An IEEE Societies and Technical Councils Publication
IEEE Data Descriptions

Descriptor: *UNSW IoT Traffic Data with Packets, Flows, and Protocols* (UNSW-IoTraffic)

Savindu Wannigama[1], Arunan Sivanathan[2], and Hassan Habibi Gharakheili[2]

[1]Department of Computer Engineering, University of Peradeniya, Sri Lanka
[2]School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia

Corresponding author: Hassan Habibi Gharakheili (email: h.habibi@unsw.edu.au).

ABSTRACT This work describes UNSW IoT traffic data (UNSW-IoTraffic), a dataset comprising (a) raw network packet traces with full headers and payload, (b) flow-level metadata summarizing fine-grained bidirectional activity behaviors, and (c) protocol parameters describing network protocol characteristics. It also provides protocol data models for six dominant protocols (TLS, HTTP, DNS, DHCP, SSDP, and NTP). In addition, the dataset includes scripts for statistical summarization, visualization using state diagrams, and device fingerprinting using machine learning. The dataset contains 95.5 million packets of IoT communications captured over 203 days, organized into 27 per-device packet capture (PCAP) files. Derived flow data, categorized based on the 5-tuple attributes (source IP address, destination IP address, transport-layer protocol number, source port number, destination port number), are provided as 27 per-device CSV files. Finally, protocol-specific parameters for 70% flows are extracted and written into 450 CSV files across 27 device types, covering 25 protocols, each with request and response data. The three-level structure of our dataset, which encompasses packets, flows, and protocols, caters to a diverse range of users, from students learning data networking concepts to experienced researchers and industry professionals. It enables the behavioral analysis of consumer IoT devices, the detection of temporal anomalies, and the validation of protocols.

IEEE SOCIETY/COUNCIL Computer Society (CS)

DATA DOI/PID 10.5061/dryad.w0vt4b94b

DATA TYPE/LOCATION PCAP, CSV, and JSON files; Dryad

INDEX TERMS IoT, Network Behavior, Packets, Flows, Protocols

BACKGROUND

In recent years, publicly available datasets of IoT network traffic traces have contributed to research in areas like device classification, malware and attack detection, and cyber risk assessment. Although these datasets offer valuable information, there remains a gap in "comprehensive" datasets that cover a wide range of heterogeneous devices over extended periods, as well as those that provide "multiresolution" data (*e.g.,* packets, flows, protocols) and are "structured" in a modular manner (*e.g.,* per-device, per-protocol) with integrated traffic processing tools. The availability of such datasets and accompanying software tools is essential for deepening our understanding of IoT behaviors and advancing data-driven models in network asset management, traffic classification, and anomaly detection.

Strengths and Gaps in Existing IoT Traffic Datasets

Several datasets, such as YourThings [1], SHiOT [2], VAR-IoT [17], and IPFIX-IoT [3] focus on benign IoT traffic captured under realistic conditions. YourThings provides publicly downloadable and relatively detailed data, but is limited to a 10-day capture period that primarily includes idle and interaction phases, omitting the initial device setup phase. This omission restricts comprehensive behavioral analysis. IPFIX-IoT provides fine-grained flow records collected over three months, covering all three operational phases. However, the YourThings and IPFIX-IoT datasets lack representation of device-specific or protocol-specific network data. SHiOT offers statistical summaries and flow-level characterizations but is not publicly available.

Other datasets, including IoT-23 [4], CIC-IoT-2023 [5], and ACI-IoT-2023 [6], contain labeled malicious traffic,

TABLE 1: Summary of public IoT traffic datasets. Fields marked as "*not-reported*" indicate that the original sources did not provide exact values. In some cells, values are approximated or inferred based on available descriptions.

Dataset	# IoT Devices	Capture Duration	Flow Data	# Packets
UNSW-IoTraffic (ours)	27	203 days	Yes	95.5M
YourThings [1]	45	13 days	No	451.5M
SHiOT [2]	36	144 days	No	555.5M
IPFIX [3]	26	3 months	Yes	*not-reported*
Benign IoT-23 [4]	3	Few hours	Yes	428K
CIC-IoT-2023 [5]	105	*not-reported*	Yes	*not-reported*
ACI-IoT-2023 [6]	30	5 days	*not-reported*	8.0M
Mon(IoT)r [7]	81	1 months	No	40.8M
LSIF [8]	22	20 days	*not-reported*	32.4M
IoTFinder [9]	53	2 months	No	2.0M
HomeMole-Ind [10]	10	2 days	No	7.2M
IoT Sentinel [11]	31	2 mins	No	193K
BoT-IoT [12]	5	*not-reported*	Yes	*not-reported*
N-BaIoT [13]	9	*not-reported*	Yes	*not-reported*
ToN_IoT [14]	10	*not-reported*	Yes	*not-reported*
Edge-IIoTset [15]	>10	51 days	Yes	*not-reported*
D-IoT filtered [16]	24	120 days	No	110M

including IoT-specific malware, DDoS attacks, and port scanning activities. Although these datasets are valuable for security analytics, they exhibit notable limitations. These include limited device diversity (*e.g.,* IoT-23 contains traffic from only a very small number of devices), short capture durations (typically ranging from a few hours to a few days), or unbalanced representations of benign versus malicious behaviors. For example, IoT-23 includes more than 320M malicious flows and fewer than 2,000 benign flows. The most frequent attack type within the malicious class accounts for more than 200M flows, whereas the least frequent occurs only twice. Such synthetic imbalances can lead to overly simplistic detection scenarios or pose challenges to developing and evaluating models under realistic conditions.

Previous research utilizing datasets such as Mon(IoT)r [7], LSIF [8], and IoTFinder [9] has primarily focused on fine-grained annotations or lightweight fingerprints tailored for device classification and identification tasks. Although these datasets are valuable for such targeted applications, they often lack access to complete raw packet captures. For example, Mon(IoT)r provides only packet headers, LSIF offers filtered PCAPs per day per device, and IoTFinder provides only DNS-response-filtered PCAPs. This limited visibility at the protocol level constrains broader analytical opportunities and restricts deeper insights into the full spectrum of device communication patterns and protocol usage.

Public availability is another crucial factor that affects the utility and impact of IoT traffic datasets. Several high-quality datasets remain partially or entirely inaccessible, limiting reproducibility and broader community adoption. For example, SHiOT's detailed flow statistics and summaries are not publicly hosted, restricting opportunities for reuse and validation. Similarly, VARIoT offers only coarse-grained flow data without raw PCAPs; although parts of the dataset are accessible, its availability is restricted by amalgamating

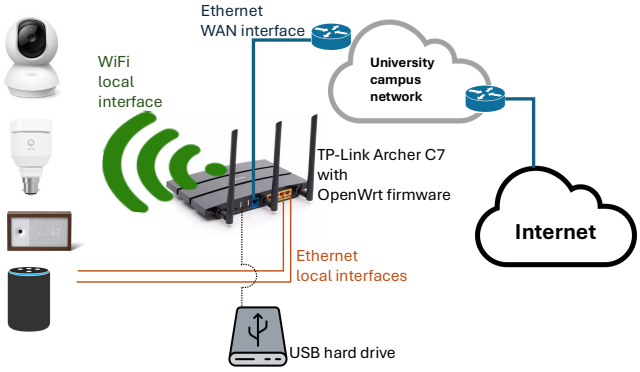

FIGURE 1: Our lab setup for collecting IoT raw traffic.

multiple subdatasets, each with different access conditions. In contrast, Deakin IoT Traffic (D-IoT) [16] provides publicly available aggregated daily PCAP files covering over 20 devices and, like our dataset, captures network activity in all three operational phases: setup, idle, and interaction. However, it lacks segmentation per device, flow-level data, and protocol-level annotations and parameters. Table 1 summarizes public IoT traffic datasets, highlighting their coverage of devices, capture duration, and data types.

Motivation for a New Dataset

The limitations highlighted above underscore the need for a unified and publicly accessible dataset that provides:

1) Long-term captures of benign traffic from a diverse set of IoT devices, spanning all operational phases (setup, idle, and active interactions),
2) Fine-grained visibility at flow-level and protocol-level,
3) Protocol-specific parameter extraction for deeper semantic analysis,
4) Reusable scripts and analytical models to support reproducibility and facilitate benchmarking,
5) Enabling various research applications, including security analytics, traffic classification, and network behavior profiling.

Our Contribution: UNSW IoT Traffic Data (UNSW-IoTraffic)

To address these gaps, we introduce the UNSW-IoTraffic Dataset, a publicly available, unrestricted, and reusable resource designed to support various IoT research needs. This dataset offers several key advantages:

- **Device Diversity**: Includes traffic from 27 unique consumer IoT device types, allowing detailed device-specific analysis.
- **Full Operational Coverage**: Records device behavior across setup, idle, and interaction phases over 203 days, enabling comprehensive behavioral modeling.
- **Extensive Protocol Coverage**: Features detailed flow-level data for 25 common IoT protocols, supporting multiprotocol analysis.
- **Multiresolution Data**: Provides raw PCAPs, detailed flow-level data (based on 5-tuple groupings), and

protocol-specific parameters for both request and response traffic directions, enhancing analytical depth.
- **Reusable Analytical Resources**: Contains reusable scripts for statistical analysis, state diagram-based visualization, and machine learning-based device fingerprinting.
- **Broad Applicability**: Designed to support security researchers (*e.g.,* risk assessment), network analysts (*e.g.,* behavior profiling), machine learning practitioners (*e.g.,* traffic classification, anomaly detection), and educators (*e.g.,* computer networking lab exercises and teaching demonstrations).

UNSW-IoTraffic represents a significant advancement, incorporating detailed protocol-specific parameters not present in earlier datasets. This opens up new opportunities for research and educational exploration in IoT traffic analysis.

Previous Publications Made Using the Dataset

The UNSW-IoTraffic dataset is an expanded, extended, and enriched version of the packet traces analyzed earlier by our seminal work on IoT traffic classification [18]–[20]. Those earlier studies focused mainly on 20 daily raw PCAP files, representing a small subset of the current UNSW-IoTraffic dataset. Our recent work [21] analyzed the UNSW-IoTraffic dataset by focusing on six common protocols: TLS, HTTP, DNS, NTP, DHCP, and SSDP. In that study, we developed protocol-specific models capable of automatically detecting these protocols in network traffic, thereby enabling analysis of IoT device behaviors, as well as identification of protocol vulnerabilities and misconfigurations.

COLLECTION METHODS AND DESIGN

IoT Network Setup

Our lab setup included a standard gateway, model TP-Link Archer C7 v2, running an instance of OpenWrt firmware Chaos Calmer (15.05.1, r48532). As shown in Fig. 1, the gateway connected consumer IoT devices on one side through WiFi or Ethernet local interfaces (LAN: local area network) and, on the other side, provided Internet access through an external Ethernet connection (WAN: wide area network) via our university's campus network. We captured raw packet traces on the LAN side of the gateway using the `tcpdump` tool (4.5.1-4) running on OpenWrt. All network traffic was captured centrally on the OpenWrt gateway, with packet timestamps based on its clock. This eliminated the need for cross-device synchronization and ensured consistent timing for temporal analysis. Recording packets at the local interface before applying network address translation (NAT) enables one-to-one mapping between each connected IoT device and its corresponding MAC or IP address. This mapping allows us to isolate the traffic of individual devices from a mix of packets belonging to multiple devices.

The setup connected 27 IoT devices to the LAN or WLAN interfaces of the TP-Link access point, along with a few non-IoT devices. We chose specific IoT makes and models based on their global consumer popularity and market availability at the time of data collection (2016), to reflect representative (though not exhaustive) device diversity. The main difference between IoT and non-IoT devices in this context lies in their intended functionality: IoT devices are typically designed for a specific, often single purpose (*e.g.,* smart lightbulb, smoke detector), while non-IoT devices are general purpose and capable of performing a wide range of tasks depending on user needs and interactions (*e.g.,* mobile phone, personal computer).

The 27 connected IoT devices fall into the categories of cameras, switches, triggers, hubs, air quality sensors, electronics, healthcare devices, and light bulbs. All devices were configured according to the settings recommended by the respective device manufacturers.

Data collection encompasses periods during which users were present or absent within the lab environment. The recorded traffic traces reflect a range of user interactions (though without ground-truth annotations) with different devices. Examples include a Belkin Wemo sensor triggering upon motion detection, an Amazon Echo responding to voice commands, and a LiFX lightbulb adjusting color or intensity in response to requests via its companion app. More complex interactions also occur, such as a Netatmo Welcome camera detecting an occupant and signaling the LiFX lightbulb to turn on with a preconfigured color. In addition to these user-driven events, the dataset contains idle periods characterized by periodic, autonomously generated background traffic, such as DNS and NTP communications, that are unaffected by direct human interaction.

Collected Data

We used a script to automate traffic data collection and storage, starting capture at midnight local time each day using a `Cron` job on OpenWrt, following a scheduled reboot to ensure a clean state and reliable data capture. Additionally, we implemented a monitoring script on OpenWrt to check the data collection process every 5 seconds. If the logging process was found to be inactive during these checks, the script would immediately restart it, ensuring that any possible data loss was limited to a maximum of 5 seconds. That said, data loss could still occur due to power outage or failures in scheduled daily reboots (the following subsection discusses capture discontinuity). We installed additional OpenWrt packages on the gateway; `block-mount` package for mounting external USB storage on the gateway, `kmod-usb-core` and `kmod-usb-storage` (3.18.23-1) for storing the traffic trace data on the USB storage. The data traces collected were stored as PCAP files on a 1TB external USB hard drive attached to the gateway, allowing continuous traffic tracking for several months. Each daily PCAP file was filtered by the MAC address of individual IoT devices connected to the network, generating device-specific PCAP files that contained only packets sent to and from a given MAC address.

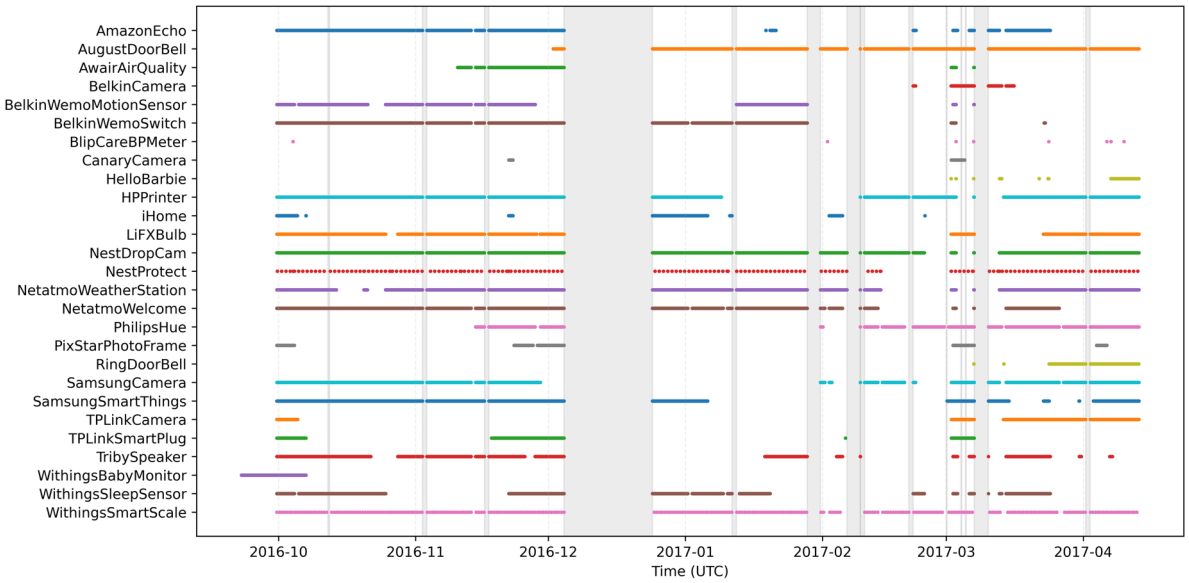

FIGURE 2: Temporal activity for each device is shown, with concurrent idle periods of $\geq$10 min highlighted by gray bands (all devices are inactive). Times are given in UTC; our lab local time corresponds to AEDT (UTC+11) until 2017-04-02 and AEST (UTC+10) thereafter.

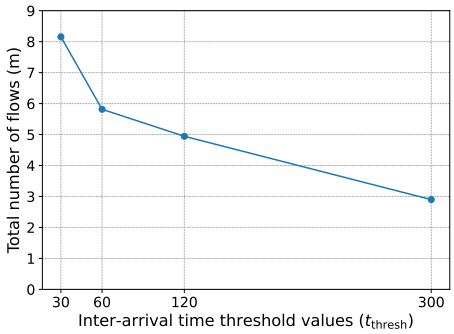

FIGURE 3: Total flow count (in millions) aggregated across all devices as a function of the inter-arrival time threshold (t_{thresh}).

Capture Continuity and Packet Loss Analysis

We evaluated capture continuity over the 203-day period by calculating inter-packet gaps for each device in our dataset and identifying *concurrent* idle windows, periods of at least 10 minutes during which all devices on the network were inactive. This threshold was selected to reliably capture genuine packet loss or disruption events. Shorter windows could introduce false positives, as it is not unusual for IoT devices to remain inactive and generate no traffic for several minutes during idle periods. Fig. 2 shows the temporal activity of each device, with gray bands indicating concurrent idle periods. In total, we identified 14 such intervals with durations varying from $\approx$4 h to $\approx$20 days, totaling $\approx$869 h ($\approx$36.2 days) of concurrent inactivity on all devices.

We experimented with idle periods shorter than 10 minutes, but this specific threshold yielded true disruptions with reasonable explanations. Manual inspection of these concurrent idle periods revealed that the start and/or end of almost all gray bands coincided with 00:00 local time (13:00 or 14:00 UTC), suggesting that they were caused by failures in reboots scheduled at midnight, which were then delayed to the following day. Note that the times in Fig. 2 are given in UTC, while our lab's local time corresponds to AEDT (UTC+11) until 2017-04-02 and AEST (UTC+10) thereafter. Finally, the longest gap, from 04-Dec-2016 to 24-Dec-2016, corresponds to our lab's annual maintenance shutdown.

Data Processing

1) Raw Network Packet Traces

After collecting the data, we filter the traffic by the MAC address of the target devices to produce device-specific PCAP files. Each device PCAP contains all packet traces associated with a single device throughout the data collection period. These device-specific PCAPs form the first abstraction level in our dataset: **raw network packet traces including full headers and payloads**. Table 2 summarizes the data collection periods for each device and the total number of packets from our lab testbed.

2) Flow-Level Metadata

We used an automation script written in Java to process the raw PCAP files and identify the communication flows generated by the devices. A communication *flow* is defined as a set of network packets grouped based on 5-tuple attributes (*i.e.,* source IP address, destination IP address, protocol number, source port number and destination port number). Note that a full communication session between a source (*e.g.,* an IoT device) and a destination (*e.g.,* a cloud server) typically consists of two unidirectional flows,

TABLE 2: Summary of device-specific PCAP traces: device identity, time period covered, number of packets, and number of flows.

Device	Device MAC	First Packet Date	Last Packet Date	# days	# Packets	# Flows	PCAP size
AmazonEcho	44:65:0d:56:cc:d3	2016-09-30	2017-03-24	175	4,666,750	245,784	990 MB
AugustDoorBell	e0:76:d0:3f:00:ae	2016-12-02	2017-04-13	132	20,040,732	1,132,276	5.81 GB
AwairAirQuality	70:88:6b:10:0f:c6	2016-11-10	2017-03-07	117	906,079	4,406	192 MB
BelkinCamera	b4:75:0e:ec:e5:a9	2017-02-21	2017-03-16	23	663,382	65,169	150 MB
BelkinWemoMotionSensor	ec:1a:59:83:28:11	2016-09-30	2017-03-07	158	5,921,154	680,235	1.61 GB
BelkinWemoSwitch	ec:1a:59:79:f4:89	2016-09-30	2017-03-24	175	4,273,643	490,562	1.02 GB
BlipCareBPMeter	74:6a:89:00:2e:25	2016-10-04	2017-04-10	188	620	54	106 KB
CanaryCamera	7c:70:bc:5d:5e:dc	2016-11-22	2017-03-05	103	6,131,096	6,598	5.64 GB
HelloBarbie	28:c2:dd:ff:a5:2d	2017-03-02	2017-04-13	42	5,123	150	1.94 MB
HPPrinter	70:5a:0f:e4:9b:c0	2016-09-30	2017-04-13	195	1,629,171	594,646	277 MB
iHome	74:c6:3b:29:d7:1d	2016-09-30	2017-02-24	147	1,635,400	60,352	409 MB
LiFXBulb	d0:73:d5:01:83:08	2016-09-30	2017-04-13	195	1,368,191	102,032	166 MB
NetatmoWeatherStation	70:ee:50:03:b8:ac	2016-09-30	2017-04-13	195	1,542,257	78,879	271 MB
NetatmoWelcome	70:ee:50:18:34:43	2016-09-30	2017-03-26	177	4,967,010	315,974	2.05 GB
NestDropCam	30:8c:fb:2f:e4:b2	2016-09-30	2017-04-13	195	23,497,598	2,108	3.68 GB
NestProtect	18:b4:30:25:be:e4	2016-09-30	2017-04-13	195	702,949	256,956	693 MB
PhilipsHue	00:17:88:2b:9a:25	2016-11-14	2017-04-13	150	3,975,530	302,097	825 MB
PixStarPhotoFrame	e0:76:d0:33:bb:85	2016-09-30	2017-04-06	188	83,832	15,358	16.2 MB
RingDoorBell	88:4a:ea:31:66:9d	2017-03-07	2017-04-13	37	39,107	1,105	23.1 MB
SamsungCamera	00:16:6c:ab:6b:88	2016-09-30	2017-04-13	195	5,880,402	379,921	2.00 GB
SamsungSmartThings	d0:52:a8:00:67:5e	2016-09-30	2017-04-13	195	3,324,934	42,912	355 MB
TPLinkCamera	f4:f2:6d:93:51:f1	2016-09-30	2017-04-13	195	435,198	48,007	49.8 MB
TPLinkSmartPlug	50:c7:bf:00:56:39	2016-09-30	2017-03-07	158	79,773	12,816	12.0 MB
TribySpeaker	18:b7:9e:02:20:44	2016-09-30	2017-04-07	189	685,844	51,650	224 MB
WithingsBabyMonitor	00:24:e4:11:18:a8	2016-09-22	2016-10-07	15	943,513	11,241	71.1 MB
WithingsSleepSensor	00:24:e4:20:28:c6	2016-09-30	2017-03-24	175	2,097,512	41,223	412 MB
WithingsSmartScale	00:24:e4:1b:6f:96	2016-09-30	2017-04-13	195	46,605	1,530	12.6 MB
			TOTAL	**203**	**95,543,405**	**4,944,041**	**26.9 GB**

TABLE 3: Protocol diversity and prevalence in UNSW-IoTraffic.

Protocol	# Devices	# Flows
DNS	27	596,341
DHCP	27	40,719
ICMP	26	476,420
TLS	24	760,434
HTTP	21	818,456
LLDP	21	24,840
SYSLOG	17	173
NTP	16	263,211
SSDP	13	264,128
IGMP	10	1,995
MDNS	9	4,758
ICMPv6	8	11,614
SNMP	7	28
RTMP	5	11
STUN	4	55,170
ISAKMP	3	240
XMPP	3	128
Classic-STUN	2	3,967
NBNS	2	1,529
SIP	2	17,068
SMB	2	2,529
RTSP	2	9
LLMNR	1	10
IPP	1	5
RTP	1	3
Others	24	1,600,255

one in each direction, like those found in a persistent TCP connection between a client and a server. Therefore, the flows in our dataset are bidirectional. However, we may observe partial flow records if a period of inactivity is detected in either direction. The packets are grouped into the same flow record as long as the inter-arrival time between consecutive packets does not exceed 120 seconds in either direction. The threshold of 120 seconds was selected based on an empirical analysis of the total number of flows exported from all 27 devices for $t_{\text{thresh}} \in \{30, 60, 120, 300\}$ seconds. As shown in Fig. 3, the flow count drops sharply between 30 s and 60 s, indicating that small thresholds cause excessive fragmentation of sessions. Beyond 60 s, the rate of decrease slows, with the reduction from 120 s to 300 s being comparatively modest. Selecting 120 s, therefore, captures the major reduction in flow count achieved by increasing the threshold, while avoiding the risk of over-aggregating distinct sessions that may occur at larger values such as 300 s. Table 4 shows how this choice balances the need to minimize unnecessary flow fragmentation with the need to preserve meaningful session boundaries.

Within each flow, the unidirectional flow that sends the first packet is designated as the initiator, while the other is considered the responder. Parameters associated with the initiator are labeled as source parameters (prefixed with "`src`"), and those for the responder as destination parameters (prefixed with "`dst`"). A complete list of flow parameters is provided in Table 7 and further discussed in Section B.

The automation script processes each device's PCAP file by grouping packets into bidirectional flows and applying protocol models, as described in [21], to identify the protocol(s) used within each flow. While a flow may contain packets from multiple protocols, it is typically dominated by a single protocol. Using protocol data models for a selected set of 55 protocols, the script labels each flow with the most representative protocol based on its packet contents. Table 3 summarizes the most frequently observed protocols in the UNSW-IoTraffic dataset across all devices.

To determine how the choice of the inter-arrival time threshold affects the detected mix of protocols, we computed the total variation distance (TVD) between the protocol distributions at successive threshold values $\in \{30, 60, 120, 300\}$ seconds. For each threshold t_i, let

$$P_{t_i}(p) = \frac{\text{number of flows per protocol } p}{\text{total number of flows}}.$$

Then for two successive thresholds t_i and t_j, the TVD is computed by:

$$\text{TVD}\big(P_{t_i}, P_{t_j}\big) = \frac{1}{2}\sum_{p}\big|P_{t_i}(p) - P_{t_j}(p)\big|,$$

which ranges from 0 (identical distributions) to 1 (no overlap). Table 4 summarizes the TVD between three pairs of successive thresholds. It can be seen that TVD first falls before it rises, experiencing its lowest value (0.071) at the transition from 60 to 120 s, indicating that the protocol mix has effectively stabilized by 120 s, while the jump to 300 s (TVD = 0.354) reflects excessive merging of distinct flows

TABLE 4: Total variation distance (TVD) between protocol distributions at successive inter-arrival time thresholds.

Threshold transition	TVD
$\mathbf{30} \rightarrow \mathbf{60}$ s	0.114
$\mathbf{60} \rightarrow \mathbf{120}$ s	0.071
$\mathbf{120} \rightarrow \mathbf{300}$ s	0.354

that can obscure protocol detection. This empirical convexity in TVD further supports our choice of 120 seconds, balancing over-splitting against over-merging flows.

The collected data are stored as device-specific CSV files, where each row corresponds to a single bidirectional flow associated with that device. These CSV files represent the second level of abstraction in our dataset: **flow-level metadata**, which captures the statistical characteristics of IoT device behavior on the network. Algorithm 1 was used to transform raw packet traces to flow-level data of 5-tuple records. Our algorithm applies the same logic of flow extraction to both TCP and UDP traffic and, as such, does not examine TCP sequence numbers. Therefore, it does not attempt to reorder packets or detect duplicates within TCP flows; instead, it remains agnostic to such variations. This design choice simplifies packet processing and ensures consistent handling across transport-layer protocols.

3) Protocol Parameters

We extended the automation script initially used to extract flow-level metadata from raw packet traces, also to extract protocol-specific parameters. For each flow associated with a certain protocol (*e.g.,* TLS), the script extracts relevant parameter values (*e.g.,* the server-hello-cipher-suite for a TLS flow) and stores them in a separate set of CSV files. Since protocols define different parameters for request and response packets, these parameters are extracted separately for each direction. For a bidirectional flow, we consider the unidirectional flow originating from the initiator as the request flow, and the flow from the responder as the response flow. This distinction is used to extract and record protocol parameters for requests and responses systematically.

Similarly, as we use protocol data models to identify the protocol associated with a flow, we also employ these models to extract protocol-specific parameters from the flow. Each protocol model [21] defines a list of request and response parameters along with the corresponding matching patterns used to locate these parameters within the flow payload. During parameter extraction, if multiple matches are found for a given pattern within the payload, we extract the value from the first occurrence only.

Similarly to flow-level data, protocol parameters are presented in CSV files. Each CSV is specific to a device (D), protocol (P) and communication direction (R) (*i.e.,* request or response). In this structure, each row in a protocol parameter CSV represents a flow from device D that uses protocol P in the direction R. These CSV files of protocol parameters constitute the third level of abstraction in our dataset:

Algorithm 1 Extracting flows from a device PCAP file.

Input: *pcapPath* (path to a PCAP file)
Load the pcap file from *pcapPath* as *pcapHandler*

Initialize maps:
  Map<FlowTuple, List<Byte[]>> ipPacketBuffer
  Map<FlowTuple, List<FPDPacket>> ipFPDPackets
while *packet = pcapHandler.getNextPacket()* **do**
  if *packet* is NULL **then**
    continue
  end if
  Extract *header* and *payload* from *packet*
  if *payload* is empty **then**
    continue
  end if
  if *header.getType()* is not (IPv4, IPv6, TCP, UDP) **then**
    continue
  end if
  Extract flow the attributes *srcIP, dstIP, ipProto, srcPort, dstPort* from *header*
  Construct *flowTuple = (srcIP, dstIP, ipProto, srcPort, dstPort)*
  Construct *fpdPacket* = store packet information
  if *flowTuple* does NOT exist in *ipPacketBuffer ipFPDPackets* **then**
    Initialize new entry for *flowTuple* in *ipPacketBuffer* and *ipFPDPackets*
  end if
  Append *payload* to corresponding flow
end while

Initialize maps:
  Map<FlowTuple, FPDDirectionPayload> ipFPDDirectionPayload

for each *flowTuple* in *ipPacketBuffer.keys()* **do**
  Initialize: *fpdPayload* for *flowTuple* from *ipPacketBuffer* and *ipFPDPackets*
  if *flowTuple.getReversed()* exists in *ipFPDDirectionPayload* **then**
    Set *fpdPayload* as reverse direction payload for the existing *flowTuple*
  else
    Add new *flowTuple* to *ipFPDDirectionPayload*
    Set *fpdPayload* as forward direction payload for *flowTuple*
  end if
end for
Return *ipFPDDirectionPayload*

TABLE 5: Performance summary of three classifier models.

Metric	Precision	Recall	F1-score
RF			
Accuracy	0.68		
Macro avg	0.70	0.77	0.68
KNN			
Accuracy	0.70		
Macro avg	0.81	0.71	0.73
LR			
Accuracy	0.36		
Macro avg	0.35	0.41	0.27

protocol parameters, which capture detailed protocol-level characteristics of IoT devices.

VALIDATION AND QUALITY

This section aims to validate the UNSW-IoTraffic dataset through comprehensive experimental analyses, highlighting its reliability, completeness, and suitability for various applications.

Machine Learning-Based Device Traffic Classification

A popular use case in network management is automatically mapping of network traffic to the corresponding devices. To this end, we study how an input flow can be classified into its corresponding IoT type using three algorithms: (1) Random Forest (RF), (2) k-Nearest Neighbors (KNN), and (3) Multinomial Logistic Regression (LR). We trained these classifiers and evaluated their predictions in an identical pipeline (*i.e.,* same features and train/test sets). This consistency allows us to attribute differences in inference outcomes to their structure: nonlinear tree ensembles, instance-based locality, and linear softmax, respectively.

We aggregate 4.9M flow records from 27 IoT device classes and assign each row a device label (obtained from its source filename). All experiments use a stratified 70/30 train–test split to preserve class proportions. To avoid overfitting on trivial identifiers (*e.g.,* device address) and to exclude less-informative or overlapping attributes (*e.g.,* timestamp, transport-layer source and destination port numbers), we omit time stamps, MAC/IP addresses, EtherType, IP protocol number, transport ports, and flow sequence numbers. The resulting feature set comprises 67 attributes: 45 categorical and 22 numeric. Categorical features are converted to indicator (one-of-K) vectors, and numeric features are standardized to have a scale comparable in dimensions. A single preprocessor is shared between models.

For RF, we tune the number of trees, the number of features considered at each split, and the minimum samples per leaf, selecting 100 trees, $\log_2$ features per split, and a minimum leaf size of 1. For KNN, we search $k \in \{3, 5, 7\}$ and select $k = 5$. For LR, we vary the *L2 regularization* strength $C \in \{0.1, 1, 10\}$ and select a moderate L2 penalty of $C = 1$. Across the three models, the hyperparameters are selected by grid search with 5-fold cross-validation in the training split, after which the best configuration is re-fit on the full training data.

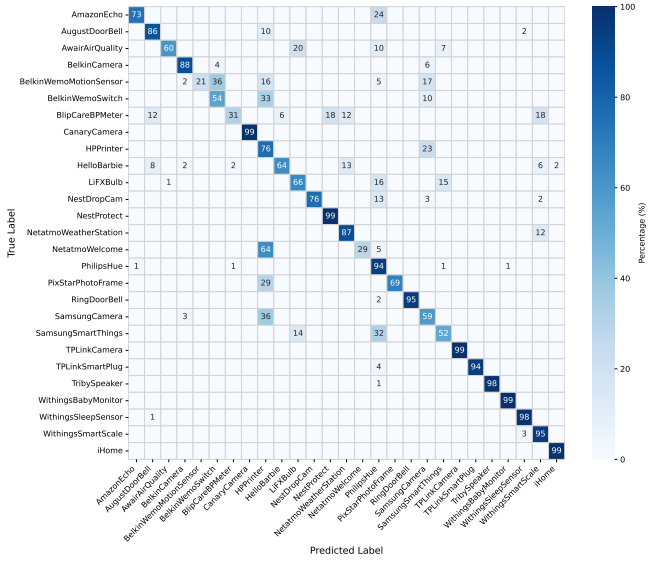

FIGURE 4: Normalized (row-normalized) confusion matrix for the Random Forest classifier, showing per-device recall and misclassification rates across 27 IoT device classes. All numbers are in percentages.

Results and Analysis

We evaluated the three models on the same held-out test set (*i.e.,* 30% of the flow records) using four metrics: overall accuracy together with macro-averaged precision, recall, and F1-score. Table 5 summarizes the performance of the three classifiers. It can be seen that the LR model struggles (overall accuracy of 0.36) due to its reliance on a linear hyperplane for class separation, where the relationships between traffic features and IoT classes are fairly non-linear [18]–[20]. Although KNN has slightly higher accuracy, RF provides more interpretable results [3], making it a preferred deployment model. We therefore conduct a deeper analysis of the RF model.

To better examine the performance of the RF classifier, we show its row-normalized (recall per true class) confusion matrix in Fig. 4. The overall accuracy is 68%. Despite this moderate overall accuracy, the model performs well in many individual classes: it achieves a recall of at least 80% for 14 out of 27 classes, and eleven are greater than 90%. These include prominent camera and speaker devices such as AugustDoorBell (86%), CanaryCamera (99%), iHome (99%), TPLinkCamera (99%), WithingsBabyMonitor (99%), and RingDoorBell (95%), highlighting the model's high reliability for these IoT classes. It can be seen in Fig. 4 that the classifier is mainly confused with a few pairs: NetatmoWelcome→HPPrinter (64%), SamsungCamera→HPPrinter (36%), BelkinWemoSwitch→HPPrinter (33%), and BelkinWemoMotionSensor→BelkinWemoSwitch (36%), highlighting where similar traffic patterns contributed to misclassification.

To interpret the RF model, we extracted feature importance scores based on the mean decrease in impurity. Fig. 5

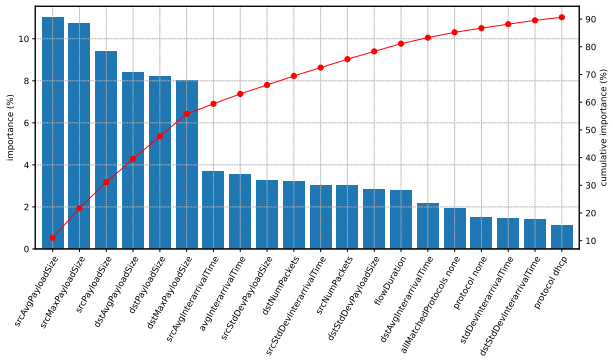

FIGURE 5: Top 20 features identified by the Random Forest classifier. The blue bars represent each feature's relative importance (as a percentage), while the red line (plotted on the right axis) indicates the cumulative importance.

shows the top 20 most important features. The blue bars represent the importance of individual features (as a percentage). At the same time, the red line (plotted on the right axis) indicates the cumulative importance, highlighting how quickly the most important features collectively contribute to the model's predictive power.

A couple of observations can be drawn from this analysis. First, statistical measures of payload size, particularly `srcAvgPayloadSize` and `srcMaxPayloadSize`, are key features in classifying IoT traffic flows. This suggests that upstream payload size distributions are among the most distinctive signatures exhibited by IoT devices in their flow-level metadata. In addition, count-based and temporal characteristics of flows, such as `srcNumPackets` and `avgInterarrivalTime`, also play a significant role. These features represent the second most influential group of device-specific class predictors.

Analyzing Service Usage Patterns of IoT Devices

To understand and manage the behavior of IoT devices, aggregate-level behavioral fingerprints can be constructed without necessarily relying on machine learning algorithms. This can be achieved through an aggregate analysis of the network services each device uses to communicate with its intended endpoints. An approach to conducting such an analysis involves processing flow-level data to create a state machine that represents the network services each IoT device interacts with over time.

For each IoT device, we sorted its network flows chronologically throughout the data collection period and assigned a service label to each flow based on its protocol information. If the flow's protocol could not be identified (*i.e.,* it did not match any of our 58 protocol data models during flow extraction from the raw PCAP files), we assigned a service label using the transport layer protocol (*e.g.,* TCP or UDP) combined with the port number (*e.g.,* `TCP/8883`). Flows using transport layer protocols other than TCP or UDP were labeled as `OTHER`. These service labels serve as the states in the state machine that represent the behavior of the device.

We then identified and counted the transitions from one state to another based on the chronologically sorted flow data. We computed two metrics from transitions across states: **(1)** Local Outgoing Transition Probability represents the fraction of transitions from a given source state that lead to a specific target state, indicating the likelihood of that transition given the original state. We use this metric to color the edges in our state diagram (brighter edge colors indicate higher transition probabilities). **(2)** Global Incoming Transition Count represents the total number of transitions *into* each state across all flows of the device. To visualize these counts, nodes are color-coded based on their position within three percentile-based groups, using the 33rd (T_1) and 66th (T_2) percentiles as thresholds: grey for "count $\leq T_1$" (lower third), blue for "$T_1 <$ count $\leq T_2$" (middle third), and red for "count $> T_2$" (upper third). Fig. 6 illustrates an example state diagram showing service usage patterns for the AwairAirQuality device. The AwairAirQuality's behavior is dominated by four key services (in red): DNS, DHCP, NTP and TLS, with DNS being the most dominant, receiving the largest number of incoming transitions. It can be seen that 50% of DNS lookups are followed by another DNS flow, while the remaining transitions lead to TLS or NTP. The directed graph representation shown in Fig. 6 can serve as a behavioral fingerprint for identifying and monitoring device activity at runtime. This approach is conceptually similar to the method proposed in [22], where the authors utilized a tree-based representation of Manufacturer Usage Description (MUD) data for device profiling.

Protocols for Fingerprinting and Risk Assessment

Protocol parameters offer insights into how specific protocols are configured and utilized by an IoT device connected to the network. Standards bodies advocate for visibility into protocol usage (*e.g.,* TLS [23]), as it can help reveal the presence of unauthorized or potentially malicious software on IoT devices. In our recent work [21], we analyzed a subset of the UNSW-IoTraffic dataset, focusing on protocol parameters extracted from the network traffic of ten representative IoT devices. Our analysis centered around six widely used protocols: TLS, HTTP, DNS, NTP, DHCP, and SSDP. Together, these six protocols account for 97% of all flows generated by the ten devices.

By analyzing device-specific PCAP files in conjunction with the standard protocols documentation, we constructed machine-readable data models to describe the signatures of these six protocols and extract their associated parameters. These data models are released as part of the protocols section in the UNSW-IoTraffic dataset. We applied protocol models to raw PCAP files and demonstrated high detection accuracy with zero false positives.

We extracted key parameters such as cipher suites, protocol versions, and authentication methods to characterize

TABLE 6: Device-specific protocol signatures and vulnerabilities for three representative IoT devices.

Device	# Client Cipher Suites	Server Cipher Suites	# Client Extensions	HTTP Methods	# NTP Vulns	HTTP Basic Auth?
RingDoorBell	10	{0xc027}	1	POST	2	Yes
TribySpeaker	66, 80	{0xc030}	2	GET	0	Yes
AmazonEcho	55, 91	{0x0035, 0x002f, 0xc030, 0xc02f, 0xc013}	8	GET, HEAD, PUT	1	No

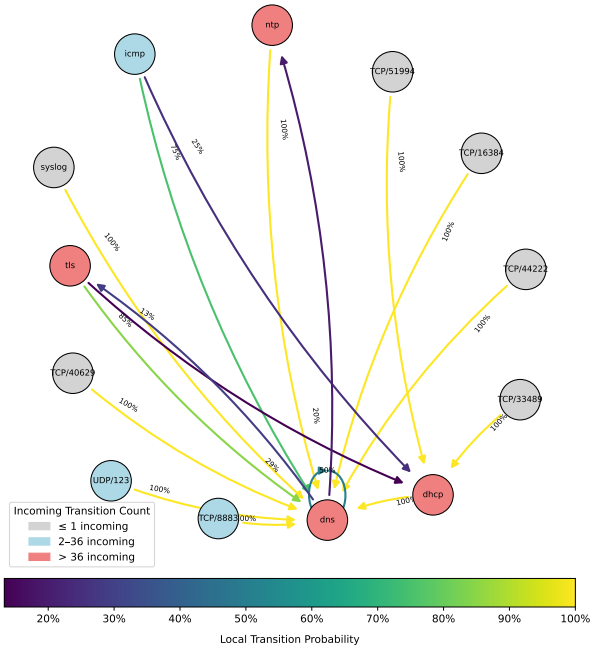

FIGURE 6: Approximate state diagram for AwairAirQuality. Nodes represent network services, while directed edges show transitions; edge color indicates local outgoing transition probability, and node color reflects the global incoming transition count.

the behavior of the IoT devices. The results revealed distinct behavioral fingerprints for each device, validating the high-fidelity, device-specific insights provided by our dataset. We also leveraged protocol models for risk assessment, using them to identify misconfigured security parameters, such as weak cipher suites in TLS flows and improper authentication mechanisms in HTTP flows.

Let us focus on three representative IoT devices from our dataset to demonstrate the utility of protocol parameters in device fingerprinting and security assessment. Table 6 presents some unique attributes with an emphasis on the TLS and HTTP protocols (columns 2-5), alongside the vulnerabilities identified by analyzing the parameters of the NTP and HTTP protocols (columns 6-7). Starting with unique signatures, the Ring doorbell consistently offers an ordered list of 10 cipher suites (in its client-hello messages) to all cloud servers with which it communicates. However, the Triby speaker and Amazon Echo use one of two unique lists specific to these devices at the beginning of their TLS connections with their intended servers. All lists offered across these three representative devices are unique. In addition, the corresponding TLS servers choose one cipher from the offered list: the server cipher suite of the Ring doorbell is unique, but this parameter for the Triby speaker overlaps with that of one of the servers with which the Amazon Echo communicates. We also note that these three devices are distinct by the number of client extensions used for the TLS protocol. Also, these three representative devices display relatively distinguishing patterns in their choice of HTTP methods when communicating with their cloud servers. Moving to vulnerabilities, we discovered the use of outdated NTP versions (*i.e.,* v3 or less) or insecure HTTP authentication methods (*i.e.,* Basic [24]), highlighting the role of protocol-level data in assessing device cyber risks. Together, these protocol parameters form a strong basis for building protocol-aware device profiles that can help identify devices and develop security guidelines to prevent attacks or compromises arising from protocol misuse.

RECORDS AND STORAGE

As mentioned in previous sections, our dataset is organized into three abstract levels to support analysis of IoT network behavior at different levels of granularity. The dataset consists of: (a) raw network packet captures (PCAPs), (b) network flows with statistical attributes, and (c) protocol models and parameters. Each representation is stored in a format that ensures ease of access and usability.

Fig. 7 illustrates the hierarchical structure of our dataset. The root folder "UNSW-IoTraffic" contains four main subfolders; **(1)** **`pcaps`** includes 27 device-specific raw packet capture (PCAP) files; **(2)** **`flows`** contains 27 CSV files, each providing flow-level metadata and statistics for a corresponding device; **(3)** **`protocols`** comprises two subfolders: **`models`** contains six JSON files, one per six standard protocols (TLS, HTTP, DNS, DHCP, SSDP, and NTP), used for protocol-specific parameter matching and extraction, and **`parameters`** structured into device-specific subfolders, each of which is further divided into **`request`** and **`response`** subfolders. These contain CSV files listing the extracted attributes for each of the six protocols; **(4)** **`scripts`** consists of three subfolders: **`random-forest-classifier`** and **`state-diagrams`** include source code for two of the validation tasks presented in the previous section, and **`summaries`** contains scripts used to extract flow-level metadata presented in Tables 2 and 3.

For ease of access, we publish five compressed `.zip` archives: one for the root folder and one for each of the four subfolders discussed above. This structure allows users to download any combination of folders they need.

TABLE 7: Descriptions of data fields across the flow-level and protocol-parameter CSV files in the UNSW-IoTraffic dataset.

Data Field	Description	Flow-Level Metadata CSV files	Protocol Parameters CSV files
`time`	Timestamp of first packet in the flow.	✓	✓
`srcMac`	MAC address of the flow initiator.	✓	✓
`dstMac`	MAC address of the flow responder.	✓	✓
`ethType`	Ethernet type field (*e.g.,* `0x0800 for IPv4).`	✓	✓
`srcIp`	IP address of the flow initiator.	✓	✓
`dstIp`	IP address of the flow responder.	✓	✓
`ipProto`	IP protocol number (*e.g.,* `6 for TCP, 17 for UDP).`	✓	✓
`srcPort`	Transport-layer port number of the flow initiator.	✓	✓
`dstPort`	Transport-layer port number of the flow responder.	✓	✓
`flowSeqNum`	Sequential occurrence of the flow within the device packet capture.	✓	✓
`srcNumPackets`	Number of packets transmitted by the flow initiator.	✓	
`dstNumPackets`	Number of packets transmitted by the flow responder.	✓	
`srcPayloadSize`	Total payload size (bytes) sent by the initiator.	✓	
`dstPayloadSize`	Total payload size (bytes) sent by the responder.	✓	
`srcAvgPayloadSize`	Average payload size (bytes) per initiator packet.	✓	
`dstAvgPayloadSize`	Average payload size (bytes) per responder packet.	✓	
`srcMaxPayloadSize`	Largest payload size (bytes) among initiator packets.	✓	
`dstMaxPayloadSize`	Largest payload size (bytes) among responder packets.	✓	
`srcStdDevPayloadSize`	Standard deviation of initiator packet payload sizes.	✓	
`dstStdDevPayloadSize`	Standard deviation of responder packet payload sizes.	✓	
`flowDuration`	Time between arrivals of first and last packets in the flow.	✓	
`srcAvgInterarrivalTime`	Average inter-arrival time between initiator packets.	✓	
`dstAvgInterarrivalTime`	Average inter-arrival time between responder packets.	✓	
`avgInterarrivalTime`	Average inter-arrival time between all packets in the flow.	✓	
`srcStdDevInterarrivalTime`	Standard deviation of inter-arrival times between initiator packets.	✓	
`dstStdDevInterarrivalTime`	Standard deviation of inter-arrival times between responder packets.	✓	
`stdDevInterarrivalTime`	Standard deviation of inter-arrival times between all packets in the flow.	✓	
`allMatchedProtocols`	A list of all potential protocols the flow may represent.	✓	
`protocol`	Protocol (*e.g.,* HTTP) identified for the content of application layer.	✓	
`<parameters>`	One or more protocol specific parameters.		✓

A. Raw Packet Captures

Our dataset provides the raw network packet traces as device-specific files in the standard `.pcap` format. Each file records all packets associated with a single IoT device over the entire data collection period. The files are named according to the pattern "`<DeviceName>_<DeviceMACaddr>.pcap`", allowing for unique identification of each device within the dataset. These capture files include both packet headers and payloads, and encompass a wide range of communication protocols, including the 25 protocols listed in Table 3.

These raw PCAPs serve as the foundational layer of our dataset, providing researchers with complete packet-level visibility. From these raw PCAPs, higher-level abstractions, including flow-level metadata and protocol parameters, are derived.

B. Flow-Level Metadata

For each IoT device, the dataset includes a corresponding CSV file that contains flow-level data extracted from the raw packet captures. These CSV files follow the naming convention "`<DeviceName>_<DeviceMACaddr>_flows.csv`", as shown in Fig. 7. As discussed earlier in this paper, each flow in our dataset is bidirectional, composed of two unidirectional flows exchanged between an initiator and a responder. Table 7 lists all the attributes included in each flow record.

The numerical attributes can be categorized into two groups: (a) amounts of data transferred, namely `NumPackets`, `PayloadSize`, `AvgPayloadSize`, `MaxPayloadSize`, and `StdDevPayloadSize`, which are calculated separately for each direction of a flow. This directional distinction enables a fine-grained analysis of how various IoT devices interact with their communication endpoints. For example, a camera device typically transmits significantly more packets and larger payloads than it receives. Note that aggregate statistics, such as the total number of packets or the average payload size for the entire bidirectional flow can be derived from the corresponding attributes of the two unidirectional flows; (b) timing metrics, specifically `AvgInterarrivalTime` and `StdDevInterarrivalTime`, are calculated using the time differences between consecutive packets within a flow. These timing metrics are calculated independently for each of the two unidirectional flows and the bidirectional flow. Note that the timing attributes of the bidirectional flow cannot be deduced from those of unidirectional flows. Additionally, each flow record includes two attributes for protocol identification: a list of all potential matching protocols (*i.e.,* `allMatchedProtocols`), and the single best-matched protocol determined (*i.e.,* `protocol`). Our protocol identification was based on protocol data models, which match the sequence of packets in each flow against a set of 55 protocol models. Using this approach, we identified 25 protocols across the dataset, as listed in Table 3.

This comprehensive set of flow attributes enables researchers to analyze the behavior of an IoT device itself (via unidirectional flows) and its interactions with other endpoints (via bidirectional flows). For example, the unidirectional flow attributes reveal the amount of data the device sends and how frequently it is sent, while bidirectional flow attributes capture the nature of the device's communication with other endpoints, including the rate at which packets are exchanged.

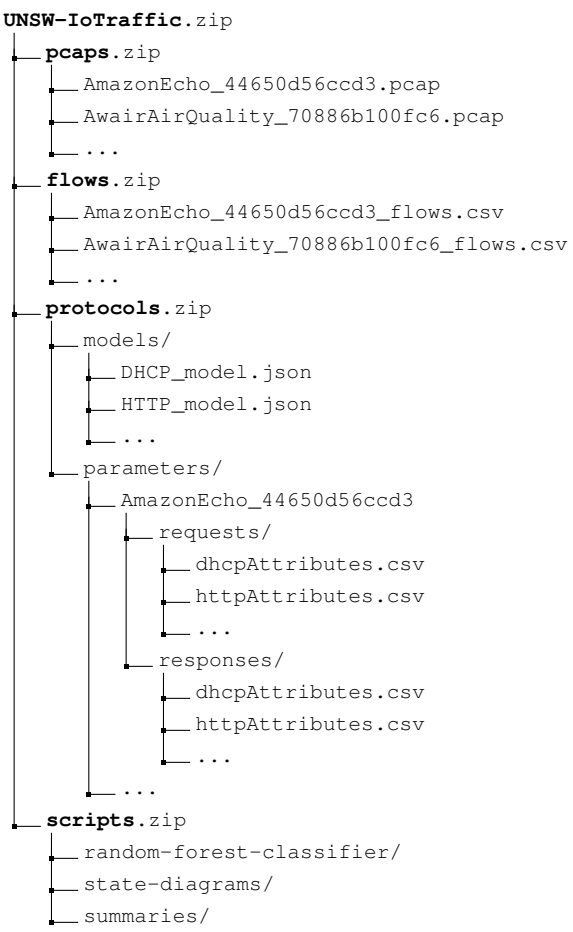

FIGURE 7: Structure of the UNSW-IoTraffic dataset.

C. Protocol Parameters

We extracted protocol parameters from the raw packets of flows listed in the CSV files of flow-level metadata, but only for those where a specific protocol was identified (*i.e.,* where the `protocol` attribute is not set to "none"). Approximately 70% of all flows in the dataset yielded protocol parameters. As previously described, our protocol data model defines the parameters to be extracted and the method for extracting them from flow payloads. The extracted protocol parameters are stored in a separate set of CSV files, each specific to a device, direction, and protocol. As a result, the request and response parameters are stored in separate CSV files. As shown in Fig. 7, the data for the protocol parameters is organized in a hierarchical folder structure of "`<DeviceName>_<DeviceMACaddr>/<requests|responses>`". Each of these folders contains protocol-specific CSV files, one for each protocol used by the corresponding device. These files follow the naming convention "`<protocol>Attributes.csv`".

The last column in Table 7 indicates the data fields found in a CSV file containing protocol parameters. Fields ranging from `time` to `flowSeqNum` are unique to each flow record and are identical to those in flow-level metadata. Therefore, the combination of fields (`srcMac`, `dstMac`, `ethType`, `srcIp`, `dstIp`, `ipProto`, `srcPort`, `dstPort`, `flowSeqNum`) can serve as a primary key to map flow-level data to the corresponding protocol parameters. The last field, denoted as `<parameters>`, contains protocol-specific values. The number and type of these parameters vary depending on the protocol. For example, for TLS, we extract three parameters from request flows: `client-hello-cipher-suites`, `client-hello-version`, `client-hello-extensions`, and three from response flows: `server-hello-cipher-suite`, `server-hello-version`, `server-hello-certificates`.

INSIGHTS AND NOTES

The UNSW-IoTraffic dataset is designed to support a wide range of educational and research applications. However, users should be aware of caveats and special considerations when working with the dataset.

Imbalanced and Sparse Device Activity: As shown in Table 2, devices in the dataset exhibit widely varying traffic volumes and activity levels. High-traffic devices, such as the Nest DropCam and August Doorbell, generate tens of millions of packets, while others, like the Blipcare BPMeter and Hello Barbie, produce minimal traffic over the same multi-month period. Additionally, specific devices, such as health sensors or baby monitors, may remain inactive for extended durations, emitting only intermittent signals. These natural variations reflect real-world usage patterns but may present challenges for downstream tasks such as traffic classification, behavioral modeling, or device fingerprinting. These imbalances and sparsity patterns should be considered when developing experiments, particularly those that are sensitive to class balance or temporal continuity.

No Privacy or Anonymization Constraints: We collected all data in a controlled testbed environment with minimal human interactions with the devices and their environment. Importantly, no personally identifiable information (PII) was involved in the data collection process. As such, no anonymization of MAC/IP addresses or payload content has been applied. This preserves full packet visibility and makes the dataset especially valuable for research that requires address-level behavior tracking or detailed payload analysis.

Beyond Flow-Based Analysis: While this paper demonstrated flow-level behavioral modeling and fingerprinting, the dataset can support a wide range of additional use cases. Our full packet captures enable applications such as time-series analysis, session reconstruction, and per-packet protocol inspection. The long capture duration and device diversity make our dataset suitable for longitudinal studies, including sequence modeling of service usage, temporal anomaly detection, concept drift, or label shift. Researchers interested in protocol-level dynamics, intrusion detection, or service fingerprinting may find the raw traces particularly valuable for deeper analysis.

SOURCE CODE AND SCRIPTS

The **`scripts`** folder in the UNSW-IoTraffic dataset contains a set of tools for analyzing, summarizing, and visualizing the flow-level metadata of the dataset. There are `readme` files inside each subfolder, with detailed explanations of the content and the usage of these scripts.

As briefly discussed in the "RECORDS AND STORAGE" section, these scripts are organized into three subfolders; **(1) `random-forest-classifier`**: Implements a machine learning pipeline to classify IoT devices based on their network flow statistics using a Random Forest classifier. This tool is particularly useful for students and educators in courses related to data analytics for computer networking or cybersecurity. The script `random_forest_classifier.py` trains and evaluates the model using the flow-level metadata, and `model_analysis.py` loads a trained model to regenerate feature importance visualizations and summaries of training results. This folder includes a `results` directory, which stores model metrics, visualizations, and summaries. Before running the scripts, the `data` directory must be populated with flow-level CSVs from the dataset. **(2) `state-diagrams`** contains a Jupyter notebook `state_diagram_notebook.ipynb` for generating and visualizing a state machine for the behavior of a given IoT device based on its service usage patterns. The network activity of the target device is captured as a directed graph showing transitions between different services (*e.g.,* `TCP/8443`, `UDP/553`, or a protocol like HTTP). The script processes per-device flow CSVs from the `data` folder. It labels each flow with a service name, sorts flows chronologically, and counts consecutive transitions (A) → (B) to compute local and global metrics. **(3) `summaries`** includes two Jupyter notebooks for generating high-level summaries from the flow-level metadata. The notebook `device_flow_counts.ipynb` counts the number of flows per device, while `protocol_summary.ipynb` computes the number of devices using each protocol and the total flow count per protocol. Outputs are written to the `results` folder and are presented in Tables 2 and 3.

ACKNOWLEDGEMENTS AND INTERESTS

A. Sivanathan designed and implemented the lab setup for traffic measurement and collected the foundational raw traffic data. S. Wannigama curated and analyzed the raw data, generated the flow-level and protocol-level datasets, conducted validation experiments, and wrote parts of the manuscript. A. Sivanathan and H. Habibi Gharakheili reviewed the data curation and analysis, and wrote parts of the manuscript. All authors reviewed the manuscript. The authors have declared no conflicts of interest.

REFERENCES

[1] O. Alrawi *et al.*, "YourThings: A Comprehensive Annotated Dataset of Network Traffic from Deployed Home-based IoT Devices," 2022.
[2] I. Cvitić *et al.*, "Ensemble Machine Learning Approach for Classification of IoT Devices in Smart Home," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 11, pp. 3179–3202, 2021.
[3] A. Pashamokhtari *et al.*, "Combining Stochastic and Deterministic Modeling of IPFIX Records to Infer Connected IoT Devices in Residential ISP Networks," *IEEE IoT Journal*, vol. 10, no. 6, pp. 5128–5145, Nov 2022.
[4] S. Garcia *et al.*, "IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic," 2020, accessed: 2025-06-26. [Online]. Available: https://zenodo.org/records/4743746
[5] E. C. P. Neto *et al.*, "CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.
[6] N. Bastian *et al.*, "ACI IoT Network Traffic Dataset 2023," https://dx.doi.org/10.21227/qacj-3x32, 2023.
[7] J. Ren *et al.*, "Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach," in *Proc. IMC*, Amsterdam, Netherlands, Oct 2019.
[8] B. Charyyev *et al.*, "IoT Traffic Flow Identification using Locality Sensitive Hashes," in *Proc. IEEE ICC*, Dublin, Ireland, Jun 2020.
[9] R. Perdisci *et al.*, "IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis," in *Proc. IEEE EuroS&P*, Genoa, Italy, Nov 2020.
[10] S. Dong, Z. Li, D. Tang, J. Chen, M. Sun, and K. Zhang, "Your Smart Home Can't Keep a Secret: Towards Automated Fingerprinting of IoT Traffic," in *Proc. ACM ASIA CCS*, Taipei, Taiwan, Oct 2020.
[11] M. Miettinen *et al.*, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," in *Proc. IEEE ICDS*, Atlanta, GA, USA, July 2017.
[12] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset," Nov. 2018.
[13] Y. Meidan *et al.*, "N-BaIoT-Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.
[14] A. Alsaedi *et al.*, "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020.
[15] M. A. Ferrag *et al.*, "Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning," *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.
[16] A. Pasquini *et al.*, "Descriptor: Deakin IoT Traffic (D-IoT)," *IEEE Data Descriptions)*, vol. 2, pp. 8–16, Mar 2025.
[17] B. M. Miele *et al.*, "VARIoT: A Dataset for IoT Traffic Analysis under Realistic Conditions," 2020.
[18] A. Sivanathan, H. Habibi Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE TMC*, vol. 18, no. 8, pp. 1745–1759, Aug 2019.
[19] A. Sivanathan *et al.*, "Managing IoT Cyber-Security Using Programmable Telemetry and Machine Learning," *IEEE TNSM*, vol. 17, no. 1, pp. 60–74, Feb 2020.
[20] A. Sivanathan, H. Habibi Gharakheili, and V. Sivaraman, "Detecting Behavioral Change of IoT Devices Using Clustering-Based Network Traffic Modeling," *IEEE IoT Journal*, vol. 7, no. 8, pp. 7295–7309, Mar 2020.
[21] S. Wannigama *et al.*, "Unveiling Behavioral Transparency of Protocols Communicated by IoT Networked Assets," in *Proc. IEEE WoWMoM*, Perth, Australia, Jun 2024.
[22] A. Hamza *et al.*, "Verifying and Monitoring IoTs Network Behavior Using MUD Profiles," *IEEE TDSC*, vol. 19, no. 1, pp. 1–18, May 2020.
[23] T. Reddy, D. Wing, and B. Anderson, "Manufacturer Usage Description (MUD) for TLS and DTLS Profiles for Internet of Things (IoT) Devices," RFC 9761, Apr 2025.
[24] J. Anand *et al.*, "PARVP: Passively Assessing Risk of Vulnerable Passwords for HTTP Authentication in Networked Cameras," in *Proc ACM CoNEXT workshop on DAI-SNAC*, Virtual Event, Germany, Dec 2021.