PicP-MUD: Profiling Information Content of Payloads in MUD Flows for IoT Devices

Arman Pashamokhtari, Arunan Sivanathan, Ayyoob Hamza, and Hassan Habibi Gharakheili
School of Electrical Engineering and Telecommunications, UNSW Sydney, Australia
Emails: {a.pashamokhtari, a.sivanathan, m.ahamedhamza, h.habibi}@unsw.edu.au

***Abstract*—The Manufacturer Usage Description (MUD) standard aims to reduce the attack surface for IoT devices by locking down their behavior to a formally-specified set of network flows (access control entries). Formal network behaviors can also be systematically and rigorously verified in any operating environment. Enforcing MUD flows and monitoring their activity in real-time can be relatively effective in securing IoT devices; however, its scope is limited to endpoints (domain names and IP addresses) and transport-layer protocols and services. Therefore, misconfigured or compromised IoTs may conform to their MUD-specified behavior but exchange unintended (or even malicious) *contents* across those flows. This paper develops *PicP-MUD* with the aim to profile the information content of packet payloads (whether unencrypted, encoded, or encrypted) in each MUD flow of an IoT device. That way, certain tasks like cyber-risk analysis, change detection, or selective deep packet inspection can be performed in a more systematic manner. Our contributions are twofold: (1) We analyze over 123K network flows of 6 transparent (*e.g.,* `HTTP`), 11 encrypted (*e.g.,* `TLS`), and 7 encoded (*e.g.,* `RTP`) protocols, collected in our lab and obtained from public datasets, to identify 17 statistical features of their application payload, helping us distinguish different content types; and (2) We develop and evaluate PicP-MUD using a machine learning model, and show how we achieve an average accuracy of 99% in predicting the content type of a flow.**

***Index Terms*—IoT, MUD, encrypted traffic analysis**

I. Introduction

IoT devices and systems are exposed to a wide range of sophisticated cyber attacks [1] exploiting vulnerabilities such as default credentials (*e.g.,* causing code injection) and unencrypted traffic exchange (*e.g.,* leading to private data leaks). Unmanaged IoT devices can present extreme risks to network operators – examples are DDoS, botnet, and ransomware attacks [2]. According to a recent survey [3] of about 150K connected cameras, about 80% of them communicate via plaintext (unencrypted) protocols `HTTP` and `FTP`.

Although unencrypted communications bring cyber risks to IoT systems, traffic encryption does not necessarily mitigate those risks and may present other challenges [4]. Security appliances like firewalls cannot thoroughly inspect encrypted traffic, especially at scale (performance bottlenecks). Hence, cybercriminals tend to employ encryption to hide their malicious activities (*e.g.,* command-and-control messages or exfiltrated data) from being detected by IDSes and firewalls [5].

IoT devices differ from non-IoTs in their network behavior and traffic characteristics. Hence, security measures used for non-IoTs cannot be directly applied to IoTs [6]. However, a limited (and identifiable) set of behaviors for IoT devices enables their operators to "whitelist" their benign network activities by specifying intended local/remote servers and network services in the form of Manufacturer Usage Description (MUD) [7] profile – a set of access flows (*e.g.,* expressing an IP protocol, source/destination ports, and source/destination IP addresses). The MUD standard aims to define lightweight formal metadata for IoT devices to reduce their attack surface – locking down their network behavior to a limited set specified by the MUD profile (disallowing unintended activities). Researchers have developed techniques to use MUD profiles as signatures for classifying IoT devices [8], systematically enforce security policies into networks [9], and generate baseline models for anomaly detection [10].

In this paper[1], we develop a classifier model to infer from traffic contents of a given MUD flow, extending the capability of MUD profiles. We classify MUD flows into three classes of content types, namely transparent (like `HTTP`), encrypted (like `TLS`), and encoded (like `RTP`). Having the ability to classify the content type of flows comes with several advantages: (a) one can perform a systematic risk assessment on the network behavior of an IoT device by quantifying whether their traffic is exchanged confidentially (encrypted) or not (transparent); (b) by knowing the type of contents, opaque flows (encrypted and/or encoded) can be offloaded from DPI-based intrusion detection systems as deep inspection of opaque data gives little (if any) insight; and, (c) one can track the content type of target flows over time and detect deviations from a "baseline".

Our contributions are as follows: **(1)** We analyze the content of a large and diverse set of flows: 106K flows collected from our IoT testbed containing traffic of `TLS`, `HTTP`, `DNS`, `NTP`, and `RTP` protocols; 5K flows generated and collected in our lab using 10 popular encryption algorithms (`AES`, `Blowfish`, `Camellia`, `RSA`, `ChaCha20`, `SEED`, `SM4`, `3DES`, `CAST5`, and `IDEA`), two compression algorithms (`gzip` and `bzip`), and four audio/video formats (`PNG`, `JPEG`, `MP3`, `MP4`); 600 `TLS` flows popular websites collected in our lab; two public datasets containing `FTP` [11] and `SMB` [12] traffic traces. We identify 17 statistical features from the byte values of packet payload for each flow and highlight their prediction power for determining the content type of flows; and **(2)** We train and evaluate the performance of a Random Forest model and show how it can give 99% accuracy in predicting the content class by analyzing the first 6KB worth of data of a flow.

[1]Funding for this project was provided by CyAmast Pty Ltd.

II. Related Work

IoT Network Security: Researchers have proposed methods for securing IoT devices at network level. Works in [13]–[16] developed models to classify network activities of connected IoT devices. Authors of [13] and [14] employed statistical traffic features like distribution of transport-layer port numbers and flow volume (flow-based), or IP time-to-live and maximum segment size (packet-based) to train a machine learning model to infer various types of IoTs from their network traffic. Works in [15] and [16] used deterministic features like IP address and/or domain name of cloud servers to identify IoT devices. Work in [17] trained machine learning models on features like packet size, packet inter-arrival time, and traffic rate (for benign and malicious instances) to train anomaly detectors. The use of MUD profiles for securing IoT devices has been studied over the last few years [8]–[10], [18]. Authors in [9] developed a method to translate MUD profiles of IoT devices into flow rules ("allowlisting") on an SDN-based gateway – that way, the majority of network attacks could be prevented given MUD profiles are tightly specified. Work in [10] developed one-class clustering-based models for time-series activity of individual MUD flows to detect anomalous behaviors (volumetric attacks) when sophisticated attackers sim to conform to MUD profiles on the network.

Encrypted Data Analysis: Inferring from encrypted data has been a subject of research for website fingerprinting [19], application classification [20], and identifying Operation Systems [21]. Work in [5] enhanced BotHunter [22] (Snort-based botnet detector) to detect botnets that transmit encrypted traffic. BotHunter is triggered when specific rules like network scans, DNS lookups for blacklisted domains, or known signatures inside "unencrypted" traffic payloads are matched. The authors employed measures of entropy to identify flows carrying encrypted traffic that, in conjunction with malicious rules (blacklisted IP/domains), enable network operators to detect certain encrypted botnets. Authors of [23] developed a model to analyze audio/video data of streaming platforms like Netflix and Spotify. Their model employs measures of entropy and chi-square to infer the media state (encrypted, encoded with media codec, and decoded for presentation), thus allowing to extract copyright-protected media from RAM once decoded. Work in [24] developed machine learning models to detect malicious traffic carried over DoH (`DNS` over `HTTPS`). They used entropy along with statistical flow-based features (packet/byte counts).

To the best of our knowledge, this paper is the first to propose a traffic content classifier for MUD flows of IoT devices. We use MUD as an enabler of our architecture for systematic and selective IoT traffic monitoring. Our model uses a comprehensive set of statistical features (both packet-level and flow-level) to determine three content classes (transparent, encrypted, or encoded). Also, in contrast to prior work, we collect and analyze traffic traces of an extended range of protocols and evaluate the efficacy of our model on unseen protocols.

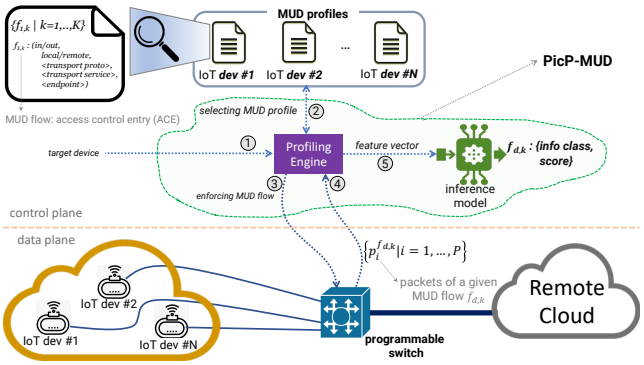


Fig. 1. System architecture of our PicP-MUD.

III. Profiling Information Content of Payloads in MUD Flows

Fig. 1 illustrates the architecture of our system for profiling the content (information) of MUD flows for a given network of IoT devices. The top half of the figure shows our control plane, which is responsible for inference and processing, while the bottom half takes care of networking and traffic forwarding. Starting from the top, MUD profiles of connected IoT devices are assumed to be available. Note that IoT manufacturers are expected to supply MUD profiles. At the time of writing, manufacturers have not published the profile of their commercial IoT devices. However, there are open-source tools like *MUDgee* [8], which can help network operators automatically generate the MUD profile for an IoT device from its traffic trace. Also, there are ongoing efforts [25] to facilitate the adoption and sharing of MUD profiles across the Internet community. A MUD profile consists of a list of access control entries (ACE). For example, the profile of $dev\ \#1$ in Fig. 1 contains K flow entries denoted by $f_{1,k}$, where $k \in [1, K]$. Each ACE specifies the flow direction (incoming↓ or outgoing↑), flow scope (local/remote), transport-layer protocol (`TCP/UDP`), transport-layer service (*i.e.,* port number), and endpoints (IP address/domain name).

Flow of Events: Given a target IoT device on the network (step (1) in Fig. 1), the Profiling Engine first fetches the corresponding MUD profile from a repository (step (2)). Next, it enforces (step (3)) an ACE flow $f_{d,k}$ into a programmable switch (*e.g.,* OpenFlow or P4-based) that is in charge of traffic forwarding. In response, the switch will provide the Engine with packets $p_i^{f_{d,k}}$ of the requested flow (step (4)) – selective traffic analysis. In §IV, we will discuss the amount traffic (in terms of byte count) is required for an accurate classification. Note that the Engine can be programmed to profile a set of (or all) ACE flows. In that case, corresponding states need to be maintained by the Engine. Finally, the Engine computes the required features of the corresponding packet set and passes a vector (per each MUD flow) to the inference model to make an inference ((5)). The model provides the class of contents (*e.g.,* transparent, encrypted, or encoded) with a score between 0 to 1 (higher scores indicate more confidence).

We note that the desired set of content classes can vary in different use cases. For example, in botnet detection [5],

TABLE I
SUMMARY OF OUR LABELED DATASET.

Protocol	# flows	Information class
DNS	62,023	transparent
FTP	11,125	transparent
HTTP	8,852	transparent
NTP	2,940	transparent
SMB	309	transparent
UDP	442	transparent
TLS	32,800	encrypted
3DES	100	encrypted
AES	99	encrypted
Blowfish	100	encrypted
CAST5	99	encrypted
Camellia	98	encrypted
ChaCha20	100	encrypted
IDEA	99	encrypted
RSA	99	encrypted
SEED	100	encrypted
SM4	100	encrypted
RTP	81	encoded
bzip	983	encoded
gzip	992	encoded
JPEG	486	encoded
MP3	480	encoded
MP4	493	encoded
PNG	495	encoded

distinguishing encrypted flows is of paramount importance, and therefore merging transparent and encoded flows into a class of non-encrypted traffic can be acceptable; or, for deep packet inspection (DPI) offloading [26], only two classes of transparent and opaque (encrypted or encoded) would suffice. For an objective like dynamic risk analysis and continuous behavioral monitoring of connected IoTs, more content types offer richer insights. For instance, the cyber risk of each class is different: transparent (high), encoded (medium as it can be decoded by just knowing the encoding algorithm), and encrypted (low as it requires a secret encryption key). Also, finer-grained anomaly detection can be performed with three classes compared to two classes. Note that this paper focuses on developing the profiling engine. The use of content profiling for risk assessment and behavioral monitoring is beyond the scope of this paper and is left to future work.

A. Our Traffic Dataset

To have a well-trained model, a rich dataset is needed. Our dataset consists of traffic traces that we collect in our lab and obtain from public sources. We process PCAP files to aggregate packets into 5-tuple flows – IP protocol, source/destination IP address, and source/destination port number are used the five-tuple key to distinguish flows. One may use flow termination signals like `TCP FIN` for `TCP` flows, or a configurable timeout for `UDP` flows, and extract multiple instances of the same 5-tuple flow over time. However, in this paper, we do not terminate 5-tuple flows in our dataset processing. Table I summarizes our labeled dataset indicating applications/protocols, flow count, and corresponding information class.

Our IoT Lab Traffic: We collect 106,070 flows from our testbed, consisting of 13 IoT devices ranging from smart cameras, speakers, and smoke sensor to weighing scale and power switch. We specifically collect `TLS`, `HTTP`, `DNS`, `NTP`, and `RTP` (video feed stream from a smart camera) traffic flows from these IoTs.

TLS web Traffic: We collect 626 `TLS` flows of popular websites like Amazon, Apple, Facebook, Google, IBM, Microsoft, Oracle, Twitter, and YouTube, initiated by our lab computers.

Synthetic Traffic: To further diversify our dataset, we wrote a script to transfer various content types between two computers via `UDP` flows. We collect traffic traces of these locally exchanged flows from our lab's network. Note that our focus is on computing features from the content (payload) of flows; therefore, the transport-layer protocol (`TCP` or `UDP`) would not affect our inference. Using our script, we generated 442 `UDP` flows containing plaintext payloads, 994 encrypted flows from 10 popular encryption algorithms, and 3929 encoded flows from a range of compression and media encoding algorithms.

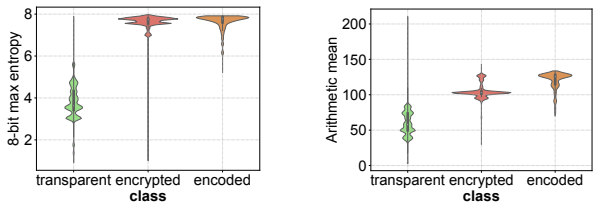
Public Datasets: In order to expand our transparent flows, we obtain `FTP` and `SMB` raw packet traces from public sources [11] and [12], respectively. The `FTP` dataset contains traffic traces of anonymous users worldwide interacting with a public `FTP` server at the Lawrence Berkeley National Laboratory. These traces only include `FTP` commands (both requests/responses on `TCP/21`) without any data flows. The `SMB` traffic traces were collected by Wireshark developers through running Samba torture test suite [27] over a Samba server. Samba torture contains a set of software tests to be performed on Samba servers to validate they are operational.

B. Traffic Features

During the data analysis phase, we found that certain encryption algorithms (like AES and 3DES) in some modes of operation like Cipher Block Chaining (CBC) tend to embed a fixed-size initialization vector at the beginning of each packet for a given encrypted flow. Initialization vector size is equal to the block size of encryption algorithms which is often 16 bytes (or less) [28]. As these bytes are non-encrypted, they can introduce noise to our inference process. Therefore, we trim the first 16 bytes of all packets in our dataset before feature extraction, regardless of their content type.

Inspired by prior works [5], [23], [29], we use a combination of features (computed on both packet-based and flow-based) from byte values of traffic content for our inference model. We employ three types of statistical metrics (explained next) to construct our feature set.

(1) Shannon's entropy: Entropy indicates the amount of information embedded in a given message (the payload of packets/flows in our context). The entropy value for random payloads (like those in encrypted packets) has higher values than plaintext payloads. To compute entropy, we first need to decide a "symbol size"; *e.g.,* for binary values, symbol size of two bits results in four possible symbols *i.e.,* 00, 01, 10, and 11. Therefore, m-bit entropy denotes entropy calculated based on m-bit symbols. In this paper, we use 4-bit and 8-bit entropy, representing HEX and ASCII encoding. According to the probability of each symbol inside a given message, entropy is computed by:



(a) Maximum 8-bit entropy (packet-based approach). (b) Arithmetic mean (flow-based approach).

Fig. 2. Predictive power of: (a) packet-based entropy, and (b) flow-based arithmetic mean, in distinguishing the three content classes.

$$H(x) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \quad (1)$$

where the message contains n distinct symbols x_i. Given a symbol x_i, probability $P(x_i)$ is equal to the number of times x_i was found in the message divided by the total number of symbols inside the message.

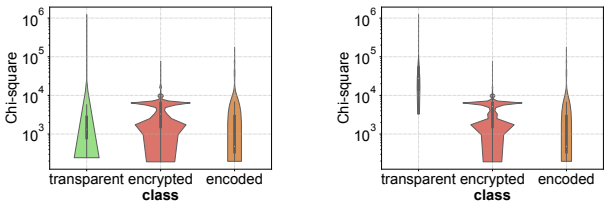
Entropy is an indicative metric for distinguishing transparent messages from opaque ones [23]. However, it falls short in a finer-grained classification of opaque contents – encrypted versus encoded. One aspect that encrypted and encoded contents are different is their byte value distribution. Strong encryption algorithms are expected to display high degrees of confusion and diffusion [30], which result in encrypted outputs becoming like pseudo-random numbers. Therefore, encrypted contents will likely display a uniform byte value distribution. However, as encoding algorithms are not designed to have such properties, their output byte distribution may not necessarily be uniform [31], [32]. Hence, the detection of encoded contents is relatively challenging. Therefore, the following two metrics are used to capture the byte value distribution of contents, helping us distinguish encrypted from encoded traffic.

(2) Arithmetic mean and standard deviation: For a given array of bytes, we first convert each byte to its corresponding decimal value (*i.e.,* $[0-255]$), followed by computing their average and standard deviation. A true random (uniform) distribution of decimal values gives arithmetic mean of 127.5 and standard deviation 73.7. We expect encrypted contents' mean and standard deviation to be closer to those metrics of the uniform distribution.

(3) Chi-square: Chi-square is a statistical test that measures the similarity between two discrete distributions. Smaller χ^2 values highlight more similarity. For us, the objective is to quantify the similarity between the distribution of measured byte values and the uniform distribution (reference). χ^2 is computed by:

$$\chi^2 = \sum_{i=0}^{255} \frac{(O_i - E_i)^2}{E_i} \quad (2)$$

where O_i denotes the number of times that byte value i was observed in a given byte array (measured payload). Also, E_i refers to the expected byte value i in the reference distribution. Given our reference distribution is uniform, E_i is constant and equal to $n/256$ in which n is the total number of bytes inside the measured payload.



(a) All flows. (b) Flows with size $\geq$1.5KB.

Fig. 3. Measure of chi-square for: (a) all flows, and (b) those flows with the payload size of $\geq$ 1.5KB, in our dataset.

Note that the three metrics discussed above can be computed on either a packet basis or flow basis. In what follows, we incorporate both methods to enrich the features for our model.

Packet-based Computing Approach: Given a flow with P packets, we first compute three metrics, namely 4-bit entropy, 8-bit entropy, and arithmetic mean for individual packets. We choose to not compute the chi-square metric for individual packets as possible fragmentation in IPv4 packets (due to maximum transmission unit of 1500 bytes imposed by Ethernet) makes this metric relatively less reliable (explained in III-C). We next obtain a corresponding distribution (*i.e.,* average, standard deviation, minimum, and maximum values) for each of these metrics. Hence, a total of 12 packet-based features will be obtained from the given flow.

Flow-based Computing Approach: For this approach, we first construct a single byte array by aggregating the payload of all packets available in a given flow. We next compute five metrics, namely 4-bit entropy, 8-bit entropy, arithmetic mean, standard deviation, and chi-square for that single array. Hence, a total of 5 flow-based features will be obtained per given flow.

C. Content Inference

We now briefly analyze the prediction power of our features (explained in §III-B) across the three classes of contents (transparent, encrypted, and encoded) in our labeled dataset. For the purpose of illustration, we focus on the first 6KB worth of contents in each flow. Later in §IV, we will analyze the impact of content sizes (from the first 2KB to the first 10KB) to determine the best option, balancing the inference accuracy against the computing cost. Let us begin with Fig. 2 where predictive power of two representative features are demonstrated. Fig. 2(a) shows the violin plot for the maximum 8-bit entropy value computed across all packets of each flow (discussed in the previous subsection). It can be seen that majority of encrypted and encoded flows have larger entropy values compared to transparent flows. In fact, 99.2% of encrypted flows and almost 100% of encoded flows have maximum 8-bit entropy value larger than 6; while this feature is less than 6 for 99.99% of transparent flows. Only 14 flows (`SMB`) have a maximum 8-bit entropy value larger than 6.

We inspected those `SMB` flows and found that they include read/write commands to the `SMB` server for target files. By looking at the payloads of those flows we noticed that they indeed contain random byte values (thus, larger entropy) – the client generates random bytes, and then requests the server to write those values to target files, and finally requests the server

TABLE II
PERFORMANCE OF THE MODEL VARIES BY FLOW SIZE ACROSS THREE CLASSES.

	Prediction accuracy			
Flow size (KB)	**transparent**	**encrypted**	**encoded**	average
2	87%	97	80%	88%
4	99%	97	78%	91%
6	99%	95	94%	96%
8	99%	96	93%	96%
10	99%	94	94%	96%

TABLE III
PERFORMANCE OF THE MODEL FOR ACCEPTED PREDICTIONS (CONFIDENCE SCORE≥ 0.8).

	Prediction accuracy			
Flow size (KB)	**transparent**	**encrypted**	**encoded**	average
2	99%	99%	86%	95%
4	99%	99%	90%	96%
6	99%	99%	99%	99%
8	99%	99%	100%	99%
10	99%	99%	99%	99%

to read those stored values. We verified this by inspecting the source code of Samba torture at [27]. Moving to Fig. 2(b), we observe how flow-based arithmetic mean can predict the three classes to a great extent. Though encrypted and encoded flows seem to overlap slightly, there exist certain distinct patterns for each class. A vast majority (97%) of transparent flows have their arithmetic mean in the range of $[10, 90]$; this metric for more than 80% of encrypted falls in the range of $[90, 115]$; and, about 75% of encoded flows have this value in the range of $[115, 140]$.

As mentioned earlier in this section, we analyzed the first 6KB worth of flow contents to compute features. Also, for a flow with a total content size less than 6KB, its entire content is considered. While the inclusion of small flows did not deteriorate the prediction power of entropy and arithmetic mean, it significantly impacted the chi-square metric. It has been shown [33] that for chi-square to give a valid output, the minimum expected value of each discrete category should be more than five. In our context, as we have 256 categories of byte values, hence we need at least $6 \times 256 = 1536$ bytes for each flow to get a reliable output from the chi-square test. Therefore, chi-square may not be a good indicator for flows with payload size less than 1.5KB.

Fig. 3 shows the impact of payload size on the measure of chi-square. We plot the distribution chi-square values for all flows in Fig. 3(a) and the same distribution for flows larger than a threshold in (*i.e.,* 1.5KB) in Fig. 3(b). Applying this threshold excluded about 93% of transparent flows in our dataset – this measure was relatively lower for encrypted (14%) and encoded (1%) flows. By comparing plots in Figures 3(a) and 3(b), it can be seen that excluding smaller flows helps to distinguish various classes – a vast majority of transparent flows have their chi-square measure over 10000, encrypted flows tend to give chi-square measures centered around 2000 and 7000 (the two peaks), and encoded flows have their chi-square measures uniformly spread in a wide range $[100, 5000]$.

We saw in Figures 2(a), 2(b), and 3 that transparent flow contents seem to have more distinct features compared to encrypted/encoded flows; while, certain overlaps are visible between characteristics of encrypted and encoded flow contents. One may attempt applying deterministic thresholds to assess whether a flow carries transparent or opaque contents (a binary classification). However, distinguishing encrypted from encoded flows is a nontrivial task with 17 features (*i.e.,* dimensions). Instead, machine learning-based models have proven to be effective in analyzing all features and finding an optimal combination of features to characterize unique patterns of each content class. Among various options of learning algorithms, decision tree-based models seem to be a good fit for our problem (a relatively small multi-class classifier), given their ability to create clear boundaries across features and assign distinct regions per class.

IV. EVALUATION RESULTS

In this section, we evaluate the efficacy of our method by generating a machine learning model trained on a subset of our data. Given a content class, we use some protocols for training and the remaining protocols for testing. This is done to reduce the chance of overfitting for our model. For training, we use `NTP` and `DNS` from transparent flows, IoT `TLS` and `AES` from encrypted flows, and `RTP`, `gzip`, `PNG`, and `MP4` for encoded flows. The rest of flows in Table I will be used to test the trained model. It is importation to note that those 626 `TLS` web flows are only present in our testing dataset. In total, we have 99,297 and 24,198 flows for training and testing, respectively.

For the learning algorithm, we choose Random Forest as it has shown reasonable accuracy and explainability in the existing literature of network traffic analysis. We train our model through a ten-fold cross-validation process to find the best combination of hyper-parameters for the final model. We tune the following parameters of our Random Forest model: number of decision trees, maximum depth of trees, and maximum features to consider at each split.

Since the amount of data analyzed in each flow can affect the quality of inference, we experiment with different flow sizes and quantify the model accuracy accordingly. Note that the flow size is referred to as an upper cap for the number of payload bytes we process in each flow – we used 6KB flow size for our preliminary data analysis (§III-C) and insights presented in Figures 2 and 3. Table II shows the performance of the model (the accuracy measure of predicting testing instances) with five different flow sizes ranging from 2KB to 10KB.

It can be clearly seen that having 4KB of contents for a transparent flow is sufficient for the model to correctly predict its class – the prediction accuracy saturates (99%) at flow size of 4KB for the transparent class (second column in Table II). Moving to encrypted flows, the accuracy seems to slightly fall from the flow size of 6KB, but the accuracy is relatively acceptable ($\geq 94\%$) for all flow sizes. For encoded flows, the accuracy drops by increasing the flow size from 2KB to 4KB but it becomes relatively stable and acceptable ($\approx 94\%$) after

that. The average accuracy (of all classes) seems to saturate at the flow size of 6KB.

Another observation is that our model faces some difficulties in correctly predicting the class of encoded flows. Encoded flows (mostly `JPEG` ones) are mispredicted as an encrypted class. For the flow size of 4KB, almost half of the `JPEG` flows are misclassified. We observed that the model displays different levels of confidence (a number between 0 and 1) when testing various flow types. Inspired by our recent work [34], we determine lower-bound thresholds (one per class) for the confidence score of our model by applying it to the training dataset. As a result, if the model predicts a class with a confidence lower than the expected threshold, that prediction will be discarded. In other words, we only consider "accepted" predictions (with confidence scores higher than corresponding thresholds). Due to high variations in the model confidence for predicting transparent instances (in the training dataset), we choose a global configurable confidence threshold equal to 0.80 for all content classes. Let us revisit the model performance for accepted predictions (when the model is relatively confident) in Table III. We observe that the prediction accuracy (classes and average) saturates for the flow size of 6KB. It can be seen that 99% of accepted predictions are correct, with only computing up to 6KB worth of data from each flow. Lastly, we found that the model is generally less confident when predicting encoded flows than the other two classes. More than a third (36%) of encoded flows receive less-confident predictions (hence, "discarded"). This measure is about 20% for transparent and encrypted flows.

V. CONCLUSION

MUD has proven to be a powerful standard that enables network operators to reduce the attack surface on an IoT device by formally defining its expected network behavior (whitelisting benign behaviors). In the absence of a provision for limiting/specifying traffic contents exchanged via individual MUD flows, certain malicious network activities can go undetected while still conforming to the MUD profile of connected IoTs. In this paper, we developed PicP-MUD that profiles the information content of payloads in MUD flows. We analyzed more than 100K network flows from 24 protocols belonging to three classes (transparent, encrypted, and encoded) and identified 17 statistical features indicative of their content class. We trained a multi-class classifier that can correctly predict the content type of a flow by analyzing the first 6KB worth of its packet byte values.

REFERENCES

[1] H. Habibi Gharakheili *et al.*, "Cyber-Securing IoT Infrastructure by Modeling Network Traffic," in *Security and Privacy in the Internet of Things: Architectures, Techniques, and Applications*, A. Ismail Awad and J. Abawajy, Eds. John Wiley & Sons, 2021, ch. 6, pp. 151–176.

[2] Fortinet, "Examining Top IoT Security Threats and Attack Vectors," Jun 2021. [Online]. Available: https://tinyurl.com/2p8624t7

[3] Palo Alto Networks, "Are the Security Cameras in Your Organization Safe from Cyber Attacks?" Mar 2021. [Online]. Available: https://tinyurl.com/2p9anezb

[4] Fortinet, "The Challenges of Inspecting Encrypted Network Traffic," Aug 2020. [Online]. Available: https://tinyurl.com/2p8m4tvy

[5] H. Zhang *et al.*, "Detecting encrypted botnet traffic," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013.

[6] Cyber Edge Group, "Cyberthreat Defense Report," 2020. [Online]. Available: https://cyber-edge.com/cdr/

[7] E. Lear *et al.*, "Manufacturer Usage Description Specification," RFC 8520, Mar. 2019. [Online]. Available: https://tinyurl.com/yndykju8

[8] A. Hamza *et al.*, "Verifying and Monitoring IoTs Network Behavior using MUD Profiles," *IEEE TDSC*, vol. 19, no. 1, pp. 1–18, Jan 2022.

[9] ——, "Combining MUD Policies with SDN for IoT Intrusion Detection," in *Proc. ACM IoT Security and Privacy*, Budapest, Hungary, Aug 2018.

[10] ——, "Detecting Volumetric Attacks on IoT Devices via SDN-Based Monitoring of MUD Activity," in *Proc. ACM SOSR*, San Jose, CA, USA, Apr 2019.

[11] Lawrence Berkeley National Laboratory, "LBNL-FTP-PKT," Jan 2003. [Online]. Available: https://tinyurl.com/4xrabnuf

[12] Wireshark, "Sample Captures: Server Message Block (SMB)," Nov 2005. [Online]. Available: https://tinyurl.com/34p7wbst

[13] A. Sivanathan *et al.*, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE TMC*, vol. 18, no. 8, pp. 1745–1759, 2019.

[14] K. Yang *et al.*, "Towards Automatic Fingerprinting of IoT Devices in the Cyberspace," *Computer Networks*, vol. 148, pp. 318–327, 2019.

[15] H. Guo *et al.*, "IP-Based IoT Device Detection," in *Proc. ACM Security and Privacy*, Budapest, Hungary, Aug. 2018, p. 36–42.

[16] S. J. Saidi *et al.*, "A Haystack Full of Needles: Scalable Detection of IoT Devices in the Wild," in *Proc. ACM IMC*, NY, USA, Oct 2020.

[17] R. Doshi *et al.*, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," in *Proc. IEEE SPW*, San Francisco, CA, USA, May 2018.

[18] NIST, "Securing Small-Business and Home Internet of Things (IoT) Devices," May 2021. [Online]. Available: https://tinyurl.com/2p84uwja

[19] A. Panchenko *et al.*, "Website Fingerprinting at Internet Scale," in *Proc. NDSS*, San Diego, CA, USA, Feb. 2016.

[20] V. F. Taylor *et al.*, "AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic," in *IEEE EuroS P*, Saarbrücken, Germany, Mar. 2016.

[21] J. Muehlstein *et al.*, "Analyzing HTTPS encrypted traffic to identify user's operating system, browser and application," in *IEEE CCNC*, Las Vegas, NV, USA, Jan. 2017.

[22] G. Gu *et al.*, "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation," in *Proc. USENIX Security Symposium*, Boston, MA, Aug. 2007.

[23] R. Wang *et al.*, "Steal This Movie: Automatically Bypassing DRM Protection in Streaming Media Services," in *Proc. USENIX Security Symposium*, Washington, D.C., USA, Aug. 2013.

[24] Y. Khodjaeva *et al.*, "Network Flow Entropy for Identifying Malicious Behaviours in DNS Tunnels," in *Proc. ACM ARES*, New York, NY, USA, Aug. 2021.

[25] M. Richardson *et al.*, "On loading MUD URLs from QR codes (work in progress)," Working Draft, IETF Secretariat, Internet-Draft draft-richardson-mud-qrcode-06, Mar 2022. [Online]. Available: https://tinyurl.com/mtbxjcra

[26] A. Maxwell *et al.*, "Methods, systems, and computer readable media for rapid filtering of opaque data traffic," U.S. Patent 9 973 473B2, Mar. 15, 2018.

[27] Smba Team, "SMB Torture." [Online]. Available: https://tinyurl.com/2p83ew7k

[28] NIST, "Recommendation for Block Cipher Modes of Operation," Dec 2001. [Online]. Available: https://tinyurl.com/y3fa5733

[29] S. Cha *et al.*, "Detecting Encrypted Traffic: A Machine Learning Approach," in *Proc. WISA*, Jeju Island, South Korea, Aug. 2016.

[30] C. E. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[31] J. Haggerty *et al.*, "FORWEB: File Fingerprinting for Automated Network Forensics Investigations," in *Proc. e-Forensics*, Adelaide, Australia, Jan 2008.

[32] W.-J. Li *et al.*, "Fileprints: Identifying File Types by n-gram Analysis," in *Proc. IEEE SMC IAW*, Jun 2005.

[33] M. Bland, *An Introduction to Medical Statistics*. Oxford University Press (UK), 2001.

[34] A. Pashamokhtari *et al.*, "Inferring Connected IoT Devices from IP-FIX Records in Residential ISP Networks," in *IEEE LCN*, Edmonton, Canada, Oct 2021.