Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.039980
Article

MBB-IoT: Construction and Evaluation of IoT DDoS Traffic Dataset from a New Perspective

Yi Qing[1], Xiangyu Liu[2] and Yanhui Du[2,*]

[1]Department of Artificial Intelligence, People's Public Security University of China, Beijing, 100038, China
[2]Department of Cybersecurity, People's Public Security University of China, Beijing, 100038, China
*Corresponding Author: Yanhui Du. Email: duyanhui@ppsuc.edu.cn

Received: 27 February 2023; Accepted: 09 June 2023; Published: 30 August 2023

Abstract: Distributed Denial of Service (DDoS) attacks have always been a major concern in the security field. With the release of malware source codes such as BASHLITE and Mirai, Internet of Things (IoT) devices have become the new source of DDoS attacks against many Internet applications. Although there are many datasets in the field of IoT intrusion detection, such as Bot-IoT, Constrained Application Protocol–Denial of Service (CoAP-DoS), and LATAM-DDoS-IoT (some of the names of DDoS datasets), which mainly focus on DDoS attacks, the datasets describing new IoT DDoS attack scenarios are extremely rare, and only N-BaIoT and IoT-23 datasets used IoT devices as DDoS attackers in the construction process, while they did not use Internet applications as victims either. To supplement the description of the new trend of DDoS attacks in the dataset, we built an IoT environment with mainstream DDoS attack tools such as Mirai and BASHLITE being used to infect IoT devices and implement DDoS attacks against WEB servers. Then, data aggregated into a dataset named MBB-IoT were captured at WEB servers and IoT nodes. After the MBB-IoT dataset was split into a training set and a test set, it was applied to the training and testing of the Random Forests classification algorithm. The multi-class classification metrics were good and all above 90%. Secondly, in a cross-evaluation experiment based on Support Vector Machine (SVM), Light Gradient Boosting Machine (LightGBM), and Long Short Term Memory networks (LSTM) classification algorithms, the training set and test set were derived from different datasets (MBB-IoT or IoT-23), and the test performance is better when MBB-IoT is used as the training set.

Keywords: Intrusion detection; IoT; malware; botnet; DDoS; dataset

1 Introduction

In 2016, Mirai infected millions of IoT devices and launched a DDoS attack on Dyne Therapeutics, Inc. (DYN), causing huge economic losses. Subsequently, various malicious code variants based on Mirai occupied the CTI or cybersecurity reports, warnings of Computer Emergency Response

Teams (CERTs) in various regions. It can be said that Mirai has started an arms race for the construction of DDoS attacks and defense systems [1]. In 2022, according to the statistics of A10 Network Security Company, there were over 423 thousand bots predicted to be used in DDoS attacks, most of which are IoT devices [2]. A10 also pointed out that the CoAP used in IoT is considered vulnerable to IP address spoofing and data packet amplification, and IoT DDoS attacks based on this protocol are becoming a new attack trend [3]. IoT devices being controlled to launch DDoS attacks has become one of the main problems to be solved in the field of IoT security.

Datasets are the key to building a DDoS detection system. Although there are many datasets in the field of IoT intrusion detection that focus on DDoS attacks, such as Bot-IoT [4], CoAP-DoS [5], LATAM-DDoS-IoT [6], and so on, all of them take IoT devices as the attack targets in the construction process, and only N-BaIoT [7] and IoT-23 [8] take IoT devices as the attack source to generate the datasets. This indicates that the current situation of IoT devices launching DDoS attacks on Internet applications does not yet have a dataset to describe it. Therefore, to enrich the current research status of IoT DDoS protection, this literature constructs an IoT intrusion detection dataset named "MBB-IoT" from a new perspective. "MBB" represents the attack traffic launched by Mirai and BASHLITE, the two most widely spread malware in worldwide, and the normal traffic of IoT devices (Benign). Through the comparison of data statistics and model performances, the reliability of the dataset is confirmed. The specific contributions can be summarized as follows:

• A new dataset is constructed as a supplement to the description of the new trend of DDoS attacks, to truly reflect characteristics of low-rate traffic and rich traffic protocols of DDoS attacks in the part of IoT botnets and for the purpose to build an intelligent IoT firewall to control the outbound flow. The dataset is called MBB-IoT.

• The MBB-IoT dataset was analyzed and evaluated. To test whether the dataset label is representative and independent and there are adequate samples of anomalies with different labels, we set up a multi-class classification experiment in the proposed dataset. Consequently, we use cross-validation between the MBB-IoT and IoT-23 to find a model with better generalizability. And such a model proves the training set a better application performance.

The rest of this paper is structured as follows: Chapter 2 starts with known public datasets, analyzes their characteristics, and lists research questions. The third chapter introduces the construction process of the dataset based on the research questions. Chapter 4 evaluates the dataset from the perspective of data analysis and application. The fifth chapter analyzes and summarizes the contribution of this literature and the shortcomings of the dataset.

2 Related Works

Currently, mainstream datasets include: N-BaIoT [7], Bot-IoT [4], IoTID20 [9], ToN-IoT [10], IoT-23 [8], CIC IoT Dataset 2022 [11], CoAP-Dos [5], LATAM-DDoS-IoT [6]. The literature analyzes these datasets.

In essence, the above datasets are all records of traffic attributes, reflecting the **derivation**, **flow direction**, and **statistical characteristics** of benign traffic and anomaly traffic. Due to the non-uniform measurement standard of **statistical characteristics** and high dimensionality, a comparative analysis is of little significance, so it will not be considered. Combined with the actual attack scenario of a botnet remotely launching an attack, the **flow direction** of the traffic should include: forwarding to another network, which corresponds to "DFOF" in Table 1, that is, "Data from outbound flows"; The target of the attack is the Internet application server, which corresponds to "NDAT" in Table 1, that is, "Non-IoT devices attack targets". The **derivation** of traffic should include: both benign and anomaly traffic origination from actual IoT devices, corresponding to "AFFPID" and "BFFPID" in Table 1. The former is "Attack flows from physical IoT devices", the latter is "Benign flows from physical IoT devices"; more real benign traffic should also come from the participation of external users. This feature is also emphasized in the LATAM-DDoS-IoT dataset, corresponding to "BFFREU" in Table 1, that is, "Benign flows from real external users". In addition, the above-mentioned datasets are implemented in a single local area network to implement all the dataset construction steps. According to the research background, the indicator that the Internet of Things and the hacker network are in different networks should be added, which corresponds to "AAVIBR" in Table 1, that is, "Attackers and victims isolated by routers". Table 1 briefly compares MBB-IoT with the above datasets, and the analysis of each dataset is as follows:

Table 1: Data set comparison table

	DFOF	NDAT	AFFPID	BFFPID	BFFREU	AAVIBR
N-BaIoT	×	*unknown*	✓	✓	×	×
Bot-IoT	×	×	×	×	×	×
IoTID20	×	×	×	✓	×	×
ToN-IoT	×	×	×	✓	×	×
IoT-23	×	×	✓	✓	×	×
CIC2022	×	×	×	✓	✓	×
CoAP-Dos	×	×	×	✓	×	×
LATAM-DDoS	×	×	×	✓	✓	×
MBB-IoT	✓	✓	✓	✓	✓	✓

N-BaIoT: The dataset is generated from a network of nine IoT devices infected with two kinds of malware, Mirai and BASHLITE. The advantages lie in the use of physical IoT devices to generate attacks with a rich variety of devices. However, the victims in the experiment are IoT devices and located in the same local area network as the botnet, which is inconsistent with the current situation of DDoS attacks, so the authenticity of traffic has been greatly reduced, which means N-BaIoT only meets the conditions in AFFPID and BFFPID.

Bot-IoT: This dataset simulates real IoT devices through the Ostinato tool [12] and Node-red [13] and generates attack traffic using open-source penetration testing software. Its advantage is that it simulates the data imbalance characteristics of attack traffic in a real environment. However, this dataset does not use physical IoT devices for experiments, and the attack targets are also IoT devices and are in the same Local Area Network (LAN) as the attacker. The dataset does not satisfy all the metrics presented in Table 1.

IoTID20: This dataset simulates a smart home environment for traffic collection. Its advantage is that it uses physical IoT devices, and the feature distribution is relatively even. However, the attack traffic of this dataset does not come from the physical IoT devices, and the IoT devices are not taken as attack targets. The attacked and the attacker are located on the same LAN, which means IoTID20 only satisfies the condition in BFFPID.

ToN-IoT: The intrusion detection dataset is generated by introducing fog computing and rich IoT devices to form a complex testbed. Its advantage lies in the richness of device types and attack types.

However, there are also limitations that physical IoT devices are not used to generate attack traffic, and the attack target is IoT devices and is in the same local area network as the attacker, which means ToN-IoT only satisfies the condition in BFFPID.

IoT-23: The dataset is generated through a DDoS attack towards three IoT devices by the attack script carried on a Raspberry Pi. Its advantage is that the attack types are the most abundant in all public datasets, and the experimental network topology is also rich and diverse. However, there are also limitations in that the attack target is IoT devices and is in the same local area network as the attacker. Like N-BaIoT, IoT-23 only meets the conditions in AFFPID and BFFPID.

CIC IoT Dataset 2022: This dataset is composed of traffic captured by IoT devices in scenarios such as power-on, idle, interaction, and attack. The advantage is that the traffic of the entire IoT device is collected through the gateway. However, there are the same limitations as previous datasets in the generation of attack traffic, the selection of attack targets, and the design of the experimental network topology, which means CIC IoT Dataset 2022 only meets the conditions in BFFPID and BFFREU.

CoAP-DoS: This dataset introduces the CoAP protocol commonly used by IoT devices to construct the dataset. Its advantage is that it innovates the protocol types involved in DDoS attacks. However, there are the same limitations as previous datasets in the generation of attack traffic, the selection of attack targets, and the design of the experimental network topology. Although this data set is a special data set for DDoS attacks, only BFFPID has been satisfied.

LATAM-DDoS-IoT: This data set introduces IoT devices used by external users, making traffic more authentic. However, the attack traffic comes from open-source penetration testing software, and no real IoT devices are used for the attack. The attack targets are also IoT devices, which are contrary to the status quo of DDoS attacks, which means LATAM-DDoS-IoT only meets the conditions in BFFPID and BFFREU.

Based on the above analysis, constructing a real IoT and network traffic dataset that includes recent DDoS attack scenarios is still a topic that needs to be supplemented. More importantly, there is no dataset involved in the description of the flow of IoT DDoS attack traffic to different networks and an Internet application server. To enrich the research status of this dataset, this literature constructs the MBB-IoT dataset.

3 MBB-IoT Dataset Construction

3.1 Summary of Design Ideas

The main idea of MBB-IoT dataset construction is:

(1) The core idea of **building an IoT experimental scene** is to use real IoT devices to avoid all victims being located in the same local area network, and to generate normal traffic as close to reality as possible, which corresponds to the three evaluation indicators of BFFPID, BFFREU, and AAVIBR in Table 1. Specifically, firstly, the experimental IoT device should be a mainstream IoT device with security vulnerabilities, but protected by the internal network isolation of the router, and can implement services and interact with the device through an Internet application account, and finally realize the simulation of two network scenarios of standby and high traffic, and the high traffic scenario is used to generate the normal part of the attack traffic, and the standby scenario is used for the comparison of the experiment and to ensure the comprehensiveness of the dataset. Secondly, connecting devices such as switches and routers should have practical significance and be coordinated with the access points of the broadband access network.

(2) **Infiltrate the Internet of Things, implant malicious codes, and build botnets**. Since it is unrealistic to build a complete and widely distributed botnet, we can only infer the overall characteristics of IoT DDoS attacks by building a small network to reflect the local characteristics of the botnet. The core idea is to create an environment for hackers to remotely control actual IoT devices and build a small IoT botnet, corresponding to the AFFPID construction requirements in Table 1. Specifically, first of all, the most widely used IoT penetration method should be selected, that is, to infiltrate the IoT intranet through routers [14]. Although this is an attack on IoT devices, it does not meet the main content of the dataset constructed in the paper (DDoS as the theme), and it is difficult to collect the traffic of a single IoT device in a wireless environment, so it is not necessary to include penetration traffic into the dataset. Second, choose the most representative malware to implant into the IoT device. The implantation process should be the same as the actual attack, download it to the corresponding IoT device through the cloud server as an intermediary, and then run it. This will preserve the communication between the device and the cloud;

(3) **To control the Internet of Things to attack Internet applications**, the core idea is that the victim should be set as a non-Internet of Things target, which corresponds to the construction requirements of NDAT in Table 1. Specifically, first, there is at least one hop between the Internet application server and the Internet of Things, and because the Internet of Things devices are protected by the intranet, the Internet of Things devices cannot be pinged from the Internet application server. Secondly, the corresponding botnet attack scenarios should be set according to the classification of malware variant types, that is, the scenario of a single device attacking an Internet server designed to simulate a botnet controlled by malware for a certain type of IoT devices, and the scenario of multiple devices attacking an Internet server at the same time designed to simulate a botnet controlled by malware for a certain vulnerability.

(4) **Collecting network traffic at different nodes**, the core idea is to capture the traffic sent by the IoT gateway and the traffic received by the Internet application server, that is to say, to meet the construction requirements of DFOF in Table 1. Although the packet capture for the traffic sent by a certain device reflects the characteristics of the IoT device more obviously, the captured traffic will not meet the actual situation faced by the intrusion detection system, so the outgoing traffic should be copied at the gateway to the sniffing device for capture, and the traffic in the Internet server should also be captured.

3.2 Experimental Scene Construction

According to the design idea of Section 3.1, the experimental network topology is constructed as shown in Fig. 1. For reflecting the typical application scenarios of the Internet of Things, the literature selects commonly used smart home devices to construct the mainstream network topology. The interior specifically includes two independent local area networks including the Internet of Things composed of five smart devices and the hacker's network. The WEB server is accessed by the two networks through the core switch. All their traffic converges on the Internet, so they can access the instructions and control server (Command-and-control, C&C). In addition, smartphones, smart speakers, and smart gateways in the Internet of Things are all wireless network access devices. At the same time, smart speakers and smart gateways are controlled by the LAN software of the Raspberry Pi.

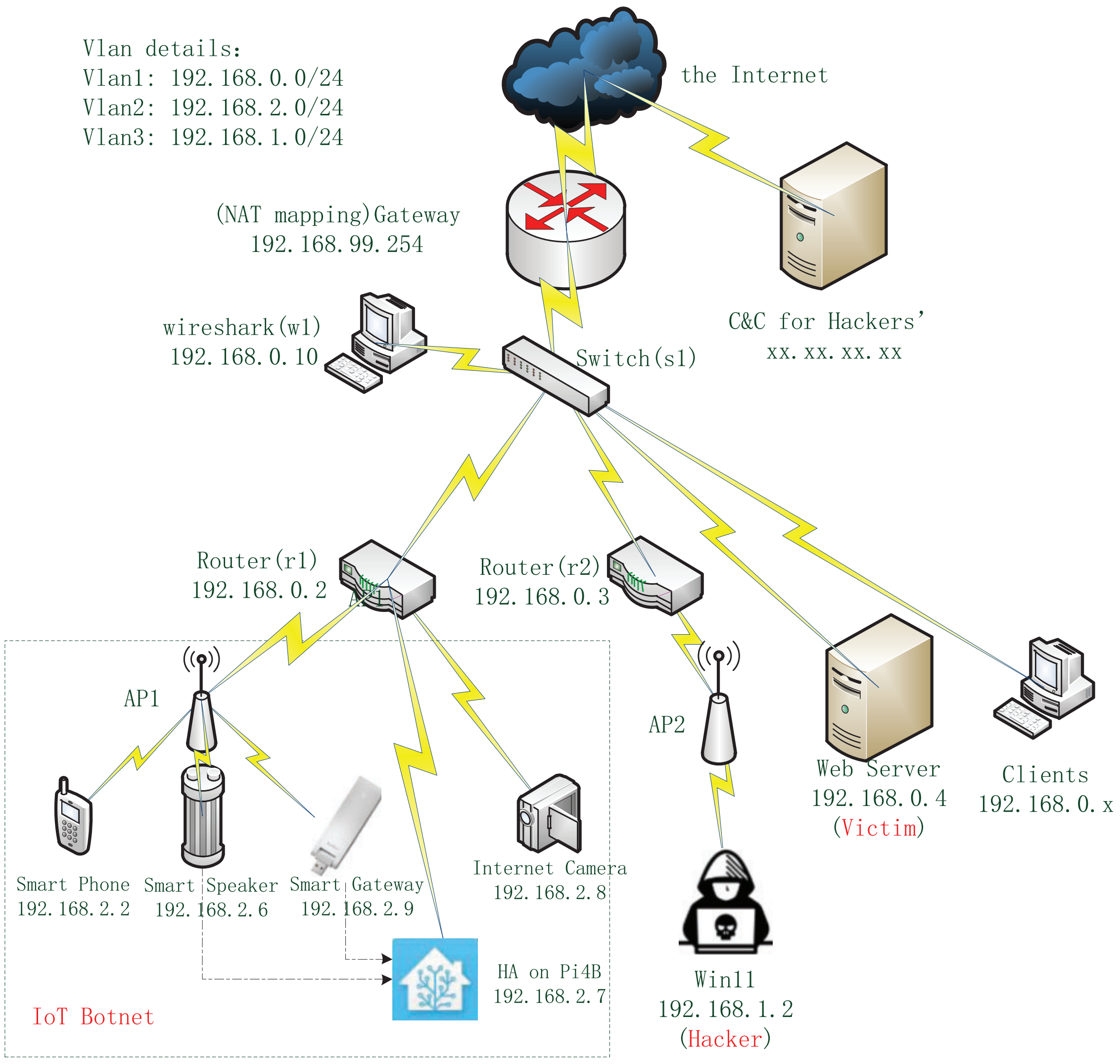

Figure 1: Experimental network topology of MBB-IoT dataset

According to the network activity, two network scenarios of standby and high traffic are constructed. The first is the **standby scenario**. Each IoT device is kept on standby, and the client's access to the WEB server is stopped. The second is the **high-traffic scenario**. In the Internet of things, the smartphone logs in to the app to watch the live broadcast, the smart speaker logs in to the Xiaomi account to play the song, and the Raspberry PI logs in to the Homeassistant account to enable the LAN control of the smart speaker and the smart gateway. For the WEB server, the python script is used to simulate the browser to access the WEB server in the personal computer (PC) client. A random pause of 1–15 s follows each visit. In addition, the resource subnet details of the experimental network are shown in Table 2, and the communication subnet details are shown in Table 3.

Table 2: Resource subnet details

Name	Model	IP address
Smartphone	Vivo neo5	192.168.2.2
Internet camera	EZVIZ CS-C6C-3B2WFR	192.168.2.8
Smart speaker	Xiaomi lx05	192.168.2.6
Smart gateway	Aqara Smart Hub E1	192.168.2.9
Raspberry Pi	4b (2 gb wifi version)	192.168.2.7
Hacker's laptop	Lenovo Xiaoxin 7000	192.168.1.2
C&C	Huawei elastic cloud server	xx.xx.xx.xx
WEB server	Lenovo desktop A4600K	192.168.0.4
WEB clients	Lenovo desktop A4600K	192.168.0.x
Wireshark device (w1)	Lenovo desktop A4600K	192.168.0.10

Table 3: Communication subnet details

Name	Model	IP address or description
Switches (s1)	Ruijie S2910 switch	Layer 2 switches
Routers (r1, r2)	Ruijie RSR20 router	192.168.0.2, 192.168.0.3
AP1, AP2	rg-ap850	Wireless Access Points
Hub (h1)	Speedlink ethernet hub	Physical Layer device

3.3 Botnet Construction

According to the research background, Mirai and BASHLITE are the most influential malicious code frameworks, and the source codes have been disclosed by hackers on the Internet. This literature downloads the original version of Mirai from GitHub [15] and the variant of BASHLITE (gummy) from the "crack.io" website [16]. The code is statically compiled and linked on the C&C cloud server to obtain executable files for various architectures. Then obtain the root authority of the router by brute-force attack to infiltrate the intranet, so that the "wget" or "tftp" command could be used to download the executable files from the C&C cloud server to the three IoT devices in Fig. 1, the smart speaker, the smart gateway and the network camera. After running the executable files, the malware implantation of the IoT devices comes to the end. This literature selects attack types according to each network layer, see Tables 4 and 5 for details.

3.4 Attack Scenario Construction

The two kinds of malware used in the experimental botnet construction have many variants circulating in the network, which can be roughly divided into malware targeting **a certain type of IoT device** and **malware targeting a certain vulnerability indiscriminately**. Therefore, combined with the research background of IoT devices launching attacks on Internet application servers, two attack scenarios are set up:

Table 4: Mirai attack types

Mirai attack types	Layer number (OSI model)	Descriptions
User datagram protocol (UDP) plain flood	4	UDP flood with fewer options. Optimized for higher PPS
SYN flood	4	Send repeated SYN packets to every port on the targeted server
Generic routing encapsulation (GRE) IP flood	3	Encapsulate IP messages inside virtual point-to-point links
GRE ethernet flood	2	Encapsulate ethernet frame inside virtual point-to-point links
Hypertext transfer protocol (HTTP) flood	7	Send repeated HTTP responses

Table 5: BASHLITE attack type

BASHLITE attack types	Layer number (OSI model)	Descriptions
Transmission control protocol (TCP) flood	4	Send repeated TCP packets with all flags to a specific port on the target server
UDP flood	4	Send repeated UDP packets to a specific port on the target server
Rand hex	7	Send a large amounts of spam data to the target server
Combo	4	Send spam data and build a UDP/TCP connection with the target server

(1) **A single device attacks the WEB server**. It is used to simulate the attack launched by the Internet of Things botnet controlled by the **first type of malware**. The hacker host connects to the C&C server to control one of the cameras, speakers, and gateways to launch the DDoS attacks listed in Tables 4 and 5 one by one on the WEB server. For Mirai, the experimental input attack command is "udpplain 192.168.0.4 3600" (taking UDP plain flood attack as an example), this command corresponds to the format of "attack type + destination address + duration", and "udpplain" is the abbreviation of attack type; "192.168.0.4" is the IP address of the WEB server, and Mirai also supports DNS resolution, so the domain name of the WEB server can also be filled in here; "3600" represents the duration of a single attack in seconds, which can be set any value from 1 to 3600. For BASHLITE, the experimental input attack command is similar to Mirai, such as "tcp 192.168.0.4 3600 80" (take TCP flood as an example), and the port number of the attack is added at the end. Since the experiment launches attacks against the WEB server, the port number is set to 80. Since the upper limit of each attack time of the two malware is 3600 s, "3600 + 600 = 4200" seconds is used as the cycle of each attack to simulate the discontinuity of the hacker's attack, and at the same time reduce the impact of network delay on packet capture;

(2) **Multiple devices attack the WEB server at the same time**. It is used to simulate the attack launched by the Internet of Things botnet controlled by the **second type of malware**. The hacker host connects to the C&C server and controls three devices to launch the DDoS attacks listed in Tables 4 and 5 one by one. The attack commands are the same as those shown in (1). The attack period is also set to 4200 s.

3.5 Network Traffic Collection

There are two key points in the traffic description of IoT devices as the attack source, namely the IoT gateway and the attack target. It can be known from "DFOF" in Table 1 that the traffic sent from the Internet of Things to the attacked network should be captured, so the following two packet capture operations are performed:

(1) **Capture the original traffic of IoT DDoS attacks through port mirroring**. Set up port mirroring in the core switch (s1), copy all the outbound traffic of the Internet of Things and forward it to the network sniffing host (w1) equipped with Wireshark, and capture data respectively in each attack cycle (4200 s), to realize the native IoT DDoS attack traffic capture, which includes attack packets generated by infected IoT devices, normal traffic of IoT services, and a small amount of traffic generated by network routing;

(2) **Capture the composite traffic of the IoT DDoS attack on the WEB server side**. Open Wireshark on the server equipped with WEB applications, and capture all received traffic in each attack cycle (4200 s), including attack traffic from the Internet of Things, access load from clients, and a small amount of traffic generated by different network exchanges.

Last but not least, to ensure the integrity of the experiment, it is also necessary to capture the normal traffic sent by the two basic scenarios of standby and high traffic constructed in Section 3.2: stop the attack and capture the normal traffic when the network activity is stopped or the network activity is in progress according to the same packet capture principle. Traffic saved as "benign1" and "benign2" respectively. Use the "tree" command to view the packet capture folder structure as follows:

```
G:\IOT_DATASET\MBB-IOT
 ├──lab-outbound
 │   ├──BASHLITE
 │   ├──benign
 │   └──mirai
 └──lab-web
     ├──BASHLITE
     ├──benign
     └──mirai
```

Among them, the BASHLITE and Mirai folders contain Packet Capture (PCAP) files named "'device name'_'DDos attack type'_'malicious code type' _70 min_'out/web'.pcap", and the benign folder contains "benign1" Two PCAP files with "benign2". According to the attack type, the data volume is counted, and the results are shown in Table 6.

Table 6: Data collective volume statistics

	Outbound's (GB)	WEB server (GB)
Mirai UDP plain flood	10.2	3.62
Mirai SYN flood	9.27	4.03
Mirai GRE IP flood	6.63	4.29
Mirai GRE ethernet flood	8.52	4.11
Mirai HTTP flood	5.53	4.25
BASHLITE TCP flood	7.76	10.5
BASHLITE UDP flood	32.3	29.5
BASHLITE Rand hex	17.2	16.9
BASHLITE combo	31.4	30.3
Benign1	0.558	0.817
Benign2	1.54	0.952

4 MBB-IoT Dataset Analysis and Evaluation

4.1 MBB-IoT Dataset Analysis

4.1.1 Description of Low-Rate DDoS Attacks in Local IoT Botnet

The packet-sending ability of IoT devices is much lower than that of general desktop computers, and the collected data sets also reflect the actual characteristics of IoT traffic. Therefore, the average traffic and packet sending speed of the MBB-IoT dataset should be much smaller than that of the data set that simulates the DDoS attack of the IoT using a desktop computer equipped with virtual software. To quantify this feature, check the captured file properties through Wireshark [17] to obtain the average packet transmission speed, average packet size, and average bit rate. These three statistics are often used to measure the size of the captured traffic [18–20], that is, the experimental network throughput. Most datasets provide captured PCAP files (data files created using a network sniffer) as well as Comma-separated Values (CSV) files. BoT-IoT [4] provides the DDoS attack traffic received by the entire Internet of Things for comparison with MBB-IoT attack scenario 2. See Tables 7 and 8 for detailed information. CIC IoT Dataset 2022 [11] provides the traffic files of specific devices under attack. Like MBB-IoT, the experiment uses network cameras and gateways. The statistical information of corresponding devices is shown in Tables 9–12.

Table 7: Bot-IoT packet information

Type (attacker: hping3)	Average (Avg). packets/sec	Avg. packet size (bytes)	Avg. bytes/sec
DDoS_HTTP	8260.852	655	5410 k
DDoS_TCP	17081.338	146	2494 k
DDoS_UDP	31777.181	209.36	6653 k
(Avg)	19039.790	336.787	4852 k

Table 8: MBB-IoT dataset scenario 2 packet information

Type (attacker:3 IoT devices)	Avg. packets/sec	Avg. packet size (bytes)	Avg. bytes/sec
DDoS_HTTP	942.5	768	723 k
DDoS_TCP	2188.1	1230	2691 k
DDoS_UDP	1396.9	1169	1633 k
(Avg)	1509.2	1056	1682 k

Table 9: CIC IoT dataset smart gateway packet information

Type (attacker: LOIC)	Avg. packets/sec	Avg. packet size (bytes)	Avg. bytes/sec
DDoS_HTTP	3960.257	66	261 k
DDoS_TCP	4285.110	952.74	4083 k
DDoS_UDP	4577.772	74	339 k
(Avg)	4274.380	364.247	1561 k

Table 10: MBB-IoT smart gateway data packet information

Type (attacker: gateway)	Avg. packets/sec	Avg. packet size (bytes)	Avg. bytes/sec
Mirai GRE Ethernet flood	947	676.26	640 k
Mirai GRE IP flood	922	529.55	488 k
Mirai HTTP flood	802	684.88	549 k
Mirai SYN flood	1145	1005.68	1151 k
Mirai UDP plain flood	1103	1013.15	1118 k
Bashlite TCP flood	2152	1210.52	2532 k
Bashlite UDP hex	2173	1176.72	2555 k
Bashlite Rand hex	3295	276.04	909 k
Bashlite UDP flood	2088	1240.67	2591 k

Table 11: CIC IoT dataset webcam packet information

Type (attacker: LOIC)	Avg. packets/sec	Avg. packet size (bytes)	Avg. bytes/sec
DDoS_HTTP	2155.8	765	1648 k
DDoS_TCP	2016.5	756	1525 k
DDoS_UDP	1859.7	764	1421 k
(Avg)	2010.7	762	1531 k

Table 12: MBB-IoT webcam packet information

Type (attacker: Camera)	Avg. packets/sec	Avg. packet size (bytes)	Avg. bytes/sec
Mirai GRE Ethernet flood	67	668.53	45 k
Mirai GRE IP flood	72	682.91	49 k
Mirai HTTP flood	63	651.71	41 k
Mirai SYN flood	113	677.46	76 k
Mirai UDP plain flood	108	663.85	72 k
Bashlite TCP flood	812	1221.54	992 k
Bashlite UDP hex	822	1200.32	987 k
Bashlite Randhex	1539	241.35	371 k
Bashlite UDP flood	802	1238.16	993 k

The longitudinal comparison shows that, due to different Central Processing Unit (CPU) capabilities, the average packet transmission speed (PPS) of simulated attacks using open-source software on desktops is much higher than that of attacks from Internet of Things devices, while the average packet size and average bit rate are affected by data link bandwidth and other factors, and do not show any regularities. Meanwhile, the horizontal comparison shows that DDoS attacks at the same logical layer (layer4: TCP flood & UDP flood; layer5: GRE IP; layer6: GRE eth; layer7: HTTP) have similar average packet transmission speed, average packet size and average bit rate on a certain device, with less variation across IoT devices and greater variation with desktops.

The average packet transmission speed, average packet size, and average bit rate have a significant impact on the features extracted by many popular feature extraction software such as Netflow [21], CICFlowMeter [22], NFStream [23]. Therefore, it is reasonable to believe that the study of attack traffic generated by real physical network devices can improve the intrusion detection system. The dataset prepared in this paper and the method of making the dataset are valuable.

4.1.2 Description of Rich Protocols in IoT DDoS Attacks Traffic

Due to the variety of IoT devices, manufacturers do not have a unified production standard, and data packets sent by IoT devices contain multiple protocol types. Through the protocol layer analysis function of Wireshark [17], the collected data packet protocol layer could be counted, and its percentage is also calculated, which has a quantitative display of the protocol richness and distribution characteristics. However, this kind of statistic is more intuitive only when the traffic contains fewer data loads because the number of bits occupied by the protocol header is much smaller than the data loads. When the traffic contains a large data load, the order of magnitude of the proportion of each protocol will become very small. To more intuitively compare the difference between MBB-IoT and public datasets, Table 13 is made according to the number of protocol types contained in different attack types of different datasets.

Obviously, the complexity of the protocol stack of the data packet obtained by using the open source penetration testing software in the desktop computer is much lower than that of the DDoS attack carried out by the IoT device having been infected. This is also one of the reasons why IoT Profiling has become a relatively hot research field recently. The various services carried out by various brands of IoT devices lead to complex traffic composition, so the construction of datasets must use real IoT devices in real scenarios. The MBB-IoT dataset truly reflects the rich characteristics of IoT DDoS attack traffic protocols.

Table 13: Comparison of protocol complexity

	Flow types	Tcp protocol suite	Udp protocol suite	Others
Bot-IoT	TCP flood	2	2	2
	UDP flood	1	2	1
	HTTP flood	2	1	1
CIC IoT dataset	TCP flood	1	1	2
	UDP flood	0	1	1
	HTTP flood	1	1	2
MBB-IoT	TCP flood	9	14	9
	UDP flood	14	9	9
	HTTP flood	12	9	9

4.2 MBB-IoT Dataset Application Evaluation

By analyzing the detection application of DDoS attacks launched by the Internet of Things, the support of this dataset for subsequent applications is demonstrated. In this section, two stages of experimental analysis will be carried out. First, the training and testing are carried out inside the dataset, and the evaluation conclusions on the quality of the MBB-IoT data set are obtained by multi-class classification; Second, the model trained by MBB-IoT is verified by using the public comprehensive dataset as the test set. At the same time, the model trained by the comprehensive dataset is verified by MBB-IoT.

To complete the first phase of the experiment, the specific model which may have a more outstanding state-of-the-art than other classic models should be selected, and [24] proved the Random Forest to be a good choice. Moving on to the performance evaluation, mean precision (P), mean recall (R), and mean f1 score (F) are utilized as evaluation criteria. To complete the second phase of the experiment, it should be considered that the heterogeneity of the data set is better than that of MBB-IoT, and it is easy to find a common feature subset. IoT-23 [8] contains the most abundant attack types and the most diverse experimental scenarios in the current public datasets. The features contained in it are shown in Appendix B. Through conversion, we can get "ip_num_per_mac", "src2dst_first_seen_ms", "src2dst_last_seen_ms", "src_port" and other features in Table 14 and Appendix A that perform well in feature selection, so IoT-23 meet the conditions for cross-validation.

4.2.1 Data Processing and Analysis

Data processing refers to the process of extracting features from the PCAP file obtained in the experiment and performing a series of preprocessing operations, including feature generation, feature extraction, labeling, and preprocessing, and then performing a certain algorithm or statistical knowledge on the obtained feature files to remove features with weak influence on machine learning to improve the quality of the model and reduce the training cost.

Table 14: List of new features

Feature	Data type	Description
ip_num_per_mac	Int	IP types grouped by source mac address before the flow is idle
if_same_vlan_or_not	Bool	If the vlan_id is the same as the flows whose source IP is the attack target
multi_category_name_&confidence	Int	$7x + 8y$, x from "category_name" feature, y from confidence feature

(1) feature generation

There are many network traffic feature generation tools to choose from, including YAF [25], pmacct [26], CAIDA CoralReef [27], SoftFlowd [28], nProbe [29], etc. However, they are often not comprehensive enough, and some cannot perform real-time packet parsing. Some do not parse the application layer, and some do not support decoding the encapsulation protocol. The feature selection of the MBB-IoT dataset is based on the latest network flow analysis framework—NFStream [23], which has the advantages of comprehensive functions, and good scalability, and supports the expansion of machine learning algorithms for traffic analysis. The MBB-IoT data set selects the basic information generated by the framework, tunnel decoding, application layer dialysis, and time series statistical features as a preliminary data set. There are 87 features in total. See Appendix A for specific data types and feature descriptions.

(2) feature extraction

To make the generated features better describe the trend and pattern of DDoS attacks, three additional features are added as shown in Table 14. "ip_num_per_mac" describes the characteristics of DDos attack random source IP address, and has a trend of increasing in stages; "if_same_vlan_or_not" describes the distribution trend of DDoS attacks originating from external network IP addresses, including the authenticated communication between hackers and infected devices through routers; "multi_category_name_&confidence" further makes the attributes parsed by the application layer of the NFStream framework unique and having a certain distribution trend through function mapping.

(3) labeling

From a macro point of view, traffic is divided into abnormal traffic and normal traffic, and has different attack types at the same time, so two sets of labels can be set. One is the "benign" and "anomaly" labels for binary classification, and the other is the specific attack type and the "benign" label for multi-class classification.

From a microscopic point of view, the maximum attack time of malicious codes Mirai and BASHLITE is 60 min, and the experiment time is 70 min, so the first 86% of the traffic is marked as an "anomaly", and the last 14% of the traffic is marked as "benign".

(4) preprocessing

Through the above operations, the number of features has reached 90, and certain preprocessing is required. In the first stage, the characteristics of problematic data types are removed. The feature data in the first column of Table 15 is with poor data performance, and there are many null values. The features in the second column are difficult to convert into numeric values. The features in Table 15 are removed.

Table 15: The feature list removed in the first stage of preprocessing

Features with poor data performance	Features that cannot map into integer or float
client_fingerprint, server_fingerprint, user_agent, content_type	src_ip, src_mac, src_oui, dst_ip, dst_mac, dst_oui, application_name, requested_server_name

In the second stage, the problem of the distribution of each eigenvalue is solved. Remove features whose eigenvalues are all of a certain value, and features that are highly relevant to model training but meaningless. See Table 16 for specific removed features.

Table 16: The feature list removed in the second stage of preprocessing

High categorical features	Meaningless label
bidirectional_ece_packets, src2dst_cwr_packets, bidirectional_cwr_packets, src2dst_ece_packets, if_same_vlan_or_not, dst2src_urg_packets, dst2src_ece_packets, dst2src_cwr_packets, vlan_id	Id

After preprocessing, 68 columns of features are left, and the data obtained from the two experiments are merged into two large CSV files, namely "outbound.csv" and "web_server.csv".

(5) feature selection

For the design of data analysis, the focus is on feature selection. Feature selection is not a necessary process in general machine learning, but in the field of intrusion detection, feature selection is often a step that cannot be bypassed. Because the data generally has a high dimensionality, the proportion of positive samples and negative samples is extremely inconsistent, and there is often a high correlation between the generated features, which will have a greater impact on the training effect of machine learning, and it is necessary to remove features that do not meet the requirements. There are generally three methods for feature selection, namely the filter method, wrapper method, and embedded method. The filter method generally uses statistical methods to evaluate the correlation between features to remove features that do not meet the requirements. The computational complexity of this method is not high. However, the stability is poor, and the optimal feature selection of the training algorithm often could not be achieved [29]; Wrapper method uses a specific machine learning algorithm to perform a subset search and quality evaluation of the feature set, and its stability is better than filter method, but its computational complexity is high [29], while the embedded method is equivalent to the complex of the first two methods, which has better stability and lower computational complexity [30]. At present, the feature selection of most IoT intrusion detection datasets uses the filter method or wrapper method [31]. From the perspective of comprehensive performance and stability, it should be optimized to the more advanced embedded method to ensure that the data set shows the best effect in the application evaluation.

It can be seen from [24] that random forest is better than Naive Bayes (NB), Principal Component Analysis (PCA), and other methods, and has an advantage in efficiency compared with the genetic algorithm [32]. Therefore, this paper uses the random forest to select MBB-IoT datasets.

The first step is to use 40% of the data to train the random forest, temporarily remove the multi-category label "types", and arrange them in order of size according to the Gini importance to obtain the feature importance distribution map of each feature of the data set, as shown in Fig. 2. Select according to the first five gradients, that is, select features until "dst_port". The specific characteristics are shown in Table 17.

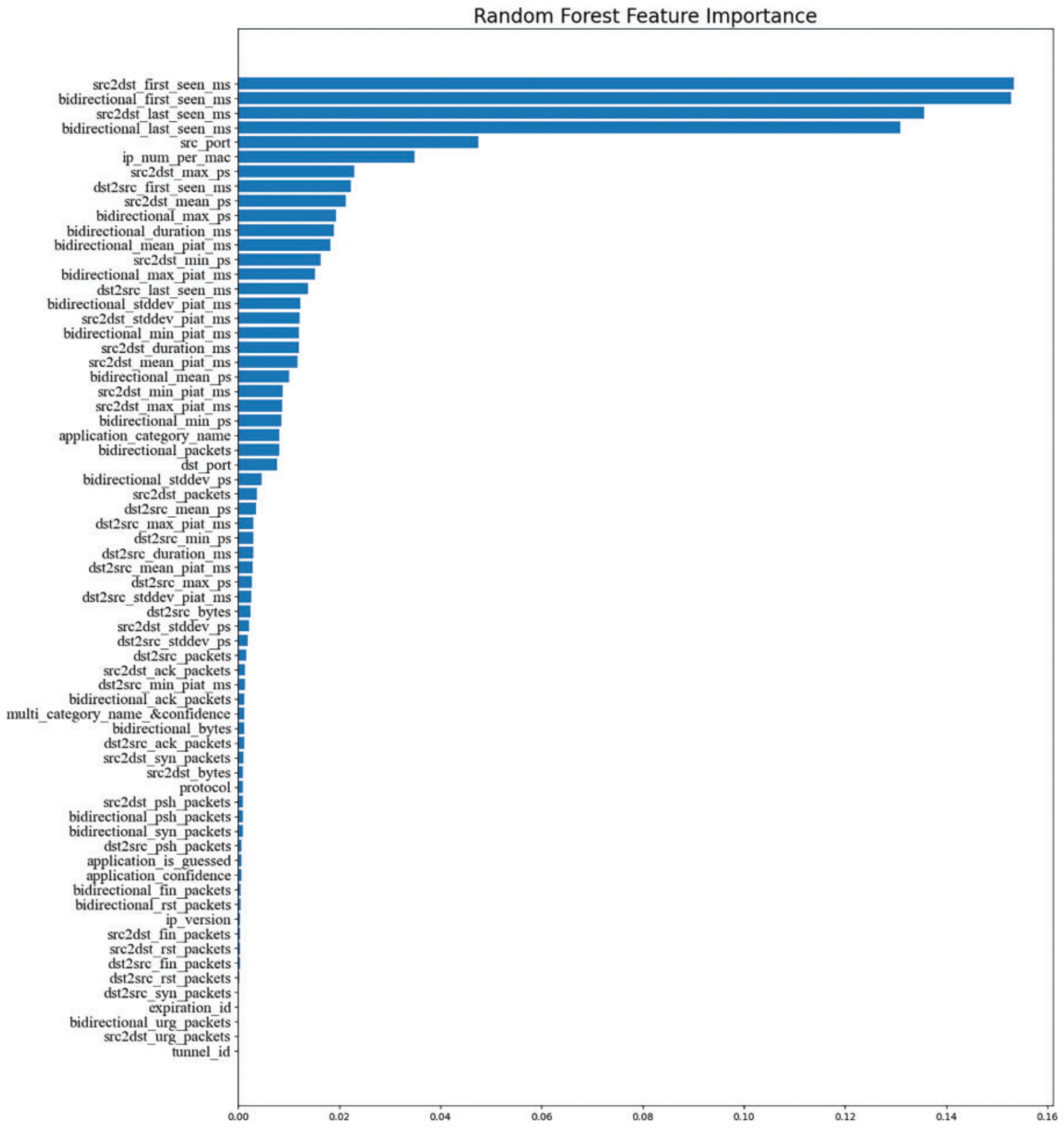

Figure 2: Random forest feature importance histogram

Table 17: Feature selection features

Selection in Fig. 2	Number of features
src2dst_first_seen_ms, bidirectional_first_seen_ms, src2dst_last_seen_ms, bidirectional_last_seen_ms, src_port, ip_num_per_mac, src2dst_max_ps, dst2src_first_seen_ms, src2dst_mean_ps, bidirectional_max_ps, bidirectional_duration_ms, bidirectional_mean_piat_ms, src2dst_min_ps, bidirectional_max_piat_ms, dst2src_last_seen_ms, bidirectional_stddev_piat_ms, src2dst_stddev_piat_ms, bidirectional_min_piat_ms src2dst_duration_ms, src2dst_mean_piat_ms bidirectional_mean_ps, src2dst_min_piat_ms src2dst_max_piat_ms, bidirectional_min_ps application_category_name, bidirectional_packets dst_port	27

4.2.2 Dataset Internal Evaluation

Abundant data samples for every category is a basic requirement of high-quality datasets, and also facilitates the modeling of machine learning algorithms. Especially, MBB-IoT dataset contains various data of attack types which are shown in Tables 4 and 5 and their labels need to be substituted by 1–9. Obviously, the label of benign samples could be set as ‘0’. After removing the binary labels, the dataset is divided into a training set and a test set according to 8:2. The training set is used to build a Random Forest model, which are evaluated on the remaining test and the performance could be seen in Table 18.

Test results show an excellent performance as all the attack types attain high P, R, and F scores, which means labels are representative and independent and the data samples in different labels are linearly separable from each other, so the dataset internal evaluation can give an evidence about dataset quality.

4.2.3 Dataset Cross-Evaluation

The methods used are SVM [33], LightGBM [34], and LSTM [35], which cover gradient, tree-related algorithms, and deep learning algorithms, and can achieve a more comprehensive evaluation of the dataset. The training set and test set are intercepted from different datasets, and the performance of the two datasets is comprehensively evaluated by comparing the parameters Accuracy, F1 score, and

AUC. The results are shown in Tables 19–21. First of all, it can be seen that although the accuracy of the model obtained by MBB-IoT is only slightly higher than that of IoT-23, F1 score and AUC are much higher than those of IoT-23, which means the misjudgment rate of the model based on the MBB-IoT dataset is much lower. In general, the test performance based on the MBB-IoT dataset is better under the same model.

Table 18: Multi-class classification performance with random forest

Criteria labels	Precision (%)	Recall (%)	F1 score (%)
UDP plain flood	88.63	94.56	91.50
SYN flood	89.22	95.06	92.05
GRE IP flood	90.14	93.57	91.82
GRE ethernet flood	89.85	94.01	91.88
HTTP flood	89.48	93.93	91.65
TCP flood	100.0	100.0	100.0
UDP flood	92.27	97.99	95.04
Rand hex	88.11	93.14	90.56
Combo	89.44	94.42	91.86
Benign	100.0	100.0	100.0

Table 19: Cross-validation of SVM machine learning algorithms

SVM	Test on the other (MBB-IoT or IoT-23)		
	Accuracy (%)	F1 (%)	AUC (%)
Train on MBB-IoT	80.256	75.654	79.189
Train on IoT-23	78.591	50.854	55.645

Table 20: LightGBM machine learning algorithm cross-validation

LightGBM	Test on the other (MBB-IoT or IoT-23)		
	Accuracy (%)	F1 (%)	AUC (%)
Train on MBB-IoT	85.346	79.654	78.128
Train on IoT-23	81.654	65.265	66.785

In the literature of ToN-IoT [10], it comes to the point that the cross-validation result of the dataset is better than IoT-23 due to its heterogeneity better than IoT-23; XeNIDS intrusion detection system based on various cross-validation scenarios also attribute the poor performance of cross-validation results to the lower heterogeneity of training set than test set [36]. However, the cross-validation results of this literature show that the MBB-IoT dataset with less heterogeneity than IoT-23 performs better. Indeed, the heterogeneity of the dataset is a very important factor, but the model interpretation in the cross-validation process needs to be further explored.

Table 21: LSTM deep learning algorithms cross-validation

LSTM	Test on the other (MBB-IoT or IoT-23)		
	Accuracy (%)	F1 (%)	AUC (%)
Train on MBB-IoT	70.154	70.062	70.365
Train on IoT-23	61.897	60.951	61.902

5 Conclusion

It is one of the main problems in the field of IoT security that IoT devices launch DDoS attacks on Internet applications. However, the data set describing the DDoS attack scenario is relatively scarce at present. To enrich the research status of the Internet of Things DDoS attack data set, this paper proposes a dataset named "MBB-IoT" containing the attack traffic of malware Mirai and BASHLITE and the normal traffic of the Internet of Things. The dataset is derived from DDoS attack traffic launched by controlled IoT devices, which ensures that IoT is protected by an internal network and the attack target is a WEB server of another network. The MBB-IoT dataset provides two versions of DDoS native traffic and server-side mixed traffic. In data processing, features describing DDoS attack characteristics are proposed, and the new features show high importance for model training in feature selection based on random forest. After analysis, MBB-IoT truly reflects the characteristics of low-rate traffic and rich traffic protocols in the implementation of DDoS attacks in the Internet of Things. Finally, the paper also evaluates the application of machine learning algorithms on the MBB-IoT dataset. MBB-IoT dataset is divided into a training set and a test set, and then applied to Random Forests and MLP classification algorithm for training and testing. The test results are better than LATAM-DDoS-IoT. Secondly, using SVM, LightGBM, and LSTM algorithms to cross-evaluate MBB-IoT and IoT-23 across datasets, the detection performance is better when MBB-IoT is used as a training set. However, because of the limitations of the number of devices and the environment, there is still much room for improvement in the dataset, and it is a direction worth studying in terms of the extent to which the semantic gap is resolved, and more in-depth research will continue in the future.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. Hughes, EMEA and A10 Networks, "The DDoS arms race," 2016. [Online]. Available: https://m.digitalisationworld.com/blogs/49783/the-ddos-arms-race

[2] A10 Company System Engineers, "A10 networks DDoS threat report," 2022. [Online]. Available: https://www.a10networks.com/resources/reports/2022-ddos-threat-report/

[3] Bitdefender Company and F. TRUŢĂ, "IoT devices a growing part of global DDoS weapon arsenals," 2022. [Online]. Available: https://www.bitdefender.com/blog/hotforsecurity/iot-devices-growing-part-global-ddos-weapon-arsenals/

[4] 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," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[5] J. Mathews, P. Chatterjee and S. Banik, "CoAP-DoS: An IoT network intrusion data set," in *2022 6th Int. Conf. on Cryptography, Security and Privacy (CSP)*, Tianjin, China, pp. 91–95, 2022.

[6] J. G. Almaraz-Rivera, J. A. Perez-Diaz, J. A. Cantoral-Ceballos, J. F. Botero and L. A. Trejo, "Toward the protection of IoT networks: Introducing the LATAM-DDoS-IoT dataset," *IEEE Access*, vol. 10, pp. 106909–106920, 2022.

[7] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai *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.

[8] S. Garcia, A. Parmisano and M. J. Erquiaga, *IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0)*. Zenodo, 2020.

[9] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in IoT networks," in *Advances in Artificial Intelligence: 33rd Canadian Conf. on Artificial Intelligence*, Canadian AI 2020, Ottawa, ON, Canada, pp. 13–15, 2020.

[10] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa and F. T. den Hartog, "ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets," *IEEE Internet of Things Journal*, vol. 9, pp. 485–496, 2021.

[11] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong *et al.,* "Towards the development of a realistic multidimensional IoT profiling dataset," in *19th Annual Int. Conf. on Privacy, Security & Trust (PST)*, Fredericton, NB, Canada, vol. 1, pp. 1–11, 2022.

[12] B. R. Patil, M. Moharir, P. K. Mohanty, G. Shobha and S. Sajeev, "Ostinato-a powerful traffic generator," in *2nd Int. Conf. on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, Bengaluru, India, vol. 1, pp. 1–5, 2017.

[13] OpenJS Foundation and Node-RED contributors, "Node-RED: Low-code programming for event-driven applications," 2022. [Online]. Available: https://nodered.org/

[14] G. J. Blinowski and P. Piotrowski, "CVE based classification of vulnerable IoT systems," in *Theory and Applications of Dependable Computer Systems: Proc. of the Fifteenth Int. Conf. on Dependability of Computer Systems DepCoS-RELCOMEX*, Brunów, Poland, Springer, pp. 82–93, 2020.

[15] J. Gamblin, "Mirai-source-code: Leaked Mirai source code for research/IOC development purposes," 2017. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code

[16] J. Netizen, Leaked hybrid botnet gummy qbot/Mirai ultra HQ. 2022. [Online]. Available: https://cracked.io/Thread-LEAKED-HYBRID-BOTNET-GUMMY-QBOT-MIRAI-ULTRA-HQ

[17] U. Lamping and E. Warnicke, "Wireshark user's guide," *Interface*, vol. 4, no. 6, pp. 1, 2004.

[18] A. Hickey, "A. Why packets per second (PPS) matter in DDoS defense," 2017. [Online]. Available: https://www.a10networks.com/blog/why-packets-per-second-pps-matter-in-ddos/

[19] Y. D. Lin, C. N. Lu, Y. C. Lai, W. H. Peng and P. C. Lin, "Application classification using packet size distribution and port association," *Journal of Network and Computer Applications*, vol. 32, no. 5, pp. 1023–1030, 2009.

[20] I. Khider, S. Elfaki, M. Elhassan and M. Siddig, "Evaluate the performance of internet protocol television," in *Int. Conf. on Electrical and Control Engineering*, Yichang, China, vol. 1, pp. 5902–5905, 2011.

[21] B. Claise, "Cisco systems netflow services export version 9," 2004. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3954

[22] A. H. Lashkari, "CICFlowmeter-v4. 0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyser for anomaly detection," 2018. [Online]. Available: https://github.com/ahlashkari/CICFlowMeter

[23] Z. Aouini and A. Pekar, "NFStream: A flexible network data analysis framework," *Computer Networks*, vol. 204, 108719, 2022.

[24] B. B. Zarpelão, R. S. Miani, C. T. Kawakani and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[25] C. M. Inacio and B. Trammell, "Yaf: Yet another flowmeter," in *LISA: 24th Large Installation System Administration Conf.*, San Jose, California, USA, vol. 10, 107, 2010.

[26] P. Lucente, "Pmacct: Steps forward interface counters," 2008. [Online]. Available: http://www.pmacct.net/pmacct-stepsforward.pdf
[27] D. Moore, K. Keys, R. Koga, E. Lagache and K. C. Claffy, "The CoralReef software suite as a tool for system and network administrators," in *LISA*, San Diego, California, USA, vol. 1, pp. 133–144, 2001.
[28] E. J. Jackson, M. Walls, A. Panda, R. Pettit, B. Pfaff *et al.,* "Softflow: A middlebox architecture for open vswitch," in *Annual Technical Conf.*, Denver, CO, USA, vol. 1, pp. 15–28, 2016.
[29] L. Deri and N. SpA, "nProbe: An open source netflow probe for gigabit networks," in *TERENA Networking Conf.*, Zagreb, Croatia, vol. 1, pp. 1–4, 2003.
[30] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino *et al.,* "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.
[31] Z. Liu, N. Thapa, A. Shaver, K. Roy, M. Siddula *et al.,* "Using embedded feature selection and CNN for classification on CCD-INID-v1–A New IoT dataset," *Sensors*, vol. 21, no. 14, 4834, 2021.
[32] P. S. Muhuri, P. Chatterjee, X. Yuan, K. Roy and A. Esterline, "Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks," *Information*, vol. 11, no. 5, 243, 2020.
[33] V. Jakkula, "Tutorial on support vector machine (svm)," *School of EECS, Washington State University*, vol. 37, no. 2.5, 3, 2006.
[34] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen *et al.,* "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3149–3157, 2017.
[35] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM–a tutorial into long short-term memory recurrent neural networks," arXiv:1909.09586, 2019.
[36] G. Apruzzese, L. Pajola and M. Conti, "The cross-evaluation of machine learning-based network intrusion detection systems," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 5152–5169, 2022.

Appendix A

Feature list generated by MBB-IoT

Features	Data type	Description
id	Int	Flow identifier
expiration_id	Int	Identifier of flow expiration trigger. Can be 0 for idle_timeout, 1 for active_timeout or −1 for custom expiration.
src_ip	Str	Source IP address string representation.
src_mac	Str	Source MAC address string representation.
src_oui	Str	Source organizationally unique identifier string representation.
src_port	Int	Transport layer source port.
dst_ip	Str	Destination IP address string representation.
dst_mac	Str	Destination MAC address string representation.
dst_oui	Str	Destination organizationally unique identifier string representation.
dst_port	Int	Transport layer destination port.
protocol	Int	Transport layer protocol.
ip_version	Int	IP version.
vlan_id	Int	Virtual LAN identifier.

(Continued)

(continued)

Features	Data type	Description
bidirectional_first_seen_ms	Int	Timestamp in milliseconds on first flow bidirectional packet.
bidirectional_last_seen_ms	Int	Timestamp in milliseconds on last flow bidirectional packet.
bidirectional_duration_ms	Int	Flow bidirectional duration in milliseconds.
bidirectional_packets	Int	Flow bidirectional packets accumulator.
bidirectional_bytes	Int	Flow bidirectional bytes accumulator (depends on accounting_mode).
src2dst_first_seen_ms	Int	Timestamp in milliseconds on first flow src2dst packet.
src2dst_last_seen_ms	Int	Timestamp in milliseconds on last flow src2dst packet.
src2dst_duration_ms	Int	Flow src2dst duration in milliseconds.
src2dst_packets	Int	Flow src2dst packets accumulator.
src2dst_bytes	Int	Flow src2dst bytes accumulator (depends on accounting_mode).
dst2src_first_seen_ms	Int	Timestamp in milliseconds on first flow dst2src packet.
dst2src_last_seen_ms	Int	Timestamp in milliseconds on last flow dst2src packet.
dst2src_duration_ms	Int	Flow dst2src duration in milliseconds.
dst2src_packets	Int	Flow dst2src packets accumulator.
dst2src_bytes	Int	Flow dst2src bytes accumulator (depends on accounting_mode).
tunnel_id	Int	Tunnel identifier (O: No Tunnel, 1: GTP, 2: CAPWAP, 3: TZSP).
application_name	Str	nDPI detected application name.
application_category_name	Str	nDPI detected application category name.
application_is_guessed	Int	Indicates if detection result is based on pure dissection or on a guess heuristics.
application_confidence	Int	Indicates the underlying detection method (O: Unknown classification, 1: Classification obtained looking only at the L4 ports, 3: Classification results based on partial/incomplete DPI information, 4: Classification results based on some LRU cache with partial/incomplete DPI information, 5: Classification results based on some LRU cache (i.e., correlation among sessions), 6: Deep packet inspection).
requested_server_name	Str	Requested server name (SSL/TLS, DNS, HTTP).

(Continued)

(continued)

Features	Data type	Description
client_fingerprint	Str	Client fingerprint (DHCP fingerprint for DHCP, JA3 for SSL/TLS and HASSH for SSH).
server_fingerprint	Str	Server fingerprint (JA3 for SSL/TLS and HASSH for SSH).
user_agent	Str	Extracted user agent for HTTP or User Agent Identifier for QUIC.
content_type	Str	Extracted HTTP content type.
bidirectional_min_ps	Int	Flow bidirectional minimum packet size (depends on accounting_mode).
bidirectional_mean_ps	Float	Flow bidirectional mean packet size (depends on accounting_mode).
bidirectional_stddev_ps	Float	Flow bidirectional packet size sample standard deviation (depends on accounting_mode).
bidirectional_max_ps	Int	Flow bidirectional maximum packet size (depends on accounting_mode).
src2dst_min_ps	Int	Flow src2dst minimum packet size (depends on accounting_mode).
src2dst_mean_ps	Float	Flow src2dst mean packet size (depends on accounting_mode).
src2dst_stddev_ps	Float	Flow src2dst packet size sample standard deviation (depends on accounting_mode).
src2dst_max_ps	Int	Flow src2dst maximum packet size (depends on accounting_mode).
dst2src_min_ps	Int	Flow dst2src minimum packet size (depends on accounting_mode).
dst2src_mean_ps	Float	Flow dst2src mean packet size (depends on accounting_mode).
dst2src_stddev_ps	Float	Flow dst2src packet size sample standard deviation (depends on accounting_mode).
dst2src_max_ps	Int	Flow dst2src maximum packet size (depends on accounting_mode).
bidirectional_min_piat_ms	Int	Flow bidirectional minimum packet inter arrival time.
bidirectional_mean_piat_ms	Float	Flow bidirectional mean packet inter arrival time.
bidirectional_stddev_piat_ms	Float	Flow bidirectional packet inter arrival time sample standard deviation.
bidirectional_max_piat_ms	Int	Flow bidirectional maximum packet inter arrival time.
src2dst_min_piat_ms	Int	Flow src2dst minimum packet inter arrival time.
src2dst_mean_piat_ms	Float	Flow src2dst mean packet inter arrival time.

(Continued)

(continued)

Features	Data type	Description
src2dst_stddev_piat_ms	Float	Flow src2dst packet inter arrival time sample standard deviation.
src2dst_max_piat_ms	Int	Flow src2dst maximum packet inter arrival time.
dst2src_min_piat_ms	Int	Flow dst2src minimum packet inter arrival time.
dst2src_mean_piat_ms	Float	Flow dst2src mean packet inter arrival time.
dst2src_stddev_piat_ms	Float	Flow dst2src packet inter arrival time sample standard deviation.
dst2src_max_piat_ms	Int	Flow dst2src maximum packet inter arrival time.
bidirectional_syn_packets	Int	Flow bidirectional syn packet accumulators.
bidirectional_cwr_packets	Int	Flow bidirectional cwr packet accumulators.
bidirectional_ece_packets	Int	Flow bidirectional ece packet accumulators.
bidirectional_urg_packets	Int	Flow bidirectional urg packet accumulators.
bidirectional_ack_packets	Int	Flow bidirectional ack packet accumulators.
bidirectional_psh_packets	Int	Flow bidirectional psh packet accumulators.
bidirectional_rst_packets	Int	Flow bidirectional rst packet accumulators.
bidirectional_fin_packets	Int	Flow bidirectional fin packet accumulators.
src2dst_syn_packets	Int	Flow src2dst syn packet accumulators.
src2dst_cwr_packets	Int	Flow src2dst cwr packet accumulators.
src2dst_ece_packets	Int	Flow src2dst ece packet accumulators.
src2dst_urg_packets	Int	Flow src2dst urg packet accumulators.
src2dst_ack_packets	Int	Flow src2dst ack packet accumulators.
src2dst_psh_packets	Int	Flow src2dst psh packet accumulators.
src2dst_rst_packets	Int	Flow src2dst rst packet accumulators.
src2dst_fin_packets	Int	Flow src2dst fin packet accumulators.
dst2src_syn_packets	Int	Flow dst2src syn packet accumulators.
dst2src_cwr_packets	Int	Flow dst2src cwr packet accumulators.
dst2src_ece_packets	Int	Flow dst2src ece packet accumulators.
dst2src_urg_packets	Int	Flow dst2src urg packet accumulators.
dst2src_ack_packets	Int	Flow dst2src ack packet accumulators.
dst2src_psh_packets	Int	Flow dst2src psh packet accumulators.
dst2src_rst_packets	Int	Flow dst2src rst packet accumulators.
dst2src_fin_packets	Int	Flow dst2src fin packet accumulators.

Appendix B

List of features used in the IoT-23 dataset

Features	Data Type	Description
ts	Float	This is the time of the first packet
uid	String	A unique identifier of the connection
id.orig_h	String	The originator’s IP address

(Continued)

(continued)

Features	Data Type	Description
id.orig_p	Integer	The originator's port number
id.resp_h	String	The responder's IP address
id.resp_p	Integer	The responder's port number
proto	Category	The transport layer protocol of the connection
service	String	An identification of an application protocol being sent over the connection
duration	Float	How long the connection lasted
orig_bytes	Float	The number of payload bytes the originator sent. For TCP this is taken from sequence numbers and might be inaccurate
resp_bytes	Float	The number of payload bytes the responder sent
conn_state	Category	Connection state
local_orig	Bool	If the connection is originated locally, this value will be T
local_resp	Bool	If the connection is responded to locally, this value will be T
missed_bytes	Integer	Indicates the number of bytes missed in content gaps, which is representative of packet loss
history	Category	Records the state history of connections as a string of letters
orig_pkts	Integer	Number of packets that the originator sent
orig_ip_bytes	Integer	Number of IP level bytes that the originator sent
resp_pkts	Integer	Number of packets that the responder sent
resp_ip_bytes	Integer	Number of IP level bytes that the responder sent
label	Category	Attack types