),
({'min_c_size': 1.2, 'min_s_core': 1.5, 'n_c_size':
    846, 'n_s_core': 1269},
 {'ami': 0.7005355619637456,
  'compl': 0.6381927348100297,
  'hmg': 0.7766018622448297,
  'n_clusters': 8,
  'noise': 7225,
  'v_score': 0.7006270272113172}),
({'min_c_size': 1.3, 'min_s_core': 0.1, 'n_c_size':
    917, 'n_s_core': 91},
 {'ami': 0.7506194135200529,
  'compl': 0.6757028176046485,
  'hmg': 0.8444376810754944,
  'n_clusters': 9,
  'noise': 5146,
  'v_score': 0.7507055050367504}),
({'min_c_size': 1.3, 'min_s_core': 0.2, 'n_c_size':
    917, 'n_s_core': 183},
 {'ami': 0.7473202990051494,
  'compl': 0.6734591340198 19,
  'hmg': 0.8395989749590475,
  'n_clusters': 9,
  'noise': 5323,
  'v_score': 0.7474076444842549}),
({'min_c_size': 1.3, 'min_s_core': 0.3, 'n_c_size':
    917, 'n_s_core': 275},
 {'ami': 0.7442049502967567,
  'compl': 0.6705928316945415,
  'hmg': 0.8361940491673147,
  'n_clusters': 9,
  'noise': 5398,
  'v_score': 0.7442933601286696}),
({'min_c_size': 1.3, 'min_s_core': 0.5, 'n_c_size':
    917, 'n_s_core': 458},
 {'ami': 0.736213631877689,
  'compl': 0.6646355754379383,
  'hmg': 0.8252993714583654,
  'n_clusters': 9,
  'noise': 5903,
  'v_score': 0.7363050632518328}),
({'min_c_size': 1.3, 'min_s_core': 1, 'n_c_size': 917,
    'n_s_core': 917},
 {'ami': 0.7192457671914706,
  'compl': 0.6478123837420122,
  'hmg': 0.8086303527844301,
  'n_clusters': 9,
  'noise': 5872,
  'v_score': 0.7193427427881789}),
({'min_c_size': 1.3, 'min_s_core': 1.5, 'n_c_size':
    917, 'n_s_core': 1375},
 {'ami': 0.6942606672605046,
  'compl': 0.6334500724429734,
  'hmg': 0.7682157214074672,
  'n_clusters': 8,
  'noise': 7992,
  'v_score': 0.6943542555042401}),
({'min_c_size': 1.6, 'min_s_core': 0.1, 'n_c_size':
    1128, 'n_s_core': 112},
 {'ami': 0.7497635627642752,
  'compl': 0.6749673186736733,
  'hmg': 0.8434213203092054,
  'n_clusters': 9,
  'noise': 5178,
  'v_score': 0.7498499560200317}),
({'min_c_size': 1.6, 'min_s_core': 0.2, 'n_c_size':
    1128, 'n_s_core': 225},
 {'ami': 0.737706730007799,
  'compl': 0.6729171535330699,
  'hmg': 0.8164981747157655,
  'n_clusters': 8,
  'noise': 6455,
  'v_score': 0.737786992216165}),
({'min_c_size': 1.6, 'min_s_core': 0.3, 'n_c_size':
    1128, 'n_s_core': 338},
 {'ami': 0.7380211046835529,
  'compl': 0.6730289235301904,
  'hmg': 0.8171035161560135,
  'n_clusters': 8,
  'noise': 6408,
  'v_score': 0.73810123884964 47}),
({'min_c_size': 1.6, 'min_s_core': 0.5, 'n_c_size':
    1128, 'n_s_core': 564},
 {'ami': 0.7185146951925302,
  'compl': 0.6552692463257285,
  'hmg': 0.7954840817412223,
  'n_clusters': 8,
  'noise': 6021,
  'v_score': 0.718600805005535}),
({'min_c_size': 1.6, 'min_s_core': 1, 'n_c_size': 1128,
    'n_s_core': 1128},
 {'ami': 0.7022810565645032,
  'compl': 0.6409864676575583,
  'hmg': 0.7767607968997089,
  'n_clusters': 8,
  'noise': 7663,
  'v_score': 0.70237223779810 45}),
({'min_c_size': 1.6, 'min_s_core': 1.5, 'n_c_size':
    1128, 'n_s_core': 1692},
 {'ami': 0.6956296729799442,
  'compl': 0.6853842363982453,
  'hmg': 0.7063354099157049,
  'n_clusters': 6,
  'noise': 15876,
  'v_score': 0.695702121973076}),
({'min_c_size': 2, 'min_s_core': 0.1, 'n_c_size': 1410,
    'n_s_core': 141},
```

```
  {'ami': 0.7401844424671706,
   'compl': 0.6751055426520609,
   'hmg': 0.8193435841455243,
   'n_clusters': 8,
   'noise': 6350,
   'v_score': 0.7402639341472464}),
 ({'min_c_size': 2, 'min_s_core': 0.2, 'n_c_size': 1410,
      'n_s_core': 282},
  {'ami': 0.7378700376111559,
   'compl': 0.6729052704568231,
   'hmg': 0.8169156233011354,
   'n_clusters': 8,
   'noise': 6416,
   'v_score': 0.7379502203801983}),
 ({'min_c_size': 2, 'min_s_core': 0.3, 'n_c_size': 1410,
      'n_s_core': 423},
  {'ami': 0.7288707563630125,
   'compl': 0.6661316640985748,
   'hmg': 0.8048595613394327,
   'n_clusters': 8,
   'noise': 6904,
   'v_score': 0.7289539593290797}),
 ({'min_c_size': 2, 'min_s_core': 0.5, 'n_c_size': 1410,
      'n_s_core': 705},
  {'ami': 0.7162476620275151,
   'compl': 0.6530519507117124,
   'hmg': 0.793197385834811,
   'n_clusters': 8,
   'noise': 6208,
   'v_score': 0.7163344342211092}),
 ({'min_c_size': 2, 'min_s_core': 1, 'n_c_size': 1410, '
     n_s_core': 1410},
  {'ami': 0.6895864549428805,
   'compl': 0.6297382230683 79,
   'hmg': 0.7622372192191372,
   'n_clusters': 8,
   'noise': 8653,
   'v_score': 0.6896815811618243}),
 ({'min_c_size': 2, 'min_s_core': 1.5, 'n_c_size': 1410,
      'n_s_core': 2115},
  {'ami': 0.6358398021197752,
   'compl': 0.786300764674153,
   'hmg': 0.5338000521555784,
   'n_clusters': 4,
   'noise': 4580,
   'v_score': 0.635902021788039})]
```

B.2 Clustering performance on less represented attacks

We report below the outputs from clustering on the less represented attacks of mirai UDP flooding, and bruteforce cases.

In the first listing below, brute-force attacks have been left separated in *attacker* and *victim* scenarios.

```
{'MIRAI-HOSTBRUTEFORCE-ATTACKER':{
    'losses': 33.52007469654529,
    'matches': 62.582010582010575
    },
 'MIRAI-HOSTBRUTEFORCE-VICTIM': {
    'losses': 35.63083566760038,
    'matches': 76.19047619047619
    },
 'MIRAI-UDPFLOODING': {
    'losses': 38.23529411764706,
    'matches': 100.0}
}

Average matches: 79.590828924162 27
Average losses: 35.7954014939309
Clusters purity: 84.58937198067632

Clusters {label: [majority label percentage, majority
     label]}
{'-1': [0, ''],
 '0': [100.0, 'MIRAI-UDPFLOODING'],
 '1': [100.0, 'MIRAI-UDPFLOODING'],
 '10': [60.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '11': [100.0, 'MIRAI-UDPFLOODING'],
 '12': [100.0, 'MIRAI-UDPFLOODING'],
 '13': [58.333333333333336, 'MIRAI-HOSTBRUTEFORCE-
     ATTACKER'],
 '14': [100.0, 'MIRAI-UDPFLOODING'],
 '15': [100.0, 'MIRAI-UDPFLOODING'],
 '16': [100.0, 'MIRAI-UDPFLOODING'],
 '17': [100.0, 'MIRAI-UDPFLOODING'],
 '18': [100.0, 'MIRAI-UDPFLOODING'],
 '19': [100.0, 'MIRAI-UDPFLOODING'],
 '2': [100.0, 'MIRAI-UDPFLOODING'],
 '20': [50.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '21': [64.28571428571429, 'MIRAI-HOSTBRUTEFORCE-
     ATTACKER'],
 '22': [100.0, 'MIRAI-UDPFLOODING'],
 '23': [100.0, 'MIRAI-UDPFLOODING'],
 '24': [100.0, 'MIRAI-UDPFLOODING'],
 '25': [100.0, 'MIRAI-UDPFLOODING'],
 '26': [100.0, 'MIRAI-UDPFLOODING'],
 '27': [100.0, 'MIRAI-UDPFLOODING'],
 '28': [100.0, 'MIRAI-UDPFLOODING'],
 '29': [100.0, 'MIRAI-UDPFLOODING'],
 '3': [50.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '30': [100.0, 'MIRAI-UDPFLOODING'],
 '31': [100.0, 'MIRAI-UDPFLOODING'],
 '32': [100.0, 'MIRAI-UDPFLOODING'],
 '33': [100.0, 'MIRAI-UDPFLOODING'],
 '34': [100.0, 'MIRAI-UDPFLOODING'],
 '35': [77.77777777777777, 'MIRAI-HOSTBRUTEFORCE-
     ATTACKER'],
 '36': [80.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '37': [75.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '38': [85.71428571428571, 'MIRAI-HOSTBRUTEFORCE-VICTIM'
     ],
 '39': [60.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '4': [50.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '40': [66.66666666666667, 'MIRAI-HOSTBRUTEFORCE-
     ATTACKER'],
 '41': [66.66666666666667, 'MIRAI-HOSTBRUTEFORCE-VICTIM'
     ],
 '42': [66.66666666666667, 'MIRAI-HOSTBRUTEFORCE-
     ATTACKER'],
 '43': [80.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '44': [50.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '5': [50.0, 'MIRAI-HOSTBRUTEFORCE-ATTACKER'],
 '6': [100.0, 'MIRAI-UDPFLOODING'],
 '7': [100.0, 'MIRAI-UDPFLOODING'],
 '8': [100.0, 'MIRAI-UDPFLOODING'],
 '9': [100.0, 'MIRAI-UDPFLOODING']}
```

In this second listing, victim and attacker brute-force scenarios were merged into one brute-force label.

```
{'MIRAI-BRUTEFORCE': {
    'losses': 61.76470588235294,
    'matches': 100.0
    },
 'MIRAI-UDPFLOODING': {
    'losses': 38.23529411764706,
    'matches': 100.0}
    }
Average matches: 100.0
Average losses: 50.0
Clusters purity: 97.82608695652173

Clusters {label: [majority label percentage, majority
     label]}
{'-1': [0, ''],
 '0': [100.0, 'MIRAI-UDPFLOODING'],
 '1': [100.0, 'MIRAI-UDPFLOODING'],
 '10': [100.0, 'MIRAI-BRUTEFORCE'],
 '11': [100.0, 'MIRAI-UDPFLOODING'],
 '12': [100.0, 'MIRAI-UDPFLOODING'],
 '13': [100.0, 'MIRAI-BRUTEFORCE'],
 '14': [100.0, 'MIRAI-UDPFLOODING'],
 '15': [100.0, 'MIRAI-UDPFLOODING'],
 '16': [100.0, 'MIRAI-UDPFLOODING'],
 '17': [100.0, 'MIRAI-UDPFLOODING'],
 '18': [100.0, 'MIRAI-UDPFLOODING'],
 '19': [100.0, 'MIRAI-UDPFLOODING'],
 '2': [100.0, 'MIRAI-UDPFLOODING'],
 '20': [100.0, 'MIRAI-BRUTEFORCE'],
 '21': [100.0, 'MIRAI-BRUTEFORCE'],
 '22': [100.0, 'MIRAI-UDPFLOODING'],
 '23': [100.0, 'MIRAI-UDPFLOODING'],
 '24': [100.0, 'MIRAI-UDPFLOODING'],
 '25': [100.0, 'MIRAI-UDPFLOODING'],
 '26': [100.0, 'MIRAI-UDPFLOODING'],
 '27': [100.0, 'MIRAI-UDPFLOODING'],
 '28': [100.0, 'MIRAI-UDPFLOODING'],
 '29': [100.0, 'MIRAI-UDPFLOODING'],
 '3': [100.0, 'MIRAI-BRUTEFORCE'],
 '30': [100.0, 'MIRAI-UDPFLOODING'],
 '31': [100.0, 'MIRAI-UDPFLOODING'],
 '32': [100.0, 'MIRAI-UDPFLOODING'],
 '33': [100.0, 'MIRAI-UDPFLOODING'],
 '34': [100.0, 'MIRAI-UDPFLOODING'],
 '35': [100.0, 'MIRAI-BRUTEFORCE'],
 '36': [100.0, 'MIRAI-BRUTEFORCE'],
 '37': [100.0, 'MIRAI-BRUTEFORCE'],
 '38': [100.0, 'MIRAI-BRUTEFORCE'],
 '39': [100.0, 'MIRAI-BRUTEFORCE'],
 '4': [100.0, 'MIRAI-BRUTEFORCE'],
 '40': [100.0, 'MIRAI-BRUTEFORCE'],
 '41': [100.0, 'MIRAI-BRUTEFORCE'],
 '42': [100.0, 'MIRAI-BRUTEFORCE'],
 '43': [100.0, 'MIRAI-BRUTEFORCE'],
 '44': [100.0, 'MIRAI-BRUTEFORCE'],
 '5': [100.0, 'MIRAI-BRUTEFORCE'],
 '6': [100.0, 'MIRAI-UDPFLOODING'],
 '7': [100.0, 'MIRAI-UDPFLOODING'],
 '8': [100.0, 'MIRAI-UDPFLOODING'],
 '9': [100.0, 'MIRAI-UDPFLOODING']}
```

Despite the large number of clusters generated, attacks are correctly isolated in respective clusters.

This can be especially observed in the second listing. The clustering algorithm thus discerns attacks with appreciable results. In particular, even if the amount of produced clusters is not optimal, the results still represent a characterisation that will be consistently replicated for similar anomalous traffic captured. With these considerations, and timing limitations, we did not fine-tune the clustering further at the current stage of the project.

C MRT artifacts

C.1 Device metadata example

Below, a JSON file listing the device-deployment metadata for the TP-Link plug at the University of Twente Federated Laboratory node.

```
{
    "device_info" : {
        "device_id" : "ut-tplink-plug",
        "device_vendor" : "KASA-TPLink",
        "device_type" : "power-plug"
    },
    "deployment_info" : {
        "deployment_id" : "UT_SCS_IoT_fedlab",
        "lat" : 52.239210,
        "lon" : 6.856720,
        "country" : "The_Netherlands",
        "industry_sector" : "Academic"
    }
}
```

C.2 Cluster evolution transition metrics

Proxy transition metadata.

- centroids_dimension : int (= n);
- ch1_t_start : timestamp. Earliest timestamp in the first capture;
- ch1_t_end : timestamp. Latest timestamp in the first capture;
- ch2_t_start : timestamp. Earliest timestamp in the first capture;
- ch2_t_end : timestamp. Latest timestamp in the second capture;
- elapsed_transition_time: int. ch2_t_start – ch1_t_end, can be negative in case time windows overlap;
- ch1_tot_flows : int. Number of MRT flow entries in characterization 1;
- ch2_tot_flows : int. Number of MRT flow entries in characterization 2
- tot_flows_balance : int. Positive or negative integer indicating the variation in the number of MRT flows from the first to the second capture;
- clusters_balance : int. Positive or negative integer indicating the variation in the number of clusters from the first to the second capture;
- noise_balance : float[-1, 1]. Variation in the percentage value of the size of the 'noise' set output by the clustering algorithm;
- all_dists_avg : float. Average value over all meta-centroids distances from first to second capture;
- all_dists_std : float. Numpy standard deviation (changes deg of freedoms) over all meta-centroid distances from first to second capture;
- all_dists_deciles : float. Deciles of the distribution of all meta-centroid distances from first to second capture.

Mutual matches.

- mutual_matches_n : int. Number of mutually matching clusters from first to second capture;
- mutual_matches_percentage : float[0, 1]. Percentage of mutually matching clusters from first to second capture over all matches cases;
- mutual_vects_avg : n-dimensional floats list. Average distance vector for mutually matching clusters;
- mutual_vects_std: n-dimensional floats list. Numpy standard deviation of distance vectors for mutually matching clusters;
- mutual_ vects_decile_1, ..., 10: list of 10 n-dimensional floats lists. Deciles vectors over the distribution of distance vectors for mutually matching clusters.

Forward matches. Description of non-explained entries below are the same as for mutual matches, but referred to respectively forward, and backward matching cases.

- fwd_matches_n : int;
- fwd_matches_percentage : float[0, 1];
- fwd_matches_agglomeration_avg : float. Average number of forward matches that a column of the second characterization (the 'destination' cluster) produces. In other words, how often a column cluster is matched as the closest by different row clusters;
- fwd_matches_agglomeration_std : float. Standard deviation over the number of forward matches that a column cluster has;
- fwd_matches_agglomeration_max : float. Maximum number of forward matches for a given column cluster;
- fwd_matches_agglomeration_max_percentage : float[0, 1]. Percentage of forward matches on the maximally forward-matched column cluster, over all forward matches. If this number is

high, it means that the most of forward matches interest one single cluster. If the cluster balance is ¡= 0, this could mean that some clusters have merged. If cluster balance is ¿ 0, it could mean that a new unrecognized traffic event is taking place;
- fwd_matches_agglomeration_max_col_cluster : int. Cluster id at column that receives the most forward matches;
- fwd_vects_avg : n-dimensional floats list;
- fwd_vects_std : n-dimensional floats list;
- fwd_vects_decile_1, . . . , 10: ten decile vector features, each a n-dimensional floats lists.

Backward matches. Description of non-explained cases below are the same as for forward matches, though, agglomeration values are referred to row clusters instead of column clusters.

- bwd_matches_n : int;
- bwd_matches_percentage : float[0, 1];
- bwd_matches_agglomeration_avg : float;
- bwd_matches_agglomeration_std : float;
- bwd_matches_agglomeration_max : float;
- bwd_matches_agglomeration_max_percentage : float[0, 1]. Explanation is symmetrical to above, but to report an example for clarity: if a row shows high backward matches, it signifies that many 'destination' clusters match back to that one 'origin' cluster, which in turn, may represent a clusters split having occurred;
- bwd_matches_agglomeration_max_row_cluster : int;
- bwd_vects_avg : n-dimensional floats list;
- bwd_vects_std : n-dimensional floats list;
- bwd_ vects_decile_1, . . . , 10 : ten decile vector features, each a n-dimensional floats lists.

C.3 MRT feed example

A CSV file can be found at https://mega.nz/file/ZgtzFKKb#FNtMRfgfNvqmA4tjpE8ZyPq1BjSTAsbA_Kkniy_D45k. It shows the MRT feed for the UTwente TP-link plug, output of our first experiment with the Federated Laboratory.

D Complete results

D.1 On Kang's dataset

All of the per-feature correlation outputs can be consulted at https://mega.nz/folder/h80zxKKC#32gWaPYTtvqtVHTz9WuCrw. The `same-events` folder lists the results for when both devices are targeted by the same sequence of attacks; `diff-events`, instead, lists the results for the devices attacked with two different sequences of attacks. Results are divided per mono and multi-dimensional features.

D.2 Experiments

At https://mega.nz/folder/U891BKYa#orGVTd93Amik7KZZA0br5w, we report the complete results for each of the four experiments we run. Experiments results are divided for mono and multi-dimensional features. For the mono-dimensional features, we show separately results for attacked, non-attacked, and all four devices. For the multi-dimensional features, we only report results for attacked devices.