Stepping out of the MUD: Contextual threat information for IoT devices with manufacturer-provided behavior profiles

Luca Morgese Zangrandi
luca.morgese@tno.nl
TNO & University of Twente

Thijs van Ede
t.s.vanede@utwente.nl
University of Twente

Tim Booij
tim.booij@tno.nl
TNO

Savio Sciancalepore
s.sciancalepore@tue.nl
Eindhoven University of Technology

Luca Allodi
l.allodi@tue.nl
Eindhoven University of Technology

Andrea Continella
a.continella@utwente.nl
University of Twente

ABSTRACT

Besides coming with unprecedented benefits, the Internet of Things (IoT) suffers deficits in security measures, leading to attacks increasing every year. In particular, network environments such as smart homes lack managed security capabilities to detect IoT-related attacks; IoT devices hosted therein are thus more easily targeted by threats. As such, context awareness of IoT infections is hard to achieve, preventing prompt response. In this work, we propose MUDscope, an approach to monitor malicious network activities affecting IoT systems in real-world consumer environments. We leverage the recent Manufacturer Usage Description (MUD) specification, which defines networking whitelists for IoT devices in MUD profiles, to reflect consistent and necessarily-anomalous activities from smart things. Our approach characterizes this traffic and extracts signatures for given attacks. By analyzing attack signatures for multiple devices, we gather insights into emerging attack patterns. We evaluate our approach on both an existing dataset and a new, openly available dataset created for this research. We show that MUDscope detects several attacks targeting IoT devices with an F1-score of 95.77% and correctly identifies signatures for specific attacks with an F1-score of 87.72%.

ACM Reference Format:
Luca Morgese Zangrandi, Thijs van Ede, Tim Booij, Savio Sciancalepore, Luca Allodi, and Andrea Continella. 2022. Stepping out of the MUD: Contextual threat information for IoT devices with manufacturer-provided behavior profiles. In *Annual Computer Security Applications Conference (ACSAC '22), December 5–9, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3564625.3564644

1 INTRODUCTION

The Internet of Things (IoT) refers to a cyber-physical ecosystem of interconnected devices (things), which exchange and process data to enable intelligent decision-making [10]. IoT adoption is in continuous growth, as leading agencies forecast the number of Internet-connected devices to quadruple from 9 billion in 2020 to 38 billion in 2030 [23]. Unfortunately, increasing IoT adoption leads to an increased attack surface. In 2020 only, the number of IoT-directed attacks increased by 800% compared to the previous year [36], while the first half of 2021 saw one and a half billion attacks on smart devices, more than twice than in the same period in 2020 [59].

Research shows that a considerable share of compromised IoT devices is located within consumer environments (Internet Service Providers (ISPs) and telecommunications Internet domains [42, 65]), pointing to IoT environments that we refer to as loosely-protected: households, shops, restoration, open-access Wi-Fi areas, and alike. Attackers actively infiltrate in these environments [62] where compromised IoT devices can proliferate largely undetected by the defenders' community.

To improve detection, research and industry stress the need to enhance continuous collection and sharing of actionable IoT security intelligence, e.g., indicators of compromise, IoT vulnerabilities, attackers' goals and trends [8, 13, 42, 55, 68]. This information allows faster, more precise, and better-informed security interventions on both network administrators' and device vendors' sides [24, 42, 49, 51], making IoT integration and use safer.

Three main state-of-art approaches exist to collect and share security intelligence: (IoT) honeypots, network telescopes, and Threat Intelligence (TI) feeds. Each, though, has limitations with respect to the visibility of real-world loosely-protected IoT environments. IoT honeypots mimic vulnerable devices to study how attackers interact with them [27, 62, 72]. However, they can be fingerprinted by attackers, and do not directly represent how anomalies spread through real-world consumer deployments [12, 62, 69]. Network telescopes (or, "darknets") are portions of routable IP addresses that do not host any service, so all traffic that they receive is anomalous by definition [39]. The insights that network telescopes derive are biased toward Internet-wide activities [31, 42, 53, 60], so they cannot detect targeted IoT attacks [51]. Crowd-sourced threat intelligence feeds such as AlienVaultOTX, Censys, and Shodan [3, 5, 61], when IoT-specific, can be highly inconsistent in reporting timings and coverage of attacks [4, 15, 31]. Other approaches such as DShield [26] collect firewall logs from multiple deployments and produce large-scale information on network compromise attempts. However, they are not IoT- nor deployment-specific, and require additional configuration at the deployments' end, making them not suitable for an average consumer [29].

In summary, we identify a gap in the state-of-art about accessing and leveraging threat intelligence on IoT network threats that target consumer environments. Motivated by this, this work presents MUDscope, a tool implementing a novel approach to monitor and

ACSAC '22, December 5–9, 2022, Austin, TX, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9759-9/22/12.
https://doi.org/10.1145/3564625.3564644

analyze malicious IoT traffic in consumer environments. Asserting on a timely basis what specific devices and deployments are targeted by similar or different network anomalies, MUDscope has the aim to provide contextual information on how threats spread through IoT deployments.

Our approach leverages the recent IETF [25] Manufacturer Usage Description (MUD) specification [30]. MUD allows vendors to whitelist traffic for their devices, ensuring that any non-whitelisted traffic can safely be rejected. MUD is an effort to improve the security of IoT with a specification-based approach. Previous work has shown the effectiveness of MUD profiles to filter malicious IoT traffic [19–22]. In brief, all traffic that is not whitelisted in MUD profiles is thus necessarily anomalous.

Our approach builds on MUD profiles and uses the specification in a novel way to analyze MUD-rejected traffic (MRT) and generate previously-unavailable threat intelligence. For a specific device in a deployment, we monitor its MRT, we cluster rejected traffic flows showing similar features, and we describe how these clusters evolve in time. The evolution of these clusters provides a signature for anomalous activities related to an IoT device. Similar signatures represent similar malicious activities from a network threat. We compare signatures collected from multiple devices to gather insights on IoT network anomalies and how they spread through loosely-protected environments.

We validate MUDscope with six IoT devices from different categories and brands. We target them with different scanning and Denial of Service (DoS) attacks, collect and generate signatures of MUD-rejected traffic, and show that we can detect when similar or different network threats reach multiple devices. This information can be provided to manufacturers of targeted IoT products and deployments, and to the research community, prompting investigation and response.

In summary, we make the following contributions:

- We propose a new approach for monitoring IoT-related traffic that leverages the MUD specification to reveal malicious connections. This operates as an IoT-specialized network telescope requiring minimal configuration.
- We implement our approach in MUDscope, a tool that generates attack signatures describing the behavior of clustered anomalous network flows over time.
- We evaluate MUDscope on our new, open-source, dataset containing multiple IoT devices from different categories and brands. Our experiments show that MUDscope can characterize malicious network traffic with an F1-score of 87.72%.

In the spirit of open science, we make MUDscope open-source [40], as well as the datasets generated and used for our evaluation [41].

2 THREAT MODEL

Cyber attackers are actively engaged in compromising exposed IoT devices [43, 59, 62]. Of the targeted devices, over half are deployed within ISPs and telecommunications sectors [42, 65]. This points to devices deployed in loosely-protected environments such as households and home offices. These environments lack managed security capabilities such as intrusion detection systems and security operation centers. The devices that they host are thus more easily reached by network threats [66].

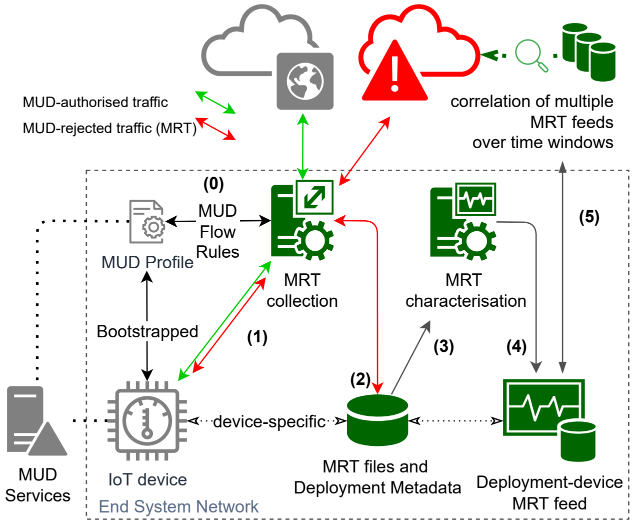

Figure 1: High-level schematic for the proposed MUD usage. (0) The local environment retrieves MUD profiles and flow rules; (1) we enforce MUD rules on device-related traffic; (2) we save the MUD-Rejected Traffic (MRT); (3) we describe how the MRT evolves through time; (4) we save the description in an MRT feed; (5) we compare multiple MRT feeds to observe fluctuations for different IoT devices.

In this threat model, MUDscope is designed to detect anomalous IoT network activities affecting devices in loosely-protected environments. We focus our research on consumer IoT devices (e.g., IP cameras, motion sensors, smart appliances, smart plugs, etc.) that communicate over the Internet via UDP/TCP-IP stack. We address external network threats, such as botnets, that attempt to gain control of IoT devices via reconnaissance (active scanning) and initial access tactics, as per MITRE ATT&CK kill-chain [38].

The MUD specification is still not widely adopted by vendors, though active engagement from standardization bodies, industry, and research is manifested [44]. In our research, we assume that MUD profiles are available, and readily deployable as OpenFlow rules [46] (forwarding rules for the allowed UDP/TCP flows associated to the MUD rules), as per indication of the MUD specification. To satisfy this assumption, we rely on MUDgee [22] to create MUD profiles. Finally, we assume that the integrity of deployed MUD profiles has been verified by a trusted component.

3 METHODOLOGY

Figure 1 presents a high-level scheme of our approach. OpenFlow rules derived from a MUD profile are available in the local environment (step 0). In step 1, we collect device-specific packets rejected by the MUD rules, namely, the MUD-Rejected Traffic (MRT). In step 2, we record MRT packet logs divided in time windows. In step 3, we describe anomalous traffic at each time window: we collect traffic flows and cluster them together, obtaining a high-level representation of anomalies through that given time window. In step 4, we create traffic signatures by describing how clusters of anomalous flows change through time windows. We refer to

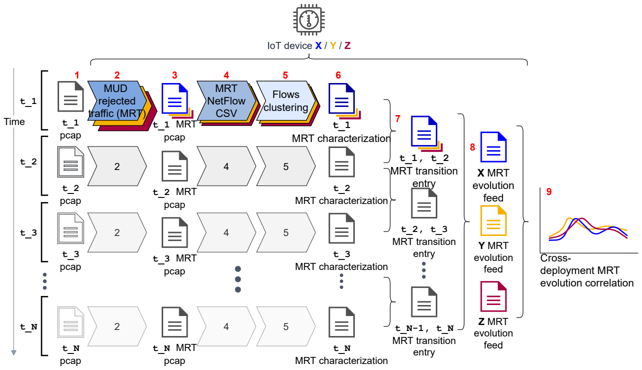

Figure 2: Pipeline for our MUD usage proposal. The sequence of steps of our pipeline is detailed for the time window t_1. One color codifies one device.

these descriptions as MRT feeds. These feeds contain signatures of recorded anomalous activities. Finally, in step 5, we compare signatures from multiple MRT feeds to detect emerging similar attack patterns.

Note that all the steps in the pipeline are specific to *one* device in *one* deployment, up to the comparison of traffic signatures from multiple devices. We explain these steps referring to Figure 2, presenting the pipeline of our approach in greater detail.

3.1 MUD profiles enforcement

Our approach starts from MUD profiles available for devices within a loosely-protected environment. In accordance with the MUD specification, the whitelist rules in a MUD profile have been translated into actionable networking rules [30] — in our case, an OpenFlow table. OpenFlow is a protocol to control packet forwarding in switches or routers in a software-defined (i.e., programmable) network [7], and is often coupled with IoT [63]. MUD rules are therefore enforced at local level via network flow rules.

Throughout a time window t_a, we filter device-specific network packets against the device’s MUD profile rules. Non-matching packets are appended to a network log (a pcap) file. We then convert the rejected packets in t_a to netflow [6] flows format to constrain the size of the MRT logs. These operations correspond to steps 1, 2 and 3 in Figure 2.

3.2 Time window anomalies characterization

MUD-Rejected Traffic is anomalous by manufacturer definition. In general, we expect that collected MRT flows belong to certain categories of anomalous traffic, such as host-discovery probes, targeted scans, credentials brute-forcing, DoS traffic, advertisement-related probes, Internet noise [51]. Moreover, different threat actors may attempt to compromise new hosts using signature routines [16, 17]. For instance, MIRAI targets a victim device via (1) scanning TCP ports 23 and 2323, (2) brute-forcing Telnet credentials, (3) injecting code upon access to the device [2]. Our approach is primarily aimed at detecting when several devices are targeted by similar anomalies underlying similar attack routines.

We make two observations: (i) the same type of malicious activities, observed through *one* time window t_a, will likely yield similar MRT profile; (ii) the same attack routine will yield a similar sequence of anomalous activities observed through *multiple* time windows.

We thus adopt the following approach to group together flows belonging to similar malicious traffic. We first conducted a preliminary analysis on a dataset for IoT network intrusion detection (Kang et al. [28]) to select the flow features that most discern IoT network attacks. That the flow features (i) bytes-per-packet, (ii) TCP flags, (iii) input and (iv) output bytes, (v) destination port, and (vi) source-address category (private, public, reserved) proved good indicators to discern different IoT attack types (we report this analysis in Section 6.2). We use these features to cluster together similar anomalous flows observed through a window t_a (step 5 in Figure 2). We choose HDBSCAN [34] as clustering algorithm due to its time performance and generalizing approach [34, 35], which suits the dynamic and unknown nature of network traffic [18].

We describe the ensemble of observed anomalous flows in t_a by means of the characteristics of the clusters space, i.e., the number of observed clusters and their positions. Specifically, we describe the clusters’ positions with the list of their *meta-centroids*. A meta-centroid of a cluster is a point defined as follows: one dimension per average value of the flow feature of all points in the cluster; plus two *meta*-dimensions for average and standard deviation values of between-cluster points distances. A meta-centroid is thus $n + 2$-dimensional, with n = number of MRT flow features (six in our case). We add the last two dimensions to have a more robust notion of the identity of a cluster. For instance, in two different time windows, two clusters may appear in similarly-centered regions, but present flows with different features. The meta-dimensions allow us to capture this difference.

Finally, the MRT for a device observed through a window t_a is represented by the set of $(n+2)$-dimensional clusters meta-centroid points $C_a : \{c_i^a : i = 0, ..., |C_a|\}$ — a *characterization*, step 6 in Figure 2 — with c_i^a the meta-centroid of the cluster i, for the clustering performed in the time window a.

3.3 Anomaly signature

Thus far, we collected the MRT for a specific device, in *one* given time window. We then distinguished different clusters in this traffic, with the notion that each attack is represented by one or more clusters. We now want to describe how the positions of the clusters change through *successive* time windows to track how an attack evolves. The track of the changes in the clusters space constitutes the *signature* of an anomalous event.

We consider the general case of two successive characterizations: $C_a : \{c_i^a, i = 0, ..., |C_a|\}$, over t_a, and $C_b : \{c_j^b, j = 0, ..., |C_b|\}$, over t_b. We assume that t_a *precedes* t_b, i.e., the end-time of t_a comes before the start-time of t_b: $t_{a\_start} \leq t_{a\_end} \leq t_{b\_start} \leq t_{b\_end}$.

We compute the distance matrix M between pairwise clusters, where $M[i, j] = \text{dist}(c_i^a, c_j^b)$, $i = 0, ..., |C_a|$, $j = 0, ..., |C_b|$. Note that row indices correspond to clusters in C_a, and column indices to clusters in C_b. We observe what clusters appear closest in space across the two different time windows, and define three match cases that describe shifts, splits, and merges of clusters.

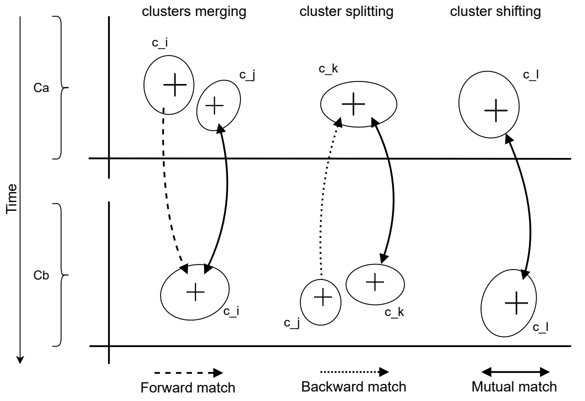

Figure 3: Conceptual illustration of changes in the anomaly clusters space through two time windows. The clusters space is simplified to two generic dimensions. We describe the dynamic behavior of anomalies by means of mutual, forward, and backward match cases.

The cluster $c_i^a \in C_a$ *forward-matches* with the cluster $c_j^b \in C_b$ if c_i^a is closest to c_j^b among all clusters in the *future* characterization C_b. Analytically, $m_{\text{fwd}}(c_i^a) = c_j^b \iff$

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

Similarly, a cluster 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 characterization C_a. We say $m_{\text{bwd}}(c_j^b) = c_i^a \iff$

$$c_i^a = \text{argmin}_{i=0,\dots,|C_a|}(\text{dist}(c_i^a, c_j^b)).$$

A *mutual match* between c_i^a and c_j^b, for fixed i and j, occurs when

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

Forward matches are only defined from C_a to C_b, and backward matches are only defined from C_b to C_a.

Figure 3 illustrates the clusters space in two consecutive characterizations. Clusters c_i^a, c_j^a, merge in c_i^b. They trigger one forward-match, and one mutual-match. Cluster c_k^a splits in two clusters c_j^b, c_k^b, and triggers one mutual-match and one backward-match. Cluster c_l^a shifts to the position of c_l^b, triggering a mutual-match.

The two distance matrices in Table 1 show two examples of clusters evolution, generated with our implementation (see Section 4). In the first matrix, we see a relatively stable evolution: just one new cluster (c_3^b) appears at a very short distance from the existing ones. In the second matrix, the high agglomeration of forward matches highlights a *merge* of clusters to c_3^a, c_5^a, and c_6^a, to c_3^b. We illustrate how network anomalies relate to these descriptions of clusters evolution in Section 3.3.2.

3.3.1 Signature definition. Using the three defined match cases we derive ten features to describe the evolution of the MRT through consecutive characterizations C_a and C_b: (1) balance of gained-to-lost clusters (`clusters balance`), to track the number of clusters observed over time; (2) average of all distances over the distance matrix of meta-centers (`all dists avg`), describing how spread clusters are; (3) `mutual matches n`, the number of mutual-match cases; similarly, (4) `backward matches n` and (5) `forward matches n`. We account for the respective percentage of match events of each type over all match events: (6) `mutual matches percentage`, (7) `backward matches percentage`, and (8) `forward matches percentage`, to record the relative share of each match type. Finally, (9) `bwd matches agglomeration avg` and (10) `fwd matches agglomeration avg` indicate over how many clusters on average forward and backward matches agglomerate, capturing the volume of splits and shifts of single clusters.

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

	Cb		
Ca	**0**	**1**	**2**
0	$\underline{3.019654}$	3.596038	3.825001
1	0.354194	2.373034	2.351585
2	2.495640	0.162561	0.352583
3	2.464904	0.412329	$\underline{0.247349}$
4	2.399984	0.330156	0.099545
5	2.564678	0.501293	$\underline{0.412108}$

Table 1: Two scenarios exemplifying cases of mutual, $\underline{\textbf{forward}}$, and $\boxed{\textbf{backward}}$ matches for clusters over consecutive time windows, extracted from our tool. Above, MUD-rejected traffic is relatively stable from C_a to C_b: only one backward match at a very close distance is recorded. Below, forward matches show merges of clusters in c_0^b and c_2^b.

We synthesize these features with the assumption that they achieve a high-level description of a transition between two MRT cluster spaces. We evaluate their effectiveness in Section 6.1.

For consecutive pairwise characterizations of MRT, we compute and write these features to MRT *transition entries* (step 7 in Figure 2). Each entry describes how anomalous traffic changes through two time windows. The sequence of MRT transition entries constitute an MRT feed (step 8 in Figure 2) spanning from the start-time of the first window to the end-time of the last window.

With this definition, the signature of an anomaly can be represented by a segment of an MRT feed. This is an $F = [N, M]$ matrix, where N is the number of MRT transition entries through which the anomaly spans, and M is the number of signature features considered in the signature. A column of $F[:, j]$ represents the values of the signature feature j through the time span where anomalous activity occurs. We denote the i^{th} signature in a feed X as F_i^X.

3.3.2 Expected behavior. If a device is subject to anomalous but harmless *noise* [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, and the percentage of mutual matches will stay close to 100%.

A *new* network event would generate new clusters, increasing the clusters balance, and the backward matches. The backward match agglomeration will also increase, because previous clusters are expected to mutually match. If an anomaly *ceases*, then the number of clusters will reduce in the next characterization, and forward matches will increase — corresponding to the previously existing clusters pointing ahead toward the remaining clusters.

3.4 Comparison of MRT feeds

A network threat using the same attacks to compromise different IoT devices will cause similar fluctuations in the MRT feeds of the respective devices. To detect this (step 9 in Figure 2), we perform the following three steps. First, we take the MRT feeds of the devices we are interested in monitoring, over arbitrary time periods. Second, we detect if and where anomalous traffic occurs in the feed, by observing if any MRT is captured in any time window in the first place. This is reflected in the signature feature reporting the clusters' balance through time windows. Doing so, we automatically gather the signature for any anomaly recorded in each feed. Third, we compare all anomaly signatures, pairwise. If the length of the signatures N is different, we compare the smaller signature with equally-sized sliding windows on the larger signature, and we record the values for the highest similarity found. In particular, for any two signatures F_n^X and F_m^Y, we compute the Pearson correlation coefficient r_j for each two column vectors of the same feature j, to observe if their values change in a similar way. The average value of r for all features is the *correlation* c of the two signatures, i.e., how similar is the trend of the anomalous activity captured between the two signatures. Analytically,

$$c(F_n^X, F_m^Y) = \frac{1}{M} \sum_{j \in \text{features}} \Big(r_j = r(F_n^X[:, j], F_m^Y[:, j]) \Big),$$

where M is the number of MRT feed signature features considered.

To capture the case where c returns a low value despite some signature features showing high correlation, we also record the maximum value of r over all features: $m(F_n^X, F_m^Y) = \max_{j \in \text{features}} r_j$. Additionally, to make our alerting method robust to comparing signatures from *different* attacks but with *similar* trends, we record the proportion p between ranges of the number of clusters generated by the two signatures, expressed as Ra_n^X and Ra_m^Y:

$$p(F_n^X, F_m^Y) = \frac{\min(\text{Ra}_n^X, \text{Ra}_m^Y)}{\max(\text{Ra}_n^X, \text{Ra}_m^Y)}.$$

We then use these values to compute the signature match value:

$$a(F_n^X, F_m^Y) = \text{avg}(c(F_n^X, F_m^Y),\ m(F_n^X, F_m^Y)) * \xi,$$

where ξ is a slack variable that constrains the value of the alert based on $p(F_n^X, F_m^Y)$, and is defined as follows:

$$\xi = p(F_n^X, F_m^Y) \text{ if } p(F_n^X, F_m^Y) \leq 0.5, \text{ else } 1.$$

A match alert is prompted when $a(F_n^X, F_m^Y)$ is greater than a given threshold, chosen to be 0.5, to capture the case where only one signature feature correlates perfectly. Further approaches can be investigated in follow-up work.

4 IMPLEMENTATION

For our proof-of-concept, we generate MUD profiles and related OpenFlow rules with MUDgee [22]. We collect local network traffic using Wireshark [71]. We divide such collected traffic in time window pcaps, according to our methodology, using `tcpdump` [64]. These pcaps are then fed to MUDscope, our tool. MUDscope is implemented in Python 3.8. We use Python's Scapy [56], a network packet manipulation library, to match device-specific packets against MUD flow rules, and generate the MRT pcap files. Packets files are aggregated into network flows in Comma-Separated Value (CSV) files, using `nfdump` [67].

From an MRT flow file, we select the attack-discriminating flow features (Sections 3.2 and 6.2), which we pre-process with the Sklearn library [58] to quantify and scale to apply clustering. We use the `HDBSCAN` python implementation [34] to produce the clustering characterization for each MRT flow file, as explained in Section 3.2.

For consecutive characterizations, we compute distance matrices among clusters' meta-centroids with Numpy's [45] euclidean distance function. From these matrices, we extract the MRT evolution signatures indicators presented in Section 3.3.1, mapping the sequential change of characterization in an MRT transition entry. We build an MRT feed CSV file by appending consecutive such entries.

Finally, we implement an MRT feed monitor module that ingests an arbitrary number of feeds, extracts anomaly signatures, and compares the similarities between signature metrics using Numpy's Pearson correlation index `corrcoef`. We recall that MUDscope is provided as open-source [40].

5 DATASETS

We evaluate our approach on both the existing IoT Network Intrusion Dataset by Kang et al. [28] and on our openly available MUDscope IoT dataset [41].

IoT Network Intrusion Dataset. Kang et al. [28] provide a dataset containing 42 raw network traces spanning 9 attacks and benign traces (2.99M packets) of two different smart home devices (a *Nugu* smart speaker and gan *EZVIZ* wi-fi camera) captured over a (non-continuous) period of 112 days. The attacks include scanning (host, port, and OS) using Nmap; flooding (SYN, UDP, ACK and HTTP) using custom scripts and MIRAI botnet attacks; and Host discovery and Telnet brute-force attacks using the MIRAI botnet. The original dataset also contains MITM ARP spoofing attacks, but are considered out of scope for this research. The IoT Network Intrusion Dataset is used for parameter optimization of our method (Sections 6.2 and 6.3). Appendix B gives an overview of the dataset.

MUDscope IoT dataset. We collect our own dataset (see Table 2), produced using the Federated IoT Laboratory [37]. In the dataset, we attack heterogeneous IoT devices deployed at the Eindhoven University of Technology, tagged *L1*. The deployed devices are: Eufy security homeBase 2 doorbell, Honeywell thermostat, two different Foscam IP cameras (models C1780P and RM2), and two Hombli smart plugs. We deploy an attacking computer in a different subnet at a second geographical location, TNO The Hague, tagged *L2*. The dataset consists of raw network traces for the six IoT devices containing both benign data and scanning attacks (Telnet SSH port scan, OS discovery, and Vulnerability scan) as well as targeted (hence only two devices) DoS TCP SYN-flood attacks.

Attack	Duration	# Flows	# Packets	# Devices
None (Benign)	1,462.93 s	627	76,055	5
Telnet/SSH port scan	1,663.82 s	424	4,815	5
OS scan	2,854.25 s	10,145	24,371	5
Vulnerability scan	3,815.83 s	3,091	14,801	5
TCP SYN flood DoS	2,131.20 s	135,472	395,771	2

Table 2: Details of MUDscope IoT dataset. For each attack scenario, we specify the capture duration, total number of flows, packets, and number of involved devices.

Device type	No. devices	Scans			DoS
		Telnet/SSH	OS	Vulnerabilities	SYN flood
Eufy HomeBase 2 doorbell	1	✗	✗	✗	
Honeywell Round T57RF2025 thermostat	1	✗	✗	✗	
Hombli smart plug HBPP-0201	2	✗	✗	✗	
Foscam C1780P IP camera	1	✗	✗	✗	✗
Foscam RM2 IP camera	1				✗

Table 3: Description of the devices utilized for the data collection, and the attacks performed on them.

Table 3 summarizes the devices and attack scenarios recorded in the dataset. Section 6.1.1 describes the procedure we used to generate the dataset. By capturing these various attacks over multiple devices in various locations and from different vendors, we attempt to show that our approach can correlate similar attacks in differing scenarios (Section 6.1).

6 EVALUATION

The main objective of our approach is to detect attacks on IoT devices and create respective signatures that can be used to identify emerging common attacks. Tables 4, 5, and 6 give an overview of our main results for detection and signature comparison (details in Section 6.1) over our MUDscope IoT dataset.

To obtain these results, we first used the Kang dataset to systematically analyze which features are most relevant for generating signatures (Section 6.2), and to perform an intermediate analysis on the clusters generated by our approach (Section 6.3).

In this Section, we report the main validation for MUDscope in 6.1. We then report the analyses performed to develop our tool, in Sections 6.2 and 6.3.

6.1 Signature matching

We aim to ascertain whether and when multiple IoT devices have been targeted by the same network threat. To this end, MUDscope correlates signature fluctuations of the MRT. To evaluate the extent to which these signatures are correctly correlated, we perform several experiments on six IoT devices. The network traces for the experiment we run constitute our MUDscope IoT dataset, as we describe in Section 5.

6.1.1 Experimental setup. We performed four experiments. In each of them, we start a packet capture at L1, then target a subset of the devices asynchronously with a given attack from L2, and stop the capture. The attacks that we launched are the following: (1) scan of Telnet and SSH ports, (2) nmap OS discovery scan, (3) a time-spread nmap vulnerabilities testing scan, and (4) TCP SYN flood DoS. For the three scanning experiments, we left the RM2 Foscam camera unharmed to have a control recording. In the targeted DoS experiment, we attacked both Foscam cameras, and left the other devices unharmed.

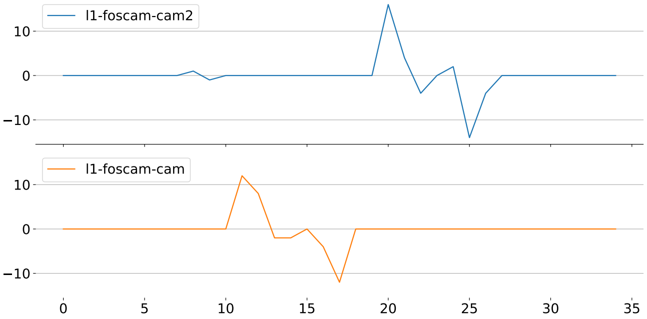

Figure 4: Anomaly signature fluctuations for the clusters balance feature, for the two Foscam cameras attacked with DoS traffic. The Y axes report the ranges of the clusters balance values. The X axis refers to the MRT transition entries.

We obtain one general capture per experiment, and process each with MUDscope, for each device's MUD profile. We so obtain four MRT feeds per device — one per experiment. We expect that each MRT feed contains the signature of the attack performed in the relative experiment. Next, for each experiment, we extract the signatures in each related MRT feed for each device, and verify if the malicious traffic was captured. Table 4 shows the detection performance based on signatures-finding in the MUDscope IoT dataset. As shown, we correctly identified all anomalies underlying attacks (100.00% recall) and incorrectly identified 9 anomalies (91.89% precision).

Next, we want to validate that signatures from the same attacks match with each other, but not with signatures from different attacks. Therefore, we run two sets of experiments. First, we check whether the signatures from the same attack correlate among all attacked devices. We report the overall results of this validation in Table 5. There, we see that although there is a complete overlap of all signatures for simple attacks, such as the Telnet/SSH port scan, more device-interactive attacks (e.g., OS scan) report some False Negatives (we explain below in Section 6.1.2).

Second, to validate that signatures from different attacks do not match, we perform six tests as follows: for each of five devices (all except the Foscam RM2), we compare all of its MRT feeds containing an attack. In the sixth test, we compare the MRT feeds from four different devices (one Hombli plug, the Honeywell thermostat, the Eufy HomeBase, and the Foscam RM2), each with a different attack. We report the overall results of this second validation in Table 6. We publish the complete set of outputs of our results at [40].

6.1.2 Same-attacks signatures matching. As shown in Table 6, we detect all Telnet and SSH port scans directed to each device, and find correlations between all signatures. Because the attack consists only of four flows, directed to ports 22, 23, 2222, and 2323, the produced fluctuations in the MRT feeds are minimal. We thus detect signature

Attack	MRT entries	Anomalous entries	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
Telnet/SSH port scan	130	15	15	115	0	0	100.00%	100.00%	100.00%	100.00%
OS scan	217	26	26	191	0	0	100.00%	100.00%	100.00%	100.00%
Vulnerability scan	310	48	45	259	3	0	98.06%	93.75%	100.00%	96.77%
TCP SYN flood DoS	170	22	16	142	6	0	96.47%	72.73%	100.00%	84.21%
Total	827	111	102	707	9	0	98.90%	91.89%	100.00%	95.77%

Table 4: Performance of anomaly detector over Novel IoT Dataset. We report the performance in terms of number of True/False Positive/Negative detected MRT entries as well as the Accuracy, Precision, Recall, and F1-score of our anomaly detector.

Attack	Total signatures	Expected matches	TP	TN*	FP	FN	Accuracy	Precision	Recall	F1-score
Telnet/SSH port scan	5	10	10	N/A*	0	0	100.00%	100.00%	100.00%	100.00%
OS scan	5	10	6	N/A*	0	4	60.00%	100.00%	60.00%	75.00%
Vulnerability scan	5	10	8	N/A*	0	2	80.00%	100.00%	80.00%	88.89%
TCP SYN flood DoS	2	1	1	N/A*	1	0	50.00%	50.00%	100.00%	66.67%
Total	17	31	25	N/A*	1	6	78.13%	96.15%	80.65%	87.72%

*This experiment compared found anomalies only of the same attack type, hence the True Negatives are 0. Table 6 shows the results when comparing signatures of different attacks.

Table 5: Evaluation matching attack signatures. We show for each attack, whether our signature matching algorithm found a match between signatures of the same attack.

Test	Device(s)	Compared MRT feeds	Incorrect matches		
			Expected	Worst	Found
1	Eufy home-kit doorbell	Scans (Telnet/SSH, OS, Vulnerabilities)	0	3	0
2	Honeywell thermostat		0	3	0
3	Hombli plug 1		0	3	0
4	Hombli plug 2		0	3	0
5	Foscam camera C1780P	All	0	6	1
6	Eufy, Honeywell, Hombli, Foscam	All	0	6	0
Overall			0	18	1
Matches correctly discarded					**94.44%**

Table 6: Summary of the tests executed for the signature-matching for different attacks. We expect no matches at each test, because we are comparing signatures from different attacks. In the worst case, we find that all signatures match with one another. For these tests, our method proved effective in not raising match alerts for different attacks.

matches only by means of the `clusters balance` metric, which shows perfect correlation for this attack, while values for the other nine metrics in the extracted signatures are flat. The non-attacked RM2 Foscam camera does not record anomalies, as expected.

For the OS scan attack, we detect all anomalous flows. We verified that recorded false-positive anomalies are caused simply by benign packets that the MUDgee tool failed to include in the generated MUD profiles. We find all signature matches between all devices except the Foscam camera C1780P, thus returning six out of ten matches. This occurs because, differently from the other four devices, the Foscam camera engaged minimally with the OS scan, thus its MRT produced a different signature. This provides the insight that anomaly signatures for the same attack can vary depending on how the devices react, and can thus be device-specific.

We detect all anomalies in the slow vulnerability scan experiment. False positive anomalies appear for the same reason as above. We detect eight out of ten matches, leaving out the match among Eufy and Honeywell, and Honeywell and one of the two Hombli plugs. Interestingly, all matching signatures correlate poorly (with an overall average of 0.33), but we are still able to detect the matches with our method, because of high maximum correlation values (0.9 on average), and comparable magnitude of generated clusters.

We detect both DoS attacks in the fourth experiment. Again, recorded false-positives are due to benign packets that were not included in the MUD profile, and barely generate a match on their signatures. Most importantly, the signatures of the two DoS attacks are strongly correlated over all signature features (average of 0.86). Figure 4 depicts the plot for the fluctuations of the `clusters balance` signature feature for the two Foscam cameras. For this experiment, we also report in Appendix E the anomalies correlation values over all features.

6.1.3 Different-attacks signatures matching. As shown in Table 6, as for the tests performed, our signature-matching method effectively discerns different attacks, and does not raise match alerts in the very most of the cases. The only one false-positive match, reported among Foscam camera’s signatures, regards the Telnet SSH scan, and a false-positive anomaly recorded during the vulnerability scan experiment. The average correlation value for the false-positive matching signature features scores relatively low, i.e., 0.13, suggesting again that the two signatures are different. Though, the match is recorded because of similar magnitude of produced clusters, and a perfect correlation on the clusters balance feature, for those minimal fluctuations.

Attack label	No. rejected flows	rejected % of total flows
dos-synflood	44,547	63.19%
unknown	10,413	14.76%
mirai-ackflood	7,659	10.85%
scan	5,815	8.24%
mirai-httpflood	1,737	2.47%
mirai-udpflood	223	< 0.01%
mirai-brutefrc-atk	94	< 0.01%
mirai-brutefrc-vict	59	< 0.01%
Total	**70,547**	-

Table 7: Layout of custom MRT flows dataset derived from Ezviz attack scenarios in Kang et al. [28].

6.2 Flow Feature Selection

We use The *IoT Network Intrusion Dataset* by Kang et al. [28] to understand what flow features help the most in discerning different IoT network attack types. We first generate a MUD profile for the *Ezviz* camera using the benign traffic, and filter all anomalous traffic from the attack scenarios. Besides filtering most malicious packets (99.7%, with 91.3% on average for each attack scenario), noise traffic (i.e., local network traffic, and packets from Amazon, Google, Microsoft, and alike) is also filtered, to which we assign the label *unknown.* Appendix C displays the outcome of this step.

We merge the MRT from all scenarios, and convert it to flows labeled according to the attack they implement. Table 7 presents the resulting labeled dataset. Each flow is described via the set of NetFlow features reported in Appendix A. We compute the Adjusted Mutual Information (AMI) score [70] of each flow feature with respect to the attack labels. AMI is a robust metric in the presence of unbalanced classes, as is the case in the dataset. For two variables, AMI measures how the entropy of one variable X is reduced once the value of the other variable Y is known. A high AMI score means that X is useful in determining the value of Y (refer to Appendix F).

Table 8 presents a list of top-scoring features and their description. Note that we resolved IP addresses to a category label among *private*, *public*, *reserved*, *broadcast*, not to bias the AMI score with specific IPs. 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` because we are more interested in discerning anomalous traffic based on sources, rather than targets.

6.3 Clusters analysis

We cluster flows together based on the features from Section 6.2. Here, the intuition is that flows related to an attack present similar characteristics and can therefore be grouped into clusters. We can evaluate to what extent our mechanism creates clusters containing the same attack (class) using the `homogeneity` [52] score h. An homogeneity score of 1 indicates that, for each cluster, all samples belong to the same class; vice versa, $h = 0$ when all clusters only contain samples of different classes. Besides computing the homogeneity score, we want to minimize the number of produced clusters in relation to the present anomalies. The reason is that one flow per cluster would discern all anomalies and therefore give a perfect homogeneity score of 1, but will not be meaningful as a cluster. To illustrate, an incoming distributed DoS produces a high number of distinct flows, but it should ideally yield a single, or at least a small number of clusters.

Feature	Description	AMI score
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 8: Selected features and their AMI scores with respect to the attack label. We discard `opkt` and `ipkt` because correlated with the higher-scoring `bpp`, `ibyt`, and `obyt`. We discard `da` in favor of `sa` because the source of anomalous traffic is more relevant to our detection approach than the destination.

We use the MRT flows dataset for the *Ezviz* device (Table 7) to perform a grid-search on `HDBSCAN`'s standard parameters `min_cluster_size` and `min_samples` [34] to find a local maximum for homogeneity, and minimum for the amount of yielded clusters. We achieve the best results ($h = 0.866$ and 11 clusters over the eight attack labels) when `min_cluster_size` = 1.2% the size of the dataset, and `min_samples` = 0.2% the size of `min_cluster_size`. On our dataset, this configuration also produces just 0.04% noise points. Table 9 lists the clustering results per produced clusters.

As shown in Table 10, the very most of the flows (93.18% on average) from all attacks except *mirai brute-force* and *UDP flooding* scenarios are part of clusters where their share is most represented (as per Table 9). This happens because, when running the clustering on the whole dataset, `min_cluster_size`'s value was selected greater than the size of brute-force and UDP-flooding classes, which are therefore undetected. We thus also run the clustering on a subset of the dataset containing only brute-force and udp-flooding scenarios, and we observe that the clustering produces usable results in this case well. We report these results in Appendix D.

Overall, these results show that our tool effectively groups anomalous flows of the same type in distinct clusters with high homogeneity. The sensitivity to different anomalous events yields a descriptive set of clusters to characterize the MRT at each time window.

7 DISCUSSION

Signature-matching results. For the experiments we performed, our method could discern 96.15% of the cases when two anomalies were from the same attack *type*, and 94.44% of the cases when anomalies were from different attack types. Note that our experimental procedure included network attacks to devices ranging from four packets (the Telnet SSH ports scan), to tens of thousands of packets (the DoS attacks), as well as a more time-spread scan (the slow vulnerability scan). Therefore, our approach proved to be applicable to different types of network anomalies.

Another interesting aspect is the amount of data generated by our approach, for if a large organization or manufacturer monitors many devices, storage may become a problem. As we report in

Cluster ID	Cluster size in % MRT flows	Most represented attack in cluster	Share of most represented attack
0	2.27%	dos-synflood	94.90%
1	11.00%	unkown	100.00%
2	1.22%	unknown	100.00%
3	1.66%	dos-synflood	99.91%
4	2.43%	dos-synflood	100.00%
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%	-	-

Table 9: Clustering results over produced clusters. Noise points map to cluster *-1*.

Attack label	Total	Represented in a cluster	
		No. flows	% of flows
dos-synflood	44,547	44,038	98.85%
unknown	10,413	8,632	82.90%
mirai-ackflood	7,659	7,659	100.00%
scan	5,815	5,342	91.86%
mirai-httpflood	1,737	871	50.14%
mirai-udpflood	223	0	0.00%
mirai-brutefrc-atk	94	0	0.00%
mirai-brutefrc-vict	59	0	0.00%

Table 10: Percentage of flows that are part of a cluster where their class is the most represented.

Table 5, the data generated by processing a capture of 30 minutes for 6 devices (5 attacked, one unharmed), consists of 310 entries for time windows of 30 seconds each. We store 8 bytes per column value, with 20 columns per transition.

Therefore, for 2 MRT feed entries produced per minute, per device, an MRT feed for 24 hours divided in 30 seconds windows would yield a file of 27.65 MB size, which is reasonable to store in the short term. Now, MRT feeds can be removed when an attack has been investigated, and MRT feeds that do not capture anomalies do not need to be stored at all. Moreover, it would be practical to only store anomaly signatures, yielding even smaller files. Finally, incoming anomalies could be matched on the fly against stored signatures databases. While these file sizes are manageable for smaller settings, we believe it is worth investigating the scalability of this approach to larger-scale scenarios.

Methodology. Our method clusters rejected flows in the attempt of capturing a high-level and concise description of an anomaly. Though we achieved this in the clustering evaluation, in our experimental evaluation we observed that different anomalies produce varying numbers of clusters — i.e., the (more volumetric) attacks we launched were not represented by a single cluster, but instead by multiple clusters. Though somewhat unexpected, this performance does not affect the effectiveness of the approach: we define a signature precisely through the evolution of the clusters' space (recall Section 3.3.1), and we showed that indeed we obtain different signatures for different attacks. As this approach captures similar and different attacks, it gathers additional and novel situational awareness regarding network threats affecting consumer IoT devices. Additionally, the proposed signature generation methodology outputs anonymized data feeds, consumable by interested parties without privacy concerns.

MUD adoption. As mentioned in Section 2, MUD profiles are not yet widely adopted by IoT manufacturers, although research and standardization bodies start to adopt MUD [9]. At the very least, MUD contributes to specification-based security for IoT devices. In turn, our work showcases an additional benefit that IoT manufacturers could gain from adopting MUD, and integrating such an approach. Adopting our solution, IoT vendors could monitor in a privacy-preserving way how malicious activities target their products. They could for instance detect when a particular model receives anomalous traffic from the same attack type at various customers. Besides hinting at possible unknown vulnerabilities, such threat intelligence would provide insights on attackers' interests and trends, and promptly reveal the emergence of large-scale malicious events such as new botnets.

Finally, note that MUDscope merely needs the NetFlow rules derived from MUD profiles to be operational. In conclusion, our results promote the use of MUD profiles to ascertain threat situational awareness for IoT devices. We believe these results motivate further research in the threat-intelligence collection methodology that we propose, and the exploration of others alike.

7.1 Limitations and future work

As our approach relies on MUD profiles for anomaly detection, it inherits their limitations. MUDscope is not able to capture network attacks that can evade MUD rules, such as vendor compromise, man-in-the-middle, or spoofing attacks.

With respect to our signature-matching methodology, we observed some limitations regarding the attacks that we can effectively isolate and match. Anomalies such as a Telnet ports scan, or one-packet Internet-wide scans, may all produce minimal MRT fluctuations and highly similar signatures, making it hard to discern these anomalies. Future work should investigate more advanced methods to differentiate these anomalies. One method could regard adding a signature feature recording the distance from the origin of the clusters space, to observe where small clusters appear, as related to the features of their flows. Additionally, it may be hard to detect matches for attacks that are spread over a long period of time, as they could generate intermittent small signatures. Future work could develop a higher-level alerting module, performing comparisons on sequences of signatures.

With the hypothesis of a MUDscope-aware attacker, we also note that our signature-matching methodology could be circumvented, e.g., by injecting noise traffic to a network attack, or mimicking signatures of non-harmful or non-targeted anomalies. This would still lead to a detection of the attack, but its signature will be unknown.

Finally, note that we could not validate MUDscope with authentic MUD profiles, and we cannot therefore grant that the profiles we generated are a perfect representation of authentic profiles.

8 RELATED WORK

IoT threat landscape monitoring. IoT honeypots and network telescopes are two state-of-art approaches that mostly align with MUDscope's objective.

One widely appreciated contribution to IoT honeypots is IoTPot [47]. With an adaptable backend, IoTPot studies Telnet-based IoT attacks directed to 8 different CPU architectures. The authors provide an analysis on the "scope and variety" of IoT Telnet attacks. In 2018, Kamoen [27] builds upon the IoTPot custom-backend approach to further propose a stateful IoT honeypot ('Honeytrack'), able to maintain the status of the attack progress from a threat agent, and re-present it to the agents upon their successive interactions with the honeypot. Kamoen's Honeytrack focuses specifically on adversary behavior, studying how it evolves when a target is compromised or weaponized. In 2020, Tabari and Ou [62] similarly addresses the challenge of the "largely unknown nature of attackers' activities toward IoT", proposing a honeypot whose interaction is incrementally designed and integrated. They do so to 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., [72] 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.

IoTPot and Honeytrack use darknets to gain preliminary results suggesting what to account for in honeypot architectures. Indeed, darknet-based approaches can produce at-scale insights into IoT threats. A 2018 work from Shaik et al. [60] achieves Internet-scale monitoring of compromised IoT devices, by correlating network telescope captures with threat intelligence feeds. Pour et al. [49] integrate the same setup with geolocation databases and ISPs' feedbacks, and infer at-scale IoT-probing campaigns 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 [53] and consumer-IoT compromises [31]. Furthermore, they implement 'ex-IoT', an IoT threat intelligence feed that streams findings from such network telescope Internet-scale IoT monitoring capability [50]. Notably, Griffioen and Doerr [16] study Mirai-like botnets evolution and behavior by leveraging on 7,500 Honeytrack [27] deployments, Delft's University network telescope, and flow probing of infected devices.

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.

MUD specification. The state-of-art on MUD profiles mostly concerns studies on its effectiveness as threat prevention tool, and proposals to extend their functionality.

In a 2018 work by Hamza et al. [22], 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 [20]. They show that MUD is able to block Internet-side threats, for devices with limited functionality, and specifically against volumetric attacks [19]. The researchers extend their MUD-based intrusion detection method in work from 2019 [21], where they infer different anomalous status cases of devices, by matching their dynamically observed behavior against their MUD-expected behavior.

Other works focus on MUD integration and enhancing profiles' functionalities. Matheu et al. propose a way to extend the profiles to integrate security-testing results [33]. In related work, the authors design an SDN-backed authentication messages exchange to bootstrap MUD profiles in industrial environments [32]. Sajjad et al. [54] extend MUD profiles with firmware integrity information, fetched from vulnerability repositories, and distributed through a blockchain framework. Furthermore, they design a MUD bootstrapping routine accounting for gateway authentication, to prevent attackers to bypass MUD through router vulnerabilities. Feraudo et al. [11] propose a federated-learning framework [14] allowing only MUD-compliant devices to publish training data. Finally, Afek et al. [1] implement an ISP-level service to enforce MUD on behalf of SOHOs, unburdening deployments from the task.

Notably, Afek's work 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. In fact, to the best of our knowledge, no other MUD work has yet explored this aspect. We acknowledge the state-of-art results on MUD as an intrusion detection tool, and as a base to instrument specification-based IoT security. We move from these findings to propose a novel use-case of MUD, focused on analyzing the traffic that is rejected by the profile. Doing so, we gather insights on network threats targeting consumer IoT devices.

9 CONCLUSION

We presented an approach to gather insights into network threats targeting IoT devices. We showed that we can produce attack signatures that identify attacks on IoT devices from various manufacturers. We argue that this technique can be leveraged by both IoT manufacturers and the defenders' community to aid in the prompt detection and analysis of emerging IoT threats. Our approach is based on the MUD specification and leverages the advantages of specification-based IoT security. We implemented our approach, open-sourced our tool, and validated its performance on both an existing dataset and our own, openly available dataset. We showed that MUDscope detects attacks with an F1-score of 95.77% and correctly identifies attack signatures with an F1-score of 87.72%.

ACKNOWLEDGMENTS

We would like to thank our reviewers for their valuable comments and inputs to improve our paper. This work has been supported by the INTERSCT project, Grant No. NWA.1160.18.301, funded by Netherlands Organisation for Scientific Research (NWO). The findings reported herein are solely responsibility of the authors.

REFERENCES

[1] Yehuda Afek, Anat Bremler-Barr, David Hay, Ran Goldschmidt, Lior Shafir, Gafnit Avraham, and Avraham Shalev. 2020. NFV-based IoT Security for Home Networks using MUD. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*. 1–9.

[2] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*. 1093–1110.

[3] AT&T. n.d. AlienVault Open Source SIEM (OSSIM). https://cybersecurity.att.com/products/ossim

[4] Xander Bouwman, Harm Griffioen, Jelle Egbers, Christian Doerr, Bram Klievink, and Michel van Eeten. 2020. A different cup of TI? The added value of commercial threat intelligence. In *Proceedings of the USENIX Security Symposium*.

[5] Censys. n.d. https://censys.io

[6] CISCO. 2011. NetFlow version 9 Flow-Record Format. https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html

[7] CISCO. n.d. Software-Defined Networking. https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html

[8] Andrei Costin and Jonas Zaddach. 2018. IoT Malware : Comprehensive Survey , Analysis Framework and Case Studies. In *BlackHat USA*. https://i.blackhat.com/us-18/Thu-August-9/us-18-Costin-Zaddach-IoT-Malware-Comprehensive-Survey-Analysis-Framework-and-Case-Studies-wp.pdf

[9] Donna Dodson, Douglas Montgomery, Tim Polk, Mudumbai Ranganathan, Murugiah Souppaya, Steve Johnson, Ashwini Kadam, Craig Pratt, Darshak Thakore, Mark Walker, Eliot Lear, Brian Weis, William C. Barker, Dean Coclin, Avesta Hojjati, Clint Wilson, Tim Jones, Adnan Baykal, Drew Cohen, Kevin Yeich, Yemi Fashina, Parisa Grayeli, Joshua Harrington, Joshua Klosterman, Blaine Mulugeta, Susan Symington, and Jaideep Singh. 2021-05-25. 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*. https://doi.org/10.6028/NIST.SP.1800-15

[10] ENISA. 2021. Internet of Things (IoT) — ENISA. https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures/iot

[11] Angelo Feraudo, Poonam Yadav, Vadim Safronov, Diana Andreea Popescu, Richard Mortier, Shiqiang Wang, Paolo Bellavista, and Jon Crowcroft. 2020-04-27. CoLearn: enabling federated learning in MUD-compliant IoT edge networks. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking (EdgeSys '20)*. Association for Computing Machinery, 25–30. https://doi.org/10.1145/3378679.3394528

[12] Javier Franco, Ahmet Aris, Berk Canberk, and A. Selcuk Uluagac. 2021-08-04. 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). arXiv:2108.02287 http://arxiv.org/abs/2108.02287

[13] Oscar Garcia-Morchon, Sandeep Kumar, and Mohit Sethi. 2019-04. RFC 8576 - Internet of Things (IoT) Security: State of the Art and Challenges. https://tools.ietf.org/html/rfc8576

[14] Google. 2017. Definition of Federated Learning. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[15] Harm Griffioen, Tim M. Booij, and Christian Doerr. 2020. Quality Evaluation of Cyber Threat Intelligence Feeds. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*, Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi (Eds.).

[16] Harm Griffioen and Christian Doerr. 2020. Examining Mirai's Battle over the Internet of Things. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[17] H. J. Griffioen. 2018. Scanners: Discovery of distributed slow scanners in telescope data. (2018). https://repository.tudelft.nl/islandora/object/uuid%3Adcb1669d-d81e-4aa3-bbd1-65049c3209c5

[18] Shuai Guo, Wenbing Lin, Kaiyang Zhao, and Yang Su. 2021. Comparison of Clustering-based Network Traffic Anomaly Detection Methods. In *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Vol. 4. 365–369. https://doi.org/10.1109/IMCEC51613.2021.9482304 ISSN: 2693-2776.

[19] Ayyoob Hamza, Hassan Habibi Gharakheili, Theophilus A. Benson, and Vijay Sivaraman. 2019. Detecting Volumetric Attacks on loT Devices via SDN-Based Monitoring of MUD Activity. In *Proceedings of the ACM Symposium on SDN Research*. Association for Computing Machinery.

[20] Ayyoob Hamza, Hassan Habibi Gharakheili, and Vijay Sivaraman. 2018. Combining MUD Policies with SDN for IoT Intrusion Detection. In *Proceedings of the Workshop on IoT Security and Privacy (IoT S&P '18)*. Association for Computing Machinery. https://doi.org/10.1145/3229565.3229571

[21] Ayyoob Hamza, Dinesha Ranathunga, Hassan Habibi Gharakheili, Theophilus Benson, Matthew Roughan, and Vijay Sivaraman. 2019. Verifying and Monitoring IoTs Network Behavior using MUD Profiles. *CoRR* abs/1902.02484 (2019). arXiv:1902.02484 http://arxiv.org/abs/1902.02484

[22] Ayyoob Hamza, Dinesha Ranathunga, Hassan Habibi Gharakheili, Matthew Roughan, and Vijay Sivaraman. 2018. Clear as MUD: Generating, Validating and Applying IoT Behavioral Profiles. In *Proceedings of the Workshop on IoT Security and Privacy*. Association for Computing Machinery, 8–14.

[23] Arne Holst. 2021. IoT connected devices worldwide 2019-2030. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[24] Martin Husák, Nataliia Neshenko, Morteza Safaei Pour, Elias Bou-Harb, and Pavel Čeleda. 2018. Assessing Internet-wide Cyber Situational Awareness of Critical Sectors. In *Proceedings of the International Conference on Availability, Reliability and Security*. ACM, 1–6.

[25] IETF. n.d. Internet Engineering Task Force. https://www.ietf.org/

[26] Internet-Storm-Center. n.d. DShield Honeypot. https://isc.sans.edu/honeypot.html

[27] Sille Kamoen. 2018. Honeytrack: Persistent honeypot for the Internet of Things. (2018). https://repository.tudelft.nl/islandora/object/uuid%3A344bd7aa-0a17-47dc-92fd-bd6f7e7b08c8

[28] Hyunjae Kang, Dong Hyun Ahn, Gyung Min Lee, Jeong Do Yoo, Kyung Ho Park, and Huy Kang Kim. 2019. IoT network intrusion dataset. IEEE Dataport. https://doi.org/10.21227/q70p-q449

[29] Biar Koshnaw and Steven Furnell. 2022. Assessing cyber security consumer support from technology retailers. *Computer Fraud & Security* 2022, 3 (2022).

[30] Eliot Lear, Dan Romascanu, and Ralph Droms. 2019-03. IETF RFC 8520 - Manufacturer Usage Description (MUD) Specification. https://tools.ietf.org/html/rfc8520

[31] Antonio Mangino, Morteza Safaei Pour, and Elias Bou-Harb. 2020-10-12. Internet-scale Insecurity of Consumer Internet of Things: An Empirical Measurements Perspective. *ACM Transactions on Management Information Systems* 11, 4 (2020-10-12), 21:1–21:24. https://doi.org/10.1145/3394504

[32] Sara Nieves Matheu, Alejandro Molina Zarca, José Luis Hernández-Ramos, Jorge Bernal Bernabé, and Antonio Skarmeta Gómez. 2019-10-28. Enforcing Behavioral Profiles through Software-Defined Networks in the Industrial Internet of Things. *Applied Sciences* 9, 21 (2019-10-28), 4576. https://doi.org/10.3390/app9214576

[33] Sara Nieves Matheu, José Luis Hernández Ramos, Salvador Pérez, and Antonio F. Skarmeta. 2019. Extending MUD Profiles Through an Automated IoT Security Testing Methodology. *IEEE Access* 7 (2019), 149444–149463. https://doi.org/10.1109/ACCESS.2019.2947157

[34] Leland McInnes and Steven Horn. 2017-11-10. How HDBSCAN Works — hdbscan 0.8.1 documentation. https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

[35] Leland McInnes and Marek Suppa. 2018-12-31. Benchmarking Performance and Scaling of Python Clustering Algorithms — hdbscan 0.8.1 documentation. https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html

[36] Dave McMillen. 2021. Internet of Threats: IoT Botnets Drive Surge in Network Attacks. https://securityintelligence.com/posts/internet-of-threats-iot-botnets-network-attacks/

[37] Max Meijer, Giacomo Tommaso Petrucci, Matthjis Schotsman, Luca Morgese Zangrandi, Thjis van Ede, Andrea Continella, Ganduulga Gankhuyag, Luca Allodi, and Savio Sciancalepore. 2022. Federated Lab (FedLab): An Open-source Distributed Platform for Internet of Things (IoT) Research and Experimentation. In *IEEE World Forum on IoT*.

[38] MITRE. n.d. ATT&CK Framework. https://attack.mitre.org/

[39] David Moore. 2003. 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*, Æleen Frisch (Ed.). USENIX.

[40] Luca Morgese Zangrandi and Thijs van Ede. 2022. MUDscope tool. https://github.com/lucamrgs/MUDscope

[41] Luca Morgese Zangrandi, Thijs van Ede, Tim Booij, Savio Sciancalepore, Luca Allodi, and Andrea Continella. 2022. MUDscope Dataset. https://doi.org/10.5281/zenodo.7182597

[42] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum, and Nasir Ghani. 2019. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Commun. Surv. Tutorials* 21, 3 (2019), 2702–2733. https://doi.org/10.1109/COMST.2019.2910750

[43] NETSCOUT. 2021. Cyber Security & Threat Intelligence Report | NETSCOUT. https://www.netscout.com/threatreport/

[44] NIST. n.d. MUD Related Resources. https://www.nccoe.nist.gov/mud-related-resources

[45] Numpy. n.d. https://numpy.org/

[46] Open Networking Foundation. 2012-09-06. OpenFlow Switch Specification (1.3.1). https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf

[47] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: Analysing the Rise of IoT Compromises. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*.

[48] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. 2004. Characteristics of internet background radiation. In *Proceedings of the ACM SIGCOMM conference on Internet measurement (IMC)*.

[49] Morteza Safaei Pour, Elias Bou-Harb, Kavita Varma, Nataliia Neshenko, Dimitris A.Pados, and Kim-Kwang Raymond Chooc. 2019-04-01. Comprehending the IoT cyber threat landscape: A data dimensionality reduction technique to infer and characterize Internet-scale IoT probing campaigns. *Digital Investigation* 28 (2019-04-01), S40–S49. https://doi.org/10.1016/j.diin.2019.01.014

[50] Morteza Safaei Pour, Dylan Watson, and Elias Bou-Harb. 2021. Sanitizing the IoT Cyber Security Posture: An Operational CTI Feed Backed up by Internet Measurements. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[51] Philipp Richter and Arthur Berger. 2019. Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope. In *Proceedings of the ACM Internet Measurement Conference (IMC)*. 144–157.

[52] Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 410–420.

[53] Morteza Safaei Pour, Antonio Mangino, Kurt Friday, Matthias Rathbun, Elias Bou-Harb, Farkhund Iqbal, Sagar Samtani, Jorge Crichigno, and Nasir Ghani. 2020-04-01. On data-driven curation, learning, and analysis for inferring evolving internet-of-Things (IoT) botnets in the wild. *Computers & Security* 91 (2020-04-01), 101707. https://doi.org/10.1016/j.cose.2019.101707

[54] Syed Muhammad Sajjad, Muhammad Yousaf, Humaira Afzal, and Muhammad Rafiq Mufti. 2020. eMUD: Enhanced Manufacturer Usage Description for IoT Botnets Prevention on Home WiFi Routers. *IEEE Access* 8 (2020), 164200–164213. https://doi.org/10.1109/ACCESS.2020.3022272

[55] Jibran Saleem, Mohammad Hammoudeh, Umar Raza, Bamidele Adebisi, and Ruth Ande. 2018-06-26. IoT standardisation: challenges, perspectives and solution. In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems (ICFNDS '18)*. Association for Computing Machinery, 1–9. https://doi.org/10.1145/3231053.3231103

[56] Scapy. n.d. https://scapy.net/

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

[58] Scikit-learn. n.d. https://scikit-learn.org/stable/

[59] Tara Seals. 2021-09-06. IoT Attacks Skyrocket, Doubling in 6 Months. https://threatpost.com/iot-attacks-doubling/169224/

[60] Farooq Shaikh, Elias Bou-Harb, Nataliia Neshenko, Andrea P. Wright, and Nasir Ghani. 2018-09. Internet of Malicious Things: Correlating Active and Passive Measurements for Inferring and Characterizing Internet-Scale Unsolicited IoT Devices. *IEEE Communications Magazine* 56, 9 (2018-09), 170–177. https://doi.org/10.1109/MCOM.2018.1700685

[61] Shodan. n.d. https://www.shodan.io

[62] Armin Ziaie Tabari and Xinming Ou. 2020. A First Step Towards Understanding Real-world Attacks on IoT Devices. *CoRR* abs/2003.01218 (2020). arXiv:2003.01218 https://arxiv.org/abs/2003.01218

[63] Sahrish Khan Tayyaba, Munam Ali Shah, Omair Ahmad Khan, and Abdul Wahab Ahmed. 2017-07-19. 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.

[64] tcpdump. n.d. https://www.tcpdump.org/

[65] Sadegh Torabi, Elias Bou-Harb, Chadi Assi, Mario Galluscio, Amine Boukhtouta, and Mourad Debbabi. 2018. Inferring, Characterizing, and Investigating Internet-Scale Malicious IoT Device Activities: A Network Telescope Perspective. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[66] Haseeb Touqeer, Shakir Zaman, Rashid Amin, Mudassar Hussain, Fadi Al-Turjman, and Muhammad Bilal. 2021. Smart home security: challenges, issues and solutions at different IoT layers. *The Journal of Supercomputing* 77, 12 (2021), 14053–14089.

[67] Ubuntu. 2019. nfdump manual. https://manpages.ubuntu.com/manpages/xenial/man1/nfdump.1.html

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

[69] Alexander Vetterl and Richard Clayton. 2018. Bitter Harvest: Systematically Fingerprinting Low- and Medium-interaction Honeypots at Internet Scale. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*.

[70] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research* (2010).

[71] Wireshark. n.d. https://www.wireshark.org/

[72] Chun-Jung Wu, Katsunari Yoshioka, and Tsutomu Matsumoto. 2020. ThingGate: A Gateway for Managing Traffic of Bare-metal IoT Honeypot. *Journal of Information Processing* 28, 0 (2020), 481–492. https://doi.org/10.2197/ipsjjip.28.481

A TRAFFIC FLOW FEATURES FORM NFDUMP

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 IOT INTRUSION DETECTION DATASET

By Kang et al. [28], overviewed in Table 11.

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

Table 11: Description of the used portion of the *IoT Network Intrusion Dataset*, by Kang et al. [28]. Each traffic scenario is a traffic capture file containing related activities. The target device is either victim or executor (→) of the attack.

C PRELIMINARY MUD ANOMALY PREVENTION

Table 12 overviews the result of packet-filtering with MUD profiles, for the Kang dataset. Where the MRT packets are double those of the ground truth, this happens simply because the generated MUD profiles captured both incoming and outgoing packets, whereas in the dataset, only incoming packets are marked as ground truth. Additional packets regards probes from Microsoft, Amazon, Google

Ezviz		**# of packets**				
Attack scenario	**Role**	**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 12: 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.**

and alike. Overall, the results of the filtering set the base for our further analyses.

D CLUSTERING PERFORMANCE ON LESS REPRESENTED ATTACKS

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

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

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

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': {
    'out-represented_avg_percentage': 61.76470588235294,
    'represented_avg_percentage': 100.0
    },
 'MIRAI-UDPFLOODING': {
    'out-represented_avg_percentage': 38.23529411764706,
    'represented_avg_percentage': 100.0}
    }

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'],
```

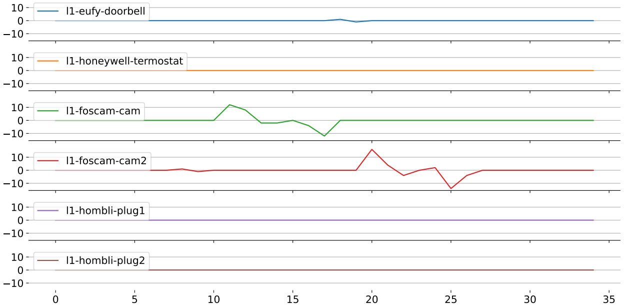

Figure 5: Sample output from the comparison of anomalous signatures through multiple feeds. In this Figure, the DoS signature for Foscam cam and Foscam cam2 can be appreciated for the `clusters balance` feature.

```
 '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 characterization that will be consistently replicated for similar anomalous traffic captured.

E MRT FEEDS SIGNATURES COMPARISON SAMPLE

Figure 5 presents a sample plot for the trendo of the `clusters balance` feature across the MRT feeds of the six tested IoT devices. We can visually appreciate the similarity of the fluctuations through the two lines. Note there are fluctuations also in the MRT feed for the Eufy doorbell, through the time windows 17 to 20, and a false-positive fluctuation for the MRT feed of Foscam cam 2, between windows 7 and 10. In the listing below, we report the report output produced by our tool. We remind that the complete set of plots, and anomaly detection and signature-matching reports is publicly available to view [40].

```
*~*~*~*~*~*~*~* Anomalies recorded for each MRT feed submitted *~*~*~*~*~*~*~*

clusters_evols_record-session4_l1-eufy-doorbell :
        between: 20:58:32 and 20:59:52 on the 2022-06-24        ---      time
              windows [ 17 , 20 ]

clusters_evols_record-session4_l1-foscam-cam :
        between: 20:54:59 and 20:58:44 on the 2022-06-24        ---      time
              windows [ 10 , 18 ]

clusters_evols_record-session4_l1-foscam-cam2 :
        between: 20:53:48 and 20:54:38 on the 2022-06-24        ---      time
              windows [ 7 , 10 ]
        between: 20:59:38 and 21:02:59 on the 2022-06-24        ---      time
              windows [ 19 , 27 ]



 >>>>>>>>>>>> Common anomalies found <<<<<<<<<<<<


[ 0 ]Device ID          Signature transitions window    Signature time window
                                MRT Feed
----------------   ------------------------------
      ------------------------------------------
      ----------------------------------------------------
l1-eufy-doorbell   [17 - 20]                          2022-06-24 20:58:32 -
      2022-06-24 20:59:52  clusters_evols_record-session4_l1-eufy-doorbell
l1-foscam-cam2     [7 - 10]                           2022-06-24 20:53:48 -
      2022-06-24 20:54:38  clusters_evols_record-session4_l1-foscam-cam2
Max features correlation : 1.0   ---    Avg features correlation :
      3.9832882741253074e-05      ---    Combined score : 0.5000199164413707
Correlation penalty multiplyer for clusters difference: 1

        Correlation values for signature features:
        {'all_dists_avg': -0.9996016711725875, 'mutual_matches_n': 0.0, '
              mutual_matches_percentage': 0.0, 'fwd_matches_n': 0.0, '
              fwd_matches_percentage': 0.0, 'fwd_matches_agglomeration_avg':
              0.0, 'bwd_matches_n': 0.0, 'bwd_matches_percentage': 0.0, '
              bwd_matches_agglomeration_avg': 0.0, 'clusters_balance': 1.0}



[ 1 ]Device ID        Signature transitions window    Signature time window
                                MRT Feed
--------------   ------------------------------
      ------------------------------------------
      ----------------------------------------------
l1-foscam-cam    [10 - 18]                       2022-06-24 20:54:59 - 2022-06-24
       20:58:44  clusters_evols_record-session4_l1-foscam-cam
l1-foscam-cam2   [19 - 27]                       2022-06-24 20:59:38 - 2022-06-24
       21:02:59  clusters_evols_record-session4_l1-foscam-cam2
Max features correlation : 0.9998584712478614   ---    Avg features
      correlation : 0.8650292909226683   ---    Combined score :
      0.9324438810852649
Correlation penalty multiplyer for clusters difference: 1

        Correlation values for signature features:
        {'fwd_matches_agglomeration_avg': 0.5819416382552827, 'fwd_matches_n':
              0.75333824386712, 'clusters_balance': 0.7718449849879597, '
              bwd_matches_n': 0.8485552916276634, 'fwd_matches_percentage':
              0.8862326436248709, 'mutual_matches_n': 0.8978247615364664, '
              mutual_matches_percentage': 0.9501808393990387, '
              bwd_matches_percentage': 0.9615988290998247, '
              bwd_matches_agglomeration_avg': 0.9989172055805962, '
              all_dists_avg': 0.9998584712478614}
```

F AMI FORMULA

To produce the AMI score, we first compute the mutual information $MI(X, Y)$:

$$MI(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right)$$

For our analysis, x and y are the values of the flow features, and the attack class, respectively. We then adjust the value by normalizing it on the expected value for MI, and the average over the entropy of X and Y:

$$AMI(X, Y) = \frac{MI(X, Y) - E[MI(X, Y)]}{\max(H(X), H(Y)) - E[MI(X, Y)]}.$$