IoTHunter: IoT network traffic classification using device specific keywords

Pratibha Khandait | Neminath Hubballi | Bodhisatwa Mazumdar

Discipline of Computer Science and Engineering, Indian Institute of Technology Indore, Indore, India

Abstract
With the proliferation of IoT devices, network management and security monitoring are becoming a challenge. For the timely detection of IoT device status and their behaviour, traffic classification methods are used. Herein, IoTHunter, a Deep Packet Inspection based IoT traffic classifier, is described. It extracts unique keywords comprising domain names, device names etc. to identify flows belonging to a particular device. IoTHunter automates the keyword extraction using the frequency of occurrence of words belonging to flows of different devices. To further enhance the performance, IoTHunter combines device specific keywords with MAC address of device for subsequent flow labelling. A publicly available IoT dataset is experimented and a good classification accuracy of it over a range of IoT devices is demonstrated.

1 | INTRODUCTION

Today’s computer networks interconnect large number of devices other than traditional desktops, servers, printers, and mobile phones [1]. It is a common scenario to have multiple Internet of Things (IoT) devices, such as surveillance cameras, home appliances, fire alarms, temperature sensors, etc. connecting to Internet. IoT devices although perform very specific tasks; but they have a networking stack, and can send (and receive) TCP/IP packets to (and from) other devices or servers. In a smart environment comprising homes and offices, a number of unique IoT devices may be installed by different departments. With thousands of IoT devices connected together, it is difficult to identify which devices are connected to the network at a certain point of time. Creating and maintaining a database for connected devices manually is an error-prone and time-consuming process. In general, network management with IoT devices is challenging for two important reasons:

1. Security monitoring for policy enforcement: IoT devices are vulnerable to cyber-attacks. In fact several studies [2–4] have shown that these devices have little or no in-built security (due to resource-constraints) and are more susceptible to such attacks. Thus, for securing the network from cyber-attack, it is important to monitor these devices.

2. Providing meaningful connectivity: In a very heterogeneous environment comprising IoT devices, it is important to provide meaningful connectivity to every device for its proper functioning. Different devices may have different requirements in terms of bandwidth, tolerance to packet delays, jitter, etc. For example, a surveillance camera may need more bandwidth for transmitting live video capture to a server. Other devices such as, fire alarms may not require high bandwidth but are sensitive to delays and reliability of data transmission.

Thus, network administrators require visibility into types of IoT devices connected to the network in a timely manner. This is generally done with traffic classification techniques. In traditional networks traffic classification is handled either by inspecting network flows or by using Deep Packet Inspection [5]. Few of the previous works [6] have argued that IoT traffic classification is challenging because of sheer number of such devices. Some IoT traffic classification methods use machine learning algorithms [7] with flow characteristics of devices for their identification. However, identifying unique flow characteristics with thousands of devices is non-trivial as there may be significant overlap between them with respect to device characteristics and functions [8]. Another possible approach is to explore if these devices (and associated...
network flows) can be identified with Deep Packet Inspection (DPI). Herein, we study the device identification and IoT flow classification with DPI. Our proposed method identifies unique keywords from network flows and uses them to classify IoT network flows subsequently. In particular, we make following contributions here.

(i) We propose a method to automatically extract and identify device specific keywords.
(ii) We describe a flow classification technique which uses device specific keywords to classify network flows originating from IoT devices in a smart environment.
(iii) We present experimental results with an IoT dataset comprising data of 18 IoT devices and report the performance of our system.

The study is organised as follows: In Section 2, we describe prior work related to traffic classification of IoT devices. In Section 3, we explain our proposed system IoTHunter to classify network flows of IoT devices. Experimental results are described in Section 4. Performance comparison of IoTHunter with a machine learning based approach is elaborated in Section 5. Few limitations and possible enhancements of IoT-Hunter are mentioned in Section 6. Finally conclusions are given in Section 7.

2 | RELATED WORK

In this section, we describe some prior work related to classification of IoT traffic.

2.1 | Classification using flow characteristics

Network flows belonging to IoT devices can be identified with flow characteristics. Conventionally attributes, such as number of packets in a flow, frequency of flows, inter arrival time between packets within a flow, size of the packet, etc., are used as unique identifiers for each device type. Both supervised [6,7,9,10], unsupervised [11,12], and deep learning [13,14] methods have been used for classification.

1. Supervised machine learning methods: These methods require labelled feature vectors generated from network flows of each device in order to learn unique characteristics of each device type. The work described in Ref. [7] is one of the first studies to label flows of IoT devices. It uses different machine learning-based classifiers, such as Gradient Boosting Machine (GBM), Random forest, etc., for identifying flows pertaining to different devices. Shahid et al. [9] proposed a technique to recognise IoT traffic using feature vectors with six machine learning classifiers. A multi-stage classifier to classify IoT devices based on network traffic characteristics (port numbers, domain names, cipher suites, flow rate, flow volume, etc.) is proposed in Ref. [6]. Work described in Ref. [15] uses background traffic generated by IoT devices to identify and label flows generated by them. It uses flows generated periodically and extract features which are represented as device fingerprints. These fingerprints are later provided to K-nearest neighbour classifier to identify device type. Another work described in Ref. [16] uses genetic algorithm to extract features from various protocol headers. These features represented as vectors are then used with various machine learning algorithms (DecisionTable, J48 Decision Trees, OneR, and PART) to classify IoT device type. Salman et al. [17] extract flow-based features and use different supervised machine learning classification algorithms for IoT traffic classification and anomaly detection. Their experiments revealed that random forest algorithm gives the best result. Bezawada et al. [18] extract 20 features (17 binary and 3 payload-based) from first five packets of a flow and classify flows with Gradient Boosting algorithm with varied accuracy.

2. Unsupervised machine learning methods: In these methods, feature vectors generated from network flows are grouped based on their similarity and labelled. Labelled clusters are subsequently used for identifying flows belonging to different devices. IoTguard [11] and the works of Eqea et al. [12] are examples of this category. IoTguard uses a self-adaptive classification scheme (using clustering and fuzzy interpolation scheme [FIS]). It uses network activity of the device to predict traffic class (i.e. malicious or benign).

3. Deep learning techniques: These algorithms have multiple layers of learning also known as progressive learning. Each layer extracts transform input data into an abstract representation. Lopez Martin et al. [13] used recurrent neural network (RNN) and convolutional neural network (CNN) algorithms for IoT traffic classification; while Yao et al. [14] used deep learning aided capsule network forming classification mechanism that integrates feature extraction, feature selection, and classification model. The work described in Ref. [19] provides a framework to automatically extract features and learn traffic classes (IoT, non-IoT) using stacked auto-encoders which encodes TCP flow into a fixed-size vector. A feature vector in the form of distribution is generated for every device and these device specific distributions are used for classification of flows originating from various IoT devices.

Lei Bai et al. [20] proposed an LSTM-CNN cascade model to classify IoT traffic. For feature collection, the authors segmented the IoT traffic into sub-traffic flows of fixed time intervals. They extracted six features from an interval: user packet count, user packet length average, user packet length peak, control packet count, control packet average, and control packet peak. A packet is considered as user packet if it is UDP/TCP/HTTP or any other application layer protocol. Extracted features are fed as input to the LSTM-CNN cascade model to identify the device category. This implementation uses SVM adopting a one-VS-all strategy. Results reveal that the performance of the classifier degrades with the increasing device categories.
All previous works discussed above use machine learning and deep learning techniques to classify network flows belonging to IoT devices. Although machine learning algorithms are used for IoT traffic classification, Desai et al. [8], argued that the performance of a machine learning algorithm mainly depends on the identified features. Providing all extracted features as input to a machine learning-based model is not always beneficial as obtaining features adds to the computational cost. Our work IoTHunter to classify IoT traffic differs from existing work in a way that it uses DPI to recognise IoT devices using device-specific keywords rather than depending upon the feature vector.

2.2 | Alternate schemes

Noguchi et al. [21] use profiles generated from communication pattern of each device and subsequently compare these with collected communication patterns from unknown devices in real-time to identify device type. For comparison, they use a simple Euclidean distance-based metric. Authors argue that this kind of similarity comparison is feasible because communication pattern of each device is different.

There are some existing works, which perform security assessment of IoT devices [3] or identify these devices using complementary methods like IP address to which they connect [22]. For example, Sivaraman et al. [3] proposed a scheme to identify security vulnerabilities in IoT devices based on four parameters, namely confidentiality, integrity and authentication, access control, and the ability to withstand reflective-attacks. Although these works are not dealing explicitly with classification but do help in understanding which devices are vulnerable.

Hsu [23] propose to identify class of IoT device rather than specific device name. The idea is motivated by the fact that same category devices exhibit similar traffic behaviour. For example, all IP cameras may exhibit similar traffic flow behaviour. Author used three features namely bytes per second, packet direction, and time since last communication to identify the device category. Limited testing revealed that for few categories of device like IP cameras, the approach has better classification, while for others like temperature and motion sensors it has limited accuracy of prediction.

3 | DEEP PACKET INSPECTION-BASED IoT TRAFFIC CLASSIFICATION

In this section, we describe our proposed technique IoTHunter which is used to detect IoT devices connected in a smart environment. IoTHunter is a content-based IoT traffic classifier. Its goal is to label the traffic flows belonging to different IoT devices.

IoTHunter is a DPI-based network flow classifier which requires labelled flows from different devices to extract device specific keywords. These keywords are subsequently used to label the flows (classify). Its working principle is based on the idea that many devices use specific keywords in their communication. For example, many IoT devices contact regularly the servers maintained by device manufacturers and also use specific domain names for communication and name resolution. These can be used for identifying those devices reliably. A sample snapshot of payload taken from Belkin Motion Sensor device is shown in Figure 1. From this figure, we can clearly see that it is contacting domain belkin.org which can serve as unique identifier for the devices manufactured by Belkin. Once such device-specific keywords are identified they can be subsequently used to identify flows belonging to these devices. The two phases namely extracting keywords from network flows and classification of flows are elaborated in the next two subsections.

3.1 | Extracting device specific keyword

To classify network flows of IoT devices, device specific keywords are used. Hence, to begin with keywords are extracted. Keyword extraction involves operations, such as flow reconstruction, payload extraction, term extraction, filtering uninteresting terms, normalized term frequency calculation, keyword selection modules, as shown in Figure 2. The main operations of this phase are as follows.

1. Flow Reconstruction: IoTHunter classify flows belonging to different devices hence flows are reconstructed by reading packets. Packets belonging to the same flow have five attributes in common. These five attributes are the source IP address, destination IP address, source port number, destination port number, and layer four protocol (TCP/UDP). Flow reconstruction phase make use of the above five attributes to reconstruct the bidirectional network flows belonging to one session. For a TCP flow, all packets sharing common attributes between a SYN and FIN/ACK packets will be part of one flow. In case of UDP as it is a connectionless protocol; a threshold time of inactivity is considered as end of flow between the two devices.

2. Term Extraction: Flows generated in the previous phase are used for term extraction. All the packets belonging to a flow are considered and all the packets payloads are concatenated to generate a flow payload. This concatenated payload portion is used for extracting terms. For this a set of delimiters like newline character, white space and special character are used. For example, if a flow \( f_i \) has payload content ‘I am the first flow’ then the set of terms generated are \{I, am, the, first, flow\}. These terms are used for the subsequent phases.

3. Filtering Unwanted Terms: Some of the terms may not be good candidates for uniquely identifying devices or classifying flows. For example, some of the flows may contain dates, hosts names, and other related details within the payload. These terms are filtered to get a slightly meaningful set of terms for further consideration. This filtering can be automated with dictionary of negative words, a regular
expression-based filtering or can be done manually. In addition to filtering above undesirable terms, we also remove those terms which are common across the devices. The idea is to filter out terms which do not uniquely identify devices. Let the terms of device $D_i$ are $T(D_i) = \{t'_1, t'_2, ..., t'_k\}$ and of device $D_j$ are $T(D_j) = \{t'_1, t'_2, ..., t'_g\}$ then we calculate $T(D_i) \cap T(D_j)$ and remove all terms which are in $T(D_i) \cap T(D_j)$. This is repeated for all pairs of device types considered. Removing overlapping terms helps in reducing false positives.

4. Potential Keyword Selection: In this phase, device specific keywords are extracted and are recorded in the database. The terms left after previous stage filtering are considered for this case which we name as candidate terms. Each candidate term’s frequency of occurrence is calculated by counting how many times, it has appeared in the flows. For each candidate term, normalized term frequency is calculated with respect to the number of flows for that device as in Equation 1.

$$\text{Normalized}_{freq} = \frac{CT_{freq}}{TF}$$  \hspace{1cm} (1)$$

where $CT_{freq}$ = Total frequency for the candidate term, and $TF$ = Total flows used for generating keywords.

Once all candidate terms normalized term frequency is calculated, the mean and standard deviation of frequency values are used for pruning the candidate term set to get the final set of device specific keywords. This is done by setting a threshold on the normalized term frequency on keyword as in Equation 2.

$$\text{Threshold} = \mu + K \times \sigma$$  \hspace{1cm} (2)$$

where $\mu$ is the mean normalized frequency of every term, $\sigma$ is the standard deviation of normalized frequency values and $K$ is a positive constant. All the terms in the candidate term set having the normalized frequency above the Threshold frequency will be the identified as the final set of keywords for that device.

5. Finding Keyword Offset Range: In the subsequent step, we calculate the byte offset values of the keywords within the reconstructed flow payload. It is also worth noting that keyword byte offset calculation can be combined with term extraction phase itself by recording the offset values of terms (this method is adopted in Algorithm 1) to avoid second pass on payload screening.

Let the keyword set of a device be $K = \{k_1, k_2, ..., k_n\}$ then for each $k_i \in K$ find whether $k_i$ appears in flow $f_j \in F$ (where $F$ is the total flows). If $f_j$ contains $k_i$ then record the byte offset (from beginning) of $k_i$. This is repeated for $\forall \ k_i$ and $f_j$ resulting into a series of offset values for every $k_i$. Thus if $k_i$ appears in $m \leq n$ flows, then at the maximum $m$ distinct offset values will be generated $k_i(\text{Offset}) = [O_1, O_2, ..., O_m]$. Using these distinct offset values, we extract the minimum and maximum offset values $[k_i(\text{MinOffset}), k_i(\text{MaxOffset})]$. For example, if a keyword $k_1$ appears in three flows $f_1, f_2$ and $f_3$ at byte offset values 30, 32, 35 then the minimum and maximum values will be 30 and 35, respectively.

Algorithm 1 summarises the details of device-specific keyword extraction described above. Input to the algorithm is a set of packets of a device $D$ and negative word database. At the end of flow processing, it generates a set of keywords to be used for classification subsequently. The algorithm begins with flow reconstruction and for each flow, it extracts terms from flow payload by parsing payload using delimiters. At the same time, it records the byte offset values that are generated. Once all the flows are processed and terms are extracted it filters the
terms using filter database, generates unique terms and their frequency. Using the frequency of terms, it calculates the normalized term frequency. Subsequently derives a normalized frequency threshold for keyword selection using mean and standard deviation. If the normalized term frequency of any term is more than this threshold it is denoted as keyword and adds to keyword list along with its min and max byte offset values which is the final output of this algorithm.

**Algorithm 1: IoTHunter Training Algorithm**

INPUT: Set of packets $P_1, P_2, ... P_F$ for IoT Device $D$

INPUT: $FD$: Filter Database

OUTPUT: Candidate Keywords for Device $D$

1: Flows ← FlowReconstruct($P_1, P_2, ... P_F$)
2: $N ← $ sizeof(Flow)
3: TermSet ← $Φ$
4: OffsetRange ← $Φ$
5: for $i = 1$ to $N$ do
6:   $[Terms, Offset] ← GetTerms(Flow[i])$
7:   TermSet ← TermSet $∪$ Terms
8:   OffsetRange ← OffsetRange $∪$ Offset
9: end for
10: CandidateSet ← FilterTerms(TermSet, $FD$)
11: for each Word $∈$ CandidateSet do
12:   if Map.contains(Word) then
13:     Frequency[Word]++
14:   else
15:     Frequency[Word] = 1;
16: end if
17: end for
18: UniqueWordCount ← $SizeOf(Frequency)$
19: for $j = 1$ to UniqueWordCount do
20:   Word = WordAt(Frequency, $j$)
21:   Normalized[$j$] = Frequency[Word]/$N$
22: end for
23: MeanFreq ← ComputeMean(Normalized[$j$])
24: $SD ← ComputeStdDev(Normalized[$j$], MeanFreq)$
25: $MinThreshold = MeanProb + K*SD$
26: for $j = 1$ to UniqueWordCount do
27:   if (Normalized[$j$] $> MinThreshold$) then
28:     Word = WordAt(Frequency, $j$)
29:     MinOffset ← GetMin(Word, Terms OffsetRange)
30:     MaxOffset ← GetMax(Word, Terms OffsetRange)
31:     Keyword.add(Word, MinOffset, MaxOffset)
32: end if
33: end for
34: Output Keyword

3.2 Flow classification

Figure 3 shows the various stages of classification operation. As in the previous phase, in this phase too, flows are reconstructed by reading network packets and words are extracted using the set of delimiters. Flows are classified using one of the following two methods.

1. Identifying Flows with Device Keywords: In this case flows are labelled using the keywords identified in the previous phase. Let $K_{D_k} = \{k_1, k_2, ..., k_n\}$ be the keyword list for device $D_k$ and Keyword = \{ $K_{D_1}, K_{D_2}, ..., K_{D_f}$ \} be the union of the keyword sets of all devices. A test flow $f_i$ is evaluated against all device keywords. If any keyword $k_j ∈ K_{D_k}$ is found in $f_i$ within the lower and upper byte offset range then the flow $f_i$ is labelled as of device type $D_k$. If any flow $f_j$ is classified as any device type then the MAC address of device communicating is stored in a database which is subsequently used for other flow classification or identification from same device.

2. Identifying Flows with Device MAC Address: A flow is identified as belonging to a specific device if an earlier flow sharing the same source MAC address is identified as belonging to that device. As mentioned previously, this requires marking the Source and Destination MAC addresses of packets in a flow for the first flow which contains a keyword. Identifying flows with MAC address has few advantages.

(i) It can speed up the classification as parsing payload is not required. Parsing payload is computationally expensive operation.

(ii) It is worth noting that some of the flows may not contain keywords particularly if the IoT device has more than one set of flows and keywords can be seen in only one set of flows. By using MAC address such flows can also be classified.

(iii) Some of the devices may have both encrypted and non-encrypted flows. Those flows which are not encrypted may contain keywords. Using these non-encrypted flows, other flows (encrypted) can also be classified. This enhances the classification coverage.

Algorithms 2 and 3 describe the operations involved in flow classification. Algorithm 2 takes a set of network packets as input and labels the flows as belonging to various IoT devices. It starts with flow reconstruction and consults keyword database for different devices. If any of the keywords in the database is found in the flow within its minimum and maximum offset range, it is labelled as a flow of that device. It also logs the corresponding MAC addresses for subsequent flow classification. In the absence of any keyword, Algorithm 2 invokes Algorithm 3 to check if the MAC addresses of packets in the flow are identified earlier as belonging to a particular device and hence label the flow.

**Algorithm 2: IoTHunter Flow Classification Algorithm**

INPUT: Set of testing packets $P_1, ... P_l$

INPUT: Keyword = \{ $K_{D_1}, K_{D_2}, ..., K_{D_l}$ \} Device specific keyword set for all devices

OUTPUT: Set of labelled flows
In this subsection, we describe the asymptotic complexity of every module of IoTHunter. Table 1 lists the asymptotic complexities of different modules mentioning training and testing part. Flow reconstruction module reads packet header information from each packet and adds the packet with its associated flow. For $F$ flows in the trace (or live collection) and $P$ packets, its complexity is $O(F \times P)$. This is because every packet needs to be checked against every flow to decide whether it needs to be a part of an ongoing flow or a new flow. Payload parsing, term extraction module reads the entire payload to generate a set of terms. For a payload of size $B$ bytes, the parsing operation takes $O(B)$ time. For $F$ flows and $J$ candidate terms, Frequency generator has the complexity of $O(F \times J)$. Normalized term frequency calculator requires reading the candidate term set once and for $J$ number of terms it will have a complexity of $O(J)$. Byte offset calculation has a complexity of $O(F \times SK)$ for $F$ flows and $SK$ number of final selected keywords. $SK$ can be of the size of candidate term set thus its complexity can be rewritten as $O(F \times J)$. Thus the total training complexity is $O((F \times P) + (F \times J) + (F \times J))$ which is $O((F \times P) + (F \times J))$.

The MAC Detect module compares MAC addresses of a packet against address in the MAC store. It requires scanning the MAC store once, and hence has complexity of $O(M)$ where $M$ is the size of MAC store. The entire flow classification requires scanning the keyword store for classifying each flow. This in addition calls MAC Detect procedure when keyword is not found in keyword store. Thus, the overall complexity of this module is $O((F \times K) + (F \times M))$ where $K$ is the size of keyword store and $M$ is the MAC address database store.

### 3.3 Asymptotic complexity

In this subsection, we describe the asymptotic complexity of every module of IoTHunter. Table 1 lists the asymptotic complexities of different modules mentioning training and testing part. Flow reconstruction module reads packet header information from each packet and adds the packet with its associated flow. For $F$ flows in the trace (or live collection) and $P$ packets, its complexity is $O(F \times P)$. This is because every packet needs to be checked against every flow to decide whether it needs to be a part of an ongoing flow or a new flow. Payload parsing, term extraction module reads the entire payload to generate a set of terms. For a payload of size $B$ bytes, the parsing operation takes $O(B)$ time. For $F$ flows and $J$ candidate terms, Frequency generator has the complexity of $O(F \times J)$. Normalized term frequency calculator requires reading the candidate term set once and for $J$ number of terms it will have a complexity of $O(J)$. Byte offset calculation has a complexity of $O(F \times SK)$ for $F$ flows and $SK$ number of final selected keywords. $SK$ can be of the size of candidate term set thus its complexity can be rewritten as $O(F \times J)$. Thus the total training complexity is $O((F \times P) + (F \times J) + (F \times J))$ which is $O((F \times P) + (F \times J))$.

The MAC Detect module compares MAC addresses of a packet against address in the MAC store. It requires scanning the MAC store once, and hence has complexity of $O(M)$ where $M$ is the size of MAC store. The entire flow classification requires scanning the keyword store for classifying each flow. This in addition calls MAC Detect procedure when keyword is not found in keyword store. Thus, the overall complexity of this module is $O((F \times K) + (F \times M))$ where $K$ is the size of keyword store and $M$ is the MAC address database store.

### 4 EXPERIMENTS

In this section, we describe the experiments performed to evaluate IoTHunter. To assess the performance of classification we use Recall as a metric. Recall is calculated as in Equation 3.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

(3)

where TP denotes the network flows correctly classified by IoTHunter and FN denotes the flow either misclassified as
We use both keywords and MAC addresses of devices to classify flows. We denote the recall achieved with keywords as \( \text{Recall}_{kw} \) and recall achieved with both keyword and MAC addresses as \( \text{Recall}_{kw+MAC} \).

Dataset: For our experiments with IoTHunter, we used a publicly available dataset [24]. This dataset comprises of network traffic generated by 28 IoT devices collected over a six-month period. We categorised these IoT devices into three groups based on the type of data they carry:

(a) Fully Encrypted: These devices use encryption right from the beginning of their interaction and hence have no characters in human readable form in their network flows. There are ten such devices in the dataset. We performed our experiments excluding such devices from the dataset. Devices such as Drop Camera fall under this category.

(b) Partially Encrypted: These devices have a mix of encrypted and plain text flows with majority of flows being encrypted. IoT devices such as Blipcare BP Metre, LifX Bulb, Nest Smoke Sensor, Netatmo Weather Station, Smart Things, TP-Link Camera, TP-Link Plug, Pixstartphoto frame, Netatmo Camera, and Amazon Echo Hub are examples of this type.

(c) Plaintext: These devices have maximum flows in plain text form with some encrypted flows. Devices such as Belkin Motion sensor, Belkin Switch, Samsung Smart Cam, HP printer, ihome plug, Withings baby Monitor, Triby speaker, and Withings sleep sensor are examples of this category.

Pre-processing: The dataset consists of network traffic in the form of .pcap files with one file generated per day. To get the per device traffic, we filtered traffic based on the MAC addresses as mentioned by the authors of Ref. [6] using tcpdump tool [25]. Subsequently, all flows belonging to a device type are merged which gave us device-specific .pcap files.

Implementation: We implemented the keyword extraction and flow classification module as Java programs with jNetPcap library which allows to read network trace files in the form of .pcap files and parse packets for reading payloads. The keyword search operation required for searching a keyword in the flows is implemented through a method provided by jNetPcap library (payload.contains(keyword)). Using our implementation of IoTHunter, we processed flows of 18 device types which are using partial encryption or no encryption for our experiments. For IoT devices with small classification accuracy, we use the corresponding device MAC addresses for classification. MAC addresses of 18 devices we considered for our experiment are shown in Table 2.

### Table 1: Module-wise asymptotic complexity of IoTHunter

| Module Name                        | Complexity | Description                                                                 |
|-----------------------------------|------------|-----------------------------------------------------------------------------|
| Flow reconstruction               | \( O(F \times P) \) | \( F = \) number of flows, \( P = \) number of packets in the trace. |
| Keyword extraction or training    | \( O(b) \) | \( B = \) size of bytes in payload                                          |
| (i) Payload Term Extractor        | \( O(T \times FD) \) | \( T = \) Term set size and \( FD = \) Filter database size                  |
| (ii) Term filtering               | \( O(F \times J) \) | \( F = \) number of flows, \( J = \) size of candidate term set            |
| (iii) Frequency calculator        | \( O(J) \) | \( J = \) size of candidate term set                                        |
| (iv) Normalized Term frequency calculator | \( O(M) \) | \( M = \) size of MAC store                                                 |
| (v) Byte offset calculation       | \( O(F \times SK) \) | \( F = \) number of flows, \( SK = \) size of selected keyword set          |

| Flow classification | \( O(F \times K) + (F \times M) \) | \( F = \) number of flows, \( K = \) size of keyword store, \( M = \) MAC store database size |

### Table 2: IoT devices and associated MAC address

| IoT Devices                          | MAC Address   |
|-------------------------------------|---------------|
| Amazon Echo (ACH)                   | 44:65:0c:56:ccc:3d |
| Belkin Motion Sensor (BMS)          | ec:1a:59:83:28:11 |
| Belkin Switch (BS)                  | ec:1a:59:79:48:89 |
| Blipcare BP Meter (BBM)             | 74:6a:89:00:2e:25 |
| HP Printer (HPPr)                   | 70:5a:0fe:49:bc:0 |
| Ihome Plug (IP)                     | 74:c6:3b:29:7d:1d |
| LifX Bulb (LB)                      | d0:73:65:01:83:08 |
| NEST Smoke Sensor (NSS)             | 18:b4:30:25:be:ce:4 |
| Netatmo Camera (NC)                 | 70:ee:50:18:34:43 |
| Netatmo Weather Station (NWS)       | 70:ee:50:3d:68:ac |
| Pixstart Photo Frame (PSPF)         | e0:76:dc:33:bb:85 |
| Samsung Smart Cam (SSC)             | 00:16:6c:0b:00:88 |
| Smart Things (ST)                   | d0:52:a8:06:7:5e |
| TP-Link camera (TPLC)               | f4:26:d3:51:51 |
| TP-Link Plug (TPLP)                 | 50:c7:ff:00:56:39 |
| Triby Speaker (TS)                  | 18:b7:9c:02:20:44 |
| Withings Baby Monitor (WBM)         | 00:24:e4:10:ec:4e |
| Withings Scale (WS)                 | 00:24:e4:1b:6f:96 |
| Withings Sleep Sensor (WSS)         | 00:24:e4:20:28:ce |

For IoT devices with small classification accuracy, we use the corresponding device MAC addresses for classification. This dataset comprises of network traffic generated by 28 IoT devices collected over a six-month period. We categorised these IoT devices into three groups based on the type of data they carry:
Results: For our experiments, we divided the device specific network flows into two parts in the approximate ratio of 50:50. The first part is used for extracting keywords and second part is used for testing. Table 3 shows the details of the dataset in terms of number of flows and also the size of trace files used for training and testing parts, respectively.

Using the training portion of dataset, we selected keywords for all the 18 devices using the mean and standard deviation values of the terms extracted from each device type. Figure 4 shows normalized frequency histogram and the keyword selected for the device *Pix‐start photo frame*. Figure 4a shows the frequency histogram of few sample terms (incomplete) along with their normalized frequency values. The red horizontal line in this figure is the mean frequency value of the terms shown. Figure 4b shows the final set of selected keywords using the standard deviation and constant $K$ set to two for this device.

Experiment I: We performed the first set of experiments considering all device-specific keywords with their associated byte offset values as described in Algorithm 1 and Algorithm 2. In Figure 5a,b, we have shown the recall rate of IoTHunter considering the keywords byte offset for the 18 IoT devices. As evident from the Figure 5a, the keyword-based identification has the highest recall rate ($\text{Recall}_{KW}$) for *Withings Sleep Sensor* with a value of 0.96. The total number of network flows for this IoT device is 3294 in the testing part. The smallest recall rate $\text{Recall}_{KW}$, was exhibited by *Netatmo Weather Station* with value of 0.008. The total number of network flows for these IoT devices is 1213. This small accuracy for this device and few other devices is due to

a. The keywords identified appear only in a fraction of flows. This is due to the reason that more than one type of flows appear for a device type and the corresponding keywords are found in only one type of flow. IoT devices perform specialised tasks for which they communicate with limited and specific domains and servers to avail the services. IoT traffic though encrypted, contains domain names, and the server name in plain text, which can be treated as unique to a specific device. But these keywords may not appear in the payload of each flow, which leads to poor detection performance for keyword only classification.

b. Some of the devices for privacy concerns use SSL/TLS encryption allowing them to generate partially encrypted data (with only domain names and/or device name being in plain text format).

c. The flows from a particular device are either misclassified as flows of other device due to the presence of keyword from other device type.

From Figure 5a, we can notice that only keyword-based classification is not having recall which is in an acceptable range. To improve the performance (particularly for some devices), we subsequently performed classification using MAC address along with the keywords, as described in Algorithm 2. Figure 5b, shows the detection performance of IoTHunter

| Device name                  | Number of flows for training | Size of training (MB) | Number of flows for testing | Size of Testing (MB) |
|------------------------------|-------------------------------|-----------------------|----------------------------|----------------------|
| Amazon Echo                  | 13,355                        | 83.6                  | 12,382                     | 64.4                 |
| Belkin Motion Sensor         | 25,715                        | 191.4                 | 26,197                     | 195.7                |
| Belkin Switch                | 15,475                        | 143.7                 | 16,396                     | 148.5                |
| Blipcare BP Meter            | 8                             | 0.0163                | 5                          | 0.0162               |
| HP Printer                   | 318                           | 12.7                  | 442                        | 10.9                 |
| iHome Power Plug             | 145                           | 4.2                   | 50                         | 3.2                  |
| LiFX Bulb                    | 5103                          | 8.9                   | 5726                       | 10.2                 |
| NEST Smoke Sensor            | 77                            | 0.5019                | 86                         | 0.5543               |
| Netatmo Camera               | 3122                          | 112.9                 | 2799                       | 103.6                |
| Netatmo Weather Station      | 1185                          | 23.3                  | 1213                       | 24                   |
| Pix-star Photo Frame         | 3545                          | 7                     | 3066                       | 7.6                  |
| Samsung Smart Cam            | 30,555                        | 223                   | 17,980                     | 176                  |
| Smart Things                 | 2872                          | 24.2                  | 3146                       | 24.9                 |
| TP-Link Camera               | 1191                          | 7.1                   | 1305                       | 46.9                 |
| TP-Link Plug                 | 695                           | 3.1                   | 656                        | 2.8                  |
| Triby Speaker                | 222                           | 12.2                  | 266                        | 13.6                 |
| Withings Baby Monitor        | 200                           | 0.33                  | 104                        | 0.1749               |
| Withings Sleep Sensor        | 3397                          | 28.9                  | 3294                       | 27.7                 |
classifier based on keyword followed by MAC address-based classification with byte offset as constraint. These results demonstrate that keyword-based classification along with MAC address-based classification significantly improves the detection performance (We assume that MAC addresses are not spoofed). As MAC address-based classification happen only when the keyword based classification fails, that is when no keyword of any device is found in the payload. Hence, keyword-based classification serves as a baseline classifier which is improved with MAC address-based classification. To understand the extent of improvement in classification accuracy of IoTHunter, we depict the classification results in the form of confusion matrices for both cases (only keyword-based and keyword with MAC address-based classification).

Tables 4 and 5 are the two confusion matrices for these two cases. In these tables, column headers include the name of device and number within bracket indicates the number of flows used in testing. For example, in Table 4, the second column Belkin Motion Sensor (26,197) indicates there are 26,197 flows in testing dataset of this device. We can notice from Table 5 that in all the cases the number of flows classified correctly has increased along with a significant reduction in number of unclassified cases. For example, from Table 4, we can notice that 2644 flows out of 3066 for device Pixstart photo frame are classified as Pix-start photo frame and 389 are identified as Belkin Switch and remaining 33 are not classified. From Table 5, we can notice that a similar number of flows (389) are classified as Belkin Switch and 2656 flows are correctly identified as Pix-start photo frame cases. Number of unclassified cases has come down to 21 instead of 33 in previous case. In this instance, although the improvement is marginal but for other cases keyword combined with MAC address-based classification is significantly reducing the number of unclassified instances. For example, in case of Lifx Bulb 5667, unclassified instances have been reduced to just 34. It is easy to see from the two tables that similar scale improvements are made across all the devices. In general, keyword followed by MAC address-based classification makes the classifier robust as encrypted flows originating from a device which has been already identified can also be classified correctly with one caveat that MAC-based classification is dependent on the correct classification done in the first place. If the first few
| Keywords | IoT Devices flow | BMS (26197) | BS (16396) | SSC (17980) | HPPr (442) | IP (50) | TS (266) | WSS (3294) | BBM (5726) | NSS (1213) | NWS (3146) | SST (1305) | TPLC (656) | TPLP (104) | WBM (3066) | PSPF (2799) | NC (12382) | ACH (2799) |
|----------|-----------------|--------------|------------|-------------|------------|--------|-------|----------|----------|---------|----------|---------|---------|---------|--------|---------|---------|---------|---------|
| Belkin Motion Sensor | 9694 | 0 | 8728 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Belkin Switch | 13,059 | 13,155 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 389 | 0 | 0 |
| Samsung Smart Cam | 0 | 0 | 8822 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| HP-Printer | 0 | 0 | 0 | 316 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Ihome Plug | 0 | 0 | 0 | 0 | 39 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Triby Speaker | 0 | 0 | 0 | 0 | 172 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Withings Sleep Sensor | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3178 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Blipcare BP Meter | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Lifx Bulb | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Nest Smoke Sensor | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 54 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Netatmo Weather Station | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Samsung Smart Things | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 29 | 0 | 0 | 0 |
| TP-Link Camera | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1016 | 0 | 0 | 0 |
| TP-Link Plug | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 28 | 0 | 0 | 0 |
| Withings Baby Monitor | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 90 | 0 | 0 | 0 |
| Pixstar Photo Frame | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2644 | 0 | 0 |
| Netatmo Camera | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1209 |
| Amazon Echo Hub | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10,027 |
| Not classified | 3444 | 3241 | 424 | 126 | 11 | 94 | 104 | 3 | 5667 | 32 | 1203 | 3117 | 289 | 628 | 14 | 33 | 1590 | 2355 |
| Keywords                  | IoT Devices flow |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|--------------------------|------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
|                          | BMS (26197)      | BS     | SSC    | HPP_t | IP     | TS     | WSS    | BBM    | LB     | NSS    | NWS    | SST    | TPLC   | TLP    | WBM    | PSPF   | NC     | ACH    |
| Belkin Motion Sensor     | 10,073           | 0      | 8731   | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| Belkin Switch            | 15,927           | 16,036 | 6      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 389    | 0      | 0      |
| Samsung Smart Cam        | 0                | 0      | 8965   | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| HP-Printer               | 0                | 0      | 0      | 372    | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| ihome Plug               | 0                | 0      | 0      | 48     | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| Triby Speaker            | 0                | 0      | 0      | 266    | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| Withings Sleep Sensor    | 0                | 0      | 0      | 3256   | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| Blipcare BP Meter        | 0                | 0      | 0      | 0      | 4      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| Lifx Bulb                | 0                | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 5692   | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| Nest Smoke Sensor        | 0                | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 65     | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| Netatmo Weather Station  | 0                | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 1134   | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| Samsung Smart Things     | 0                | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 2845   | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| TP-Link Camera           | 0                | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 1283   | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| TP-Link Plug             | 0                | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 574    | 0      | 0      | 0      | 0      | 0      | 0      |
| Withings Baby Monitor    | 0                | 0      | 0      | 0      | 12     | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 102    | 0      | 0      |
| Pixstart Photo Frame     | 0                | 0      | 10     | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 2656   | 0      |
| Netatmo Camera           | 0                | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 2745   | 0      | 0      |
| Amazon Echo Hub          | 0                | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 12,369 | 0      | 0      | 0      |
| Not classified            | 197              | 360    | 268    | 70     | 2      | 0      | 26     | 1      | 34     | 21     | 79     | 301    | 22     | 82     | 2      | 21     | 54     | 13     |
flows are misclassified and a subsequent flow does not have a keyword, IoTHunter uses previously stored MAC address for classification which will add to the misclassification.

Experiment II: To study the classification performance of IoTHunter without byte offset values used, we performed the second set of experiments without using offset values of keywords but only the presence of keywords. The rationale for this experiment is to understand the impact of byte offset values of keywords in classification. Figure 6 shows the recall rate for only keyword based and keyword plus MAC address based flow classification for this experiment.

From Figures 5 and 6, we can notice that offset-based classifier achieved higher recall rate compared with non-offset-based classifier. The reason behind this can be understood from confusion matrices in Table 4 and Table 6 for only keyword-based classification and Table 5 and Table 7 for keyword plus MAC address-based classification. This is slightly counter intuitive as relaxing the offset constraint, we expect a higher recall rate. However, except in one case (Samsung smart things) in all other cases the recall rate is decreasing. This decrease is due to the misclassifications with one device’s keyword being found in another device flow. With no restriction on the location of keyword these are classified as other device type. Further, removing byte offset constraint increases the number of misclassifications when MAC address-based classification is combined as previous misclassifications influence the subsequent flow classification. This fact is supported by other entries in the confusion matrices in Tables 4 and 6 for keyword-based classification and Tables 5 and 7 for keyword plus MAC address-based classification, respectively. The importance of offset-based classifier can be inferred by comparing Figures 5b and 6b. Using offset-based classifier flows from Belkin motion sensor, HP printer and Withings sleep sensor are classified with the recall rate of 0.38, 0.84, and 0.98, respectively. Using non-offset-based classifier Belkin motion sensor, HP printer, and Withings sleep sensor exhibit poor recall rate of 0.01, 0.37, and 0.49, respectively. This is because the non-offset based classifier is less restrictive, and a keyword is matched if it appears anywhere in the entire payload. This relaxed version without the byte offset constraint increases the false positives and results into lower recall rates.

5 | COMPARISON

We compare IoTHunter with a recent work of Bazawada et al. [18]. They described a method to identify IoT devices in a smart environment using a collection of fingerprints representing the behavioural profile of the IoT devices. An unknown device is identified as D_i if a fingerprint corresponding to D_i matches the already captured behavioural profile B_i of the device D_i. These behavioural profiles are generated using communication patterns of devices. In particular, five packets from each device session are collected, and 20 features are extracted from each packet to generate a feature vector of one hundred features. This includes features collected from packet header and also from payload. Packet header features are binary features like whether the packet is HTTP, HTTPS, or ICMP/ARP, etc. In all, there are 17 such features extracted from every packet header. Payload-based features include size of payload, entropy of payload, and TCP window size (set to zero for UDP packet) of packet in question. Each feature vector represents a fingerprint, and a series of such vectors are collectively called the behavioural profile of an IoT device. Machine learning algorithms are then trained and tested using the generated feature vectors. Authors experimented with several machine learning algorithms, including K-nearest neighbour, Decision trees, Gradient boosting, and majority voting.

In Ref. [18], the authors claimed that the gradient boosting regression algorithm performed better than others and reported the results for each experiment using the same. We replicated the authors’ work on our dataset for classification performance comparison. As mentioned earlier, 20 features are extracted from each packet and five such packets are taken from each session. From our experiments, we noticed that some of the sessions of IoT devices have less than five packets. Table 8 shows the distribution of sessions having one to five or more number of packets for every device. We can notice that

(a) Recall of IoTHunter with Keyword (Without Offset).

(b) Recall of IoTHunter with Keyword and MAC Address (Without Offset).
| Keywords                          | BMS (26197) | BS (16396) | SSC (17980) | HPPr (442) | IP (50) | TS (266) | WSS (3294) | BBM (5726) | NSS (86) | NWS (1213) | SST (3146) | TPLC (1305) | TPLP (656) | WBM (104) | PSPF (3066) | NC (2799) | ACH (12382) |
|----------------------------------|-------------|------------|-------------|------------|---------|---------|-----------|-----------|---------|------------|-----------|-------------|------------|----------|-------------|--------|----------|
| Belkin Motion Sensor             | 311         | 0          | 66          | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| Belkin Switch                    | 22,601      | 13,218     | 8847        | 268        | 2       | 5       | 5         | 0         | 0       | 0          | 0         | 0           | 5          | 0        | 1           | 518    | 0        |
| Samsung Smart Cam                | 0           | 0          | 8741        | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 1          | 0        | 0           | 0      | 0        |
| HP-Printer                       | 0           | 0          | 88          | 72         | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| Ihome Plug                       | 0           | 0          | 0           | 40         | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| Triby Speaker                    | 0           | 0          | 0           | 0          | 171     | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| Withings Sleep Sensor            | 0           | 0          | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| Blipcare BP Meter                | 0           | 0          | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| Lifx Bulb                         | 0           | 0          | 0           | 0          | 0       | 0       | 0         | 69        | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| Nest Smoke Sensor                 | 0           | 0          | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 56         | 0         | 0           | 38         | 0        | 0           | 0      | 0        |
| Netatmo Weather Station          | 592         | 592        | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 10        | 0           | 0          | 0        | 0           | 0      | 0        |
| Samsung Smart Things             | 0           | 0          | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| TP-Link Camera                   | 0           | 0          | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 978       | 0           | 0          | 0        | 0           | 0      | 0        |
| TP-Link Plug                     | 0           | 0          | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 28          | 0          | 0        | 0           | 0      | 0        |
| Withings Baby Monitor            | 0           | 0          | 10          | 0          | 0       | 0       | 181       | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 2517        | 0      | 0        |
| Pixstart Photo Frame             | 0           | 0          | 31          | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 1209   | 0        |
| Netatmo Camera                   | 374         | 31         | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 10,032 | 0        |
| Amazon Echo Hub                  | 0           | 0          | 0           | 0          | 0       | 0       | 0         | 0         | 0       | 0          | 0         | 0           | 0          | 0        | 0           | 0      | 0        |
| Not classified                    | 2319        | 2555       | 228         | 102        | 8       | 90      | 72        | 2         | 5657    | 30         | 1203       | 3060        | 283       | 628       | 13         | 31     | 1590     | 2348   |
| Keywords                  | IoT Devices flow | BMS (26197) | BS (16396) | SSC (17980) | HPPr (442) | IP (50) | TS (266) | WSS (3294) | BBM (5) | LB (5726) | NSS (1213) | NWS (1305) | SST (3146) | TPLC (656) | TPLP (104) | WBM (3066) | PSPF (12382) | NC (2799) | ACH (12382) |
|--------------------------|------------------|-------------|------------|-------------|-----------|--------|--------|-----------|--------|--------|----------|----------|----------|----------|----------|----------|----------|-----------|-----------|
| Belkin Motion Sensor     |                  | 311         | 0          | 66          | 0         | 0      | 0      | 0         | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         |
| Belkin Switch            |                  | 24,885      | 15,477     | 8947        | 268       | 2      | 5      | 5         | 0      | 0      | 0        | 7        | 0        | 1        | 519      | 0        | 0         | 0         |
| Samsung Smart Cam        |                  | 0           | 0          | 8863        | 0         | 0      | 0      | 0         | 0      | 0      | 0        | 0        | 0        | 1        | 0        | 0        | 0         | 0         |
| HP-Printer               |                  | 0           | 0          | 88          | 165       | 0      | 0      | 0         | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         |
| Ihome Plug               |                  | 0           | 0          | 0           | 0         | 0      | 0      | 46        | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         |
| Triby Speaker            |                  | 0           | 0          | 0           | 0         | 261    | 0      | 0         | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         |
| Withings Sleep Sensor    |                  | 0           | 0          | 0           | 0         | 0      | 0      | 1641      | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         |
| Blipcare BP Meter        |                  | 0           | 0          | 0           | 0         | 0      | 0      | 0         | 4      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         |
| Lifx Bulb                |                  | 0           | 0          | 0           | 0         | 0      | 0      | 0         | 0      | 5706   | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         |
| Nest Smoke Sensor        |                  | 0           | 0          | 0           | 0         | 0      | 0      | 0         | 0      | 65     | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 0         |
| Netatmo Weather Station  |                  | 592         | 592        | 0           | 0         | 0      | 0      | 0         | 0      | 0      | 0        | 1134     | 0        | 151      | 0        | 0        | 0         | 0         |
| Samsung Smart Things     |                  | 0           | 0          | 0           | 0         | 0      | 0      | 0         | 0      | 0      | 0        | 0        | 3036     | 0        | 0        | 0        | 0         | 0         |
| TP-Link Camera           |                  | 0           | 0          | 0           | 0         | 0      | 0      | 0         | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 1130     | 0         | 0         |
| TP-Link Plug             |                  | 0           | 0          | 0           | 0         | 0      | 0      | 0         | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 574      | 0         | 0         |
| Withings Baby Monitor    |                  | 0           | 0          | 0           | 0         | 0      | 0      | 1443      | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 102      | 0         | 0         |
| Pixstar Photo Frame      |                  | 0           | 0          | 0           | 0         | 0      | 0      | 181       | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 2541      |
| Netatmo Camera           |                  | 394         | 37         | 0           | 0         | 0      | 0      | 0         | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 2745      |
| Amazon Echo Hub          |                  | 0           | 0          | 0           | 0         | 0      | 0      | 0         | 0      | 0      | 0        | 0        | 0        | 0        | 0        | 0        | 0         | 12,369    |
| Not classified           |                  | 15          | 290        | 16          | 9        | 2      | 0      | 24        | 1      | 20     | 21       | 79       | 110      | 16       | 82       | 1        | 6         | 54        |

KHANDAIT ET AL.
there are fairly large number of sessions having less than five packets. There are two ways to handle such sessions.

1. Consider only those sessions having at least five packets for classification. This will automatically mark all sessions having less than five packets as unclassified reducing the classification performance.

2. Extract 20 features from every packet and fill the remaining packet features with zeros (representing absent) for the sessions having less than five packets.

We took the second approach for our evaluation and comparison. The classification performance described in the study by Bazawada et al. [18] and IoTHunter is shown in Figure 7. We can notice that IoTHunter performs much better than the work described in Ref. [18]. One point worth noting here is that, Ref. [18] reported results with fivefold cross-validation; however in our evaluation, we omitted this. Some of the salient features that differentiate our work with Ref. [18] or any other machine learning-based classifiers for that matter are

1. IoTHunter being a DPI-based technique can handle varying number of packets in the session. For example, IoTHunter can handle payloads of varying sizes for keyword searching (through flow reconstruction) even if part of keyword is in one packet and remainder is in the subsequent packet. However, this may affect the performance of machine learning-based methods as in Ref. [18].

2. The evaluation method used by Bazawada et al. [18] is not a $K$ class classification, instead it is a binary classification based on one versus rest all. A similar approach is taken by Bai et al. [20]. We believe that this is a serious limitation of the work as our experiments showed unacceptably low classification performance with $K$ class classification with few devices showing less than 10% classification accuracy. In a real network setup, it is not possible to use binary classifiers as it requires to those many binary classifiers as there are devices.

3. IoTHunter can assist in collecting new keywords for new devices while the machine learning algorithms require explicit training.

### TABLE 8 Distribution of packets in training and test flows

| IoT Devices       | # Distribution of Packet Count for Training Flows | # Distribution of Packet Count for Test Flows |
|-------------------|-----------------------------------------------|-----------------------------------------------|
|                   | 1 Packet 2 Packets 3 packets 4 packets 5 or more packets | Total Flows | 1 Packet 2 Packets 3 packets 4 packets 5 or more packets | Total Flows |
| Amazon Echo Hub   | 1918 4422 224 4459 2332 | 13,555 | 1774 4011 171 4275 2151 | 12,382 |
| Belkin Motion Sensor | 0 8 1 6 25,700 | 25,715 | 10 8 1 3 26,121 | 26,197 |
| Belkin Switch     | 1 320 16 58 15,080 | 15,475 | 25 285 20 70 15,996 | 16,396 |
| Blipcare BP Meter | 2 0 2 0 2 | 6 | 0 0 1 0 3 | 4 |
| HP Printer        | 21 3 21 6 139 | 190 | 24 7 2 3 329 | 365 |
| Ihome Plug        | 0 10 1 2 132 | 159 | 0 2 1 2 45 | 50 |
| Lifx Bulb         | 4654 202 86 56 105 | 5103 | 5242 191 113 56 123 | 5725 |
| Nest Smoke Sensor | 9 1 14 3 41 | 68 | 8 4 17 1 46 | 76 |
| Netatmo Camera    | 1194 882 4 1 1041 | 3122 | 1092 776 0 1 930 | 2799 |
| Netatmo Weather Station | 1 1 0 0 1185 | 1187 | 5 4 0 0 1204 | 1213 |
| Pixsart Photo Frame | 2817 94 0 6 628 | 3545 | 2430 89 4 5 538 | 3066 |
| Samsung Smart Cam | 4267 4019 909 6126 15,234 | 30,555 | 2677 2191 435 1422 11,255 | 17,980 |
| Smart Things      | 2841 9 0 0 11 | 2861 | 3004 125 1 1 15 | 3146 |
| TP-Link Camera    | 529 12 13 8 656 | 1218 | 593 64 6 9 633 | 1305 |
| TP-Link Plug      | 536 1 9 7 138 | 691 | 528 3 3 2 118 | 654 |
| Triby Speaker     | 127 11 2 1 91 | 232 | 140 18 0 2 118 | 278 |
| Withings Baby Monitor | 98 0 0 0 101 | 200 | 51 0 1 0 52 | 104 |
| Withings Sleep Sensor | 1681 7 1 6 1702 | 3397 | 1626 10 1 21 1635 | 3294 |
6 | LIMITATION AND POSSIBLE EXTENSIONS

In this section, we discuss few limitation of IoTHunter and also possible extensions to it.

6.1 | Limitations

1. DPI is computationally expensive: IoTHunter although successful in identifying devices and classify network flows, it is based on Deep Packet Inspection. This is a computationally expensive operation and need to be implemented with several optimizations. Some of the works in other domain like security monitoring in IDS, etc., is sometimes implemented in hardware. Our current implementation is only a proof-of-concept code written in Java.

2. Same keyword for more than one device: If there are more than one device-type manufactured by a vendor, same keywords may appear for both the devices which makes it difficult to exactly identify the device type.

6.2 | Possible extensions

As mentioned previously, DPI is a computationally expensive operation as it requires pattern matching. We intend to work on optimising performance of DPI. This is achieved through

1. Reducing the quantum of payload data screened for finding the keywords: Our initial work [26] in this direction has proposed few optimizations.

2. Incorporating heuristic-based optimizations for reducing the overhead of string comparisons: Few works like Hyperscan [27] and DFC [28] implement such optimizations.

3. Implementing DPI on hardware platforms: Customised hardware circuits using FPGA and other similar hardware platforms can be used to implement the DPI operations on it. Several commercial vendors like CISCO have similar products in the market.

4. Machine learning-based IoT traffic classifiers with payload features can be an interesting study particularly if there are multiple keywords and presence of one or more of them might be good enough to label the flow.

7 | CONCLUSION

Here, we proposed IoTHunter which is a keyword-based IoT traffic classification method. It requires labelled flows from each device type and uses DPI to extract right keywords from each device type. These extracted keywords serve as a unique identifiers for a particular device. Words like website names, device names, etc. are the usual candidates for the keywords. IoTHunter combines keyword-based classification with MAC address-based classification to improve the performance. With experiments on a large dataset, we showed that it has a decent accuracy of classification. We also compared our proposed method IoTHunter with a machine learning-based classifier and showed its advantages.

ACKNOWLEDGEMENTS

Pratibha Khandait is financially supported with a fellowship from Tata Consultancy Services.

REFERENCES

1. Deri, L., Soldato, A.D: Enforcing security in iot and home networks. In: Ferrari, E., Baldi, M., Baldoni, R. (eds.) Proceedings of the Second Italian Conference on Cyber Security, Milan, Italy, February 6th - to - 9th, 2018. vol. 2058 of CEUR Workshop Proceedings. (CEUR-WS.org) (2018)

2. Zarpelão, B.B, et al.: A survey of intrusion detection in internet of things. J. Netw. Comput. Appl. 84, 25-37. http://www.sciencedirect.com/science/article/pii/S1084804517300062 (2017)
3. Sivaraman, V., et al.: Smart iot devices in the home: security and privacy implications. IEEE Technol. Soc. Mag. 37(2), 71–79 (2018)
4. Kumar, A., Lim, T.J: EDIMA: early detection of iot malware network activity using machine learning techniques. CoRR, abs/1906.09715. http://arxiv.org/abs/1906.09715 (2019)
5. Hubballi, N., Swarnkar, M.: bitwise: network traffic classification through encoded bit level signatures. IEEE/ACM Trans. Netw. 26(5), 2334–2346 (2018)
6. Sivaranthan, A., et al.: Classifying iot devices in smart environments using network traffic characteristics. IEEE Trans. Mobile Comput. 18(8), 1745–1759 (2019)
7. Meidan, Y., et al.: Profiliot: a machine learning approach for iot device identification based on network traffic analysis. In: Proceedings of the Symposium on Applied Computing, SAC ’17. (New York, NY, USA: ACM), pp. 506–509 (2017). http://doi.acm.org/10.1145/3019612.3019878
8. Desai, B.A., et al.: A feature-ranking framework for iot device classification. In: 2019 11th International Conference on Communication Systems Networks (COMSNETS), pp. 64–71 (2019)
9. Shahid, M.R., et al: Iot devices recognition through network traffic analysis. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 5187–5192 (2018)
10. Santos, M.R.P., et al: An efficient approach for device identification and traffic classification in iot ecosystems. In: 2018 IEEE Symposium on Computers And Communications (ISCC), pp. 00304–00309 (2018)
11. Hafeez, I., et al: Real-time iot device activity detection in edge networks. In: Au, M.H., et al. (eds.) Network and system security, pp. 221–236. Springer International Publishing, Cham (2018)
12. Egea, S., et al: Intelligent iot traffic classification using novel search strategy for fast-based-correlation feature selection in industrial environments. IEEE Internet Things J. 5(3), 1616–1624 (2018)
13. Lopez-Martin, M., et al: Network traffic classifier with convolutional and recurrent neural networks for internet of things. IEEE Access, 5, 18042–18050 (2017)
14. Yao, H., et al: Capsule network assisted iot traffic classification mechanism for smart cities. IEEE Internet Things J. 6(5), 1 (2019)
15. Marchal, S., et al: Audi: towards autonomous iot device-type identification using periodic communication. IEEE J. Sel. Area Commun. 37(6), 1402–1412 (2019)
16. Aksoy, A., Gunes, M.H.: Automated iot device identification using network traffic. In: ICC 2019 - 2019 IEEE International Conference on Communications (ICC) (2019)
17. Salman, O., et al: A machine learning based framework for iot device identification and abnormal traffic detection. Transactions on Emerging Telecommunication Technologies, pp. 1–15 (2019)
18. Bezawada, B., et al.: Behavioural fingerprinting of iot devices. In: ASHES ’18: Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security. (Association for Computing Machinery), pp. 41–50 (2018)
19. Ortiz, J., Crawford, C., Le, F. DeviceMien: Network device behaviour modelling for identifying unknown iot devices. Proceedings of the International Conference on Internet of Things Design and Implementation. IoTDI ’19. (New York, NY, USA: ACM), pp. 106–117. (2019). http://doi.acm.org/10.1145/3302805.3310073
20. Bai, L., et al: Automatic device classification from network traffic streams of internet of things. In: 2018 IEEE 43rd Conference on Local Computer Networks (LCN), pp. 1–9 (2018)
21. Noguchi, H., Kataoka, M., Yamato, Y.: Device identification based on communication analysis for the internet of things. IEEE Access, 7, 52903–52912 (2019)
22. Guo, H., Heidemann, J.: Ip-based iot device detection. In: Proceedings of the 2018 Workshop on IoT Security and Privacy. (New York, NY, USA: ACM), pp. 36–42 (2018). http://doi.acm.org/10.1145/3229565.3229572
23. Hsu, A.S.: Automatic Internet of Things device category identification using traffic rates’(Master Thesis). Virginia Polytechnic Institute and State University (2019)
24. https://iotanalytics.unsw.edu.au/
25. https://www.tcpdump.org/
26. Khandait, P., Hubballi, N., Mazumdar, B., Efficient keyword matching for deep packet inspection based network traffic classification. In: 2020 International Conference on COMmunication Systems NETworkS (COMSNETS). IEEE, pp. 567–570 (2020)
27. Wang, X., et al: Hyperscan: a fast multi-pattern regex matcher for modern cpus. In: 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). (Boston, MA: USENIX Association), pp. 631–648. https://www.usenix.org/conference/nsdi19/presentation/wang-xiang (2019)
28. Choi, B., et al: DFC: Accelerating string pattern matching for network applications. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16). (Santa Clara, CA: USENIX Association), pp. 551–565. https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/choi (2016)