


# Lightweight Malicious Packet Classifier for IoT Networks



Seyedsina Nabavirazavi , S. S. Iyengar, and Naveen Kumar Chaudhary

**Abstract** Although the Internet of Things (IoT) devices simplify and automate everyday tasks, they also introduce a tremendous amount of security flaws. The current insufficient security measures for smart device protection make IoT devices a potential victim of breaking into a secure infrastructure. This research proposes an on-the-fly intrusion detection system (IDS) that applies machine learning (ML) to detect network-based cyber-attacks on IoT networks. A lightweight ML model is trained on network traffic to defer benign packets from normal ones. The goal is to demonstrate that lightweight machine learning models such as decision trees (in contrast with deep neural networks) are applicable for intrusion detection achieving high accuracy. As this model is lightweight, it could be easily employed in IoT networks to classify packets on-the-fly, after training and evaluation. We compare our lightweight model with a more complex one and demonstrate that it could be as accurate.

**Keywords** Internet of things · IoT · Networks security · Machine learning · Fault detection · Decision trees · Neural networks

## 1 Introduction

The popularity of the Internet of Things (IoT) has significantly increased. The forecast for the total number of connected IoT devices in 2025 is 27.1 billion. Today, it seems inevitable having a smart device in our homes. The proliferation of smart devices is not only within the domestic environment but it is also the driving force behind the development of an interconnected knowledge-based world; economy, society, and machinery of government. However, IoT devices come with a tremendous amount of

---

S. Nabavirazavi (✉) · S. S. Iyengar  
Florida International University, Miami, FL 33199, USA  
e-mail: [sina.nabavi16@gmail.com](mailto:sina.nabavi16@gmail.com)

N. K. Chaudhary  
National Forensic Sciences University, Gandhinagar, Gujarat, India

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2024  
S. J. Patel et al. (eds.), *Information Security, Privacy and Digital Forensics*,  
Lecture Notes in Electrical Engineering 1075,  
[https://doi.org/10.1007/978-981-99-5091-1\\_11](https://doi.org/10.1007/978-981-99-5091-1_11)

139

security risks [3]. Synopsys has a blog post that reveals, back in 2017, a lack of confidence in the security of medical devices with 67% of manufacturers. They believed an attack on a device is probable within a year, and only 17% of manufacturers take steps to prevent them.

The protocols designed for the IoT protocol stack are different from those of the IP stack. IoT designers have to choose between WiFi, Bluetooth, ZigBee, Z-Wave, and LoRaWan as their network protocol. IoT devices are power-constrained and have specific functionality. Using general-purpose protocols for every IoT device would result in more battery consumption and less quality of service. Hence, there is no standard approach for handling security issues in IoT, such as IPSec or SSL for the Internet.

The insufficient security measures and lack of dedicated anomaly detection systems for these heterogeneous networks make them vulnerable to a range of attacks such as data leakage, spoofing, denial of service (DoS/DDoS), etc. These can lead to disastrous effects, damaging the hardware, unavailability of the system, and compromising sensitive data privacy. For example, a deauthentication attack performed on a device with critical significance, such as a steering wheel in a wireless car, can pose a threat to human life. There is a gap between security requirements and the security capabilities of currently available IoT devices.

The traditional IT security ecosystem consists of static network defenses (firewalls, IDS), the ubiquitous use of end-point defenses (e.g., anti-virus), and software patches from vendors. We can't either employ these mechanisms due to the heterogeneity in devices and their use cases. This means that traditional approaches for discovering attack signatures (e.g., honeypots) will be insufficient or non-scalable [13].

Traditional anomaly detection systems are also ineffective within IoT ecosystems since the range of possible normal behaviors of devices is significantly larger and more dynamic than in traditional IT environments. IDSs such as SNORT and Bro only work on traditional IP-only networks as they are static and use signature-based techniques [13]. To address this issue, multiple intrusion detection systems have been introduced in the context of IoT networks. Yet, the majority of them focus on detecting a limited set of attacks, in particular, routing attacks and DoS. However, there are IDSs that introduce dynamicity by employing machine learning to detect malicious behavior. Soltani et al. focused on 3 deep learning algorithms and applied them to the context of intrusion detection [12]. They also introduced a new deep learning-based classifier. Amouri et al. employed supervised machine learning for IDS [1, 5]. Shukla et al. proposed an IDS that uses a combination of machine learning algorithms, such as K-means and decision trees to detect wormhole attacks on 6LoWPAN networks [11]. Cao et al. proposed a machine learning intrusion detection system for industrial control systems [6]. Bangui et al. have employed random forests for IDS in vehicular ad hoc networks. The aim of this paper is similar. Yet, its focal point is to design the machine learning IDS as lightweight and efficient as possible. When the IDS is lightweight and effective, it may be embedded within certain IoT devices. We summarize the contributions of our work below.

- We analyze a public dataset for malicious IoT packet classification.
- We convert the raw traffic of the captured data (PCAP) to a comprehensible format for training. (Lightweight models are unable to classify malicious packets by inspecting raw bytes)
- We develop a Python code to instantiate, train, and test the specified machine-learning models.
- We document and report the evaluation metrics for these models and compare them.

The rest of the paper follows this outline. Section 2, discusses the phase of data collection and how the collected data is comprehended to train the machine learning models. For both training and testing the models, the open-source Aposemat IoT-23 dataset [7] will be used. Later, in Sect. 3, we provide clarification on which machine learning models we use, which features are selected, and how the data is labeled. Section 4 provides the results and the evaluation of these models. In this section, the performance of lightweight models is compared with a more complicated model, a neural network. Section 5 concludes the project's paper and Sect. 6 opens the door for future work.

## 2 Data Collection

### 2.1 Dataset

We use the public dataset of **Aposemat IoT-23** for our models [7]. The packets are grouped into different chapters (scenarios). The dataset contains 20 captures of malware traffic in the IoT network and 3 captures of benign traffic. The dataset contains more than 760 million packets and 325 million labeled flows with more than 500h of traffic. The IoT-23 dataset consists of 23 captures overall, called scenarios, of different IoT network traffic. We summarize this information in Figs. 1 and 2.

The malicious scenarios were created executing a specific malware in a Raspberry Pi. In the dataset, the researchers have included traffic from Mirai, Torii, Hide and Seek, and Hajime attacks. The network traffic capture for the benign scenarios was obtained by capturing the network traffic of three different IoT devices: a Philips HUE smart LED lamp, a Somfy Smart Door Lock, and an Amazon Echo home intelligent personal assistant. We should mention that these three IoT devices are actual devices and not simulated. Both malicious and benign scenarios run in a controlled network environment with an unrestrained internet connection like any other real IoT device.

#	Name of Dataset	Duration (hrs)	#Packets	#ZeekFlows	Pcap Size	Name
1	CTU-IoT-Malware-Capture-34-1	24	233,000	23,146	121 MB	Mirai
2	CTU-IoT-Malware-Capture-43-1	1	82,000,000	67,321,810	6 GB	Mirai
3	CTU-IoT-Malware-Capture-44-1	2	1,309,000	238	1.7 GB	Mirai
4	CTU-IoT-Malware-Capture-49-1	8	18,000,000	5,410,562	1.3 GB	Mirai
5	CTU-IoT-Malware-Capture-52-1	24	64,000,000	19,781,379	4.6 GB	Mirai
6	CTU-IoT-Malware-Capture-20-1	24	50,000	3,210	3.9 MB	Torii
7	CTU-IoT-Malware-Capture-21-1	24	50,000	3,287	3.9 MB	Torii
8	CTU-IoT-Malware-Capture-42-1	8	24,000	4,427	2.8 MB	Trojan
9	CTU-IoT-Malware-Capture-60-1	24	271,000,000	3,581,029	21 GB	Gagfyt
10	CTU-IoT-Malware-Capture-17-1	24	109,000,000	54,659,864	7.8 GB	Kenjiro
11	CTU-IoT-Malware-Capture-36-1	24	13,000,000	13,645,107	992 MB	Okiru
12	CTU-IoT-Malware-Capture-33-1	24	54,000,000	54,454,592	3.9 GB	Kenjiro
13	CTU-IoT-Malware-Capture-8-1	24	23,000	10,404	2.1 MB	Hakai
14	CTU-IoT-Malware-Capture-35-1	24	46,000,000	10,447,796	3.6G	Mirai
15	CTU-IoT-Malware-Capture-48-1	24	13,000,000	3,394,347	1.2G	Mirai
16	CTU-IoT-Malware-Capture-39-1	7	73,000,000	73,568,982	5.3GB	IRCBot
17	CTU-IoT-Malware-Capture-7-1	24	11,000,000	11,454,723	897 MB	Linux,Mirai
18	CTU-IoT-Malware-Capture-9-1	24	6,437,000	6,378,294	472 MB	Linux.Hajime
19	CTU-IoT-Malware-Capture-3-1	36	496,000	156,104	56 MB	Muhstik
20	CTU-IoT-Malware-Capture-1-1	112	1,686,000	1,008,749	140 MB	Hide and Seek

Fig. 1 Summary of the malicious IoT scenarios

#	Name of Dataset	Duration(-hrs)	#Packets	#ZeekFlows	Pcap Size	Device
1	CTU-Honeypot-Capture-7-1 (somfy-01)	1.4	8,276	139	2,094 KB	Somfy Door Lock
2	CTU-Honeypot-Capture-4-1	24	21,000	461	4,594 KB	Philips HUE
3	CTU-Honeypot-Capture-5-1	5.4	398,000	1,383	381 MB	Amazon Echo

Fig. 2 Summary of the benign IoT scenarios

## 2.2 Data Preparation

Our lightweight machine learning model would not be able to classify raw bytes of network traffic. Hence, we convert the raw PCAP files into Packet Description Markup Language (PDML) format. PDML conforms to the XML standard and contains details about the packet layers. We then simply represent the PDML files in Comma-separated Values (CSV) by only selecting our desired features from each PDML packet. We have implemented this in Python.

### 2.3 Feature Selection

As the feature space is relatively large (Table 4), all packet features may not be relevant. We have manually selected 13 features that appeared to have the highest correlation based on Eirini's research [2]. We state the name of the features below.

**length, caplen, frame-encapType, frame-timeShift, ip-flags, ip-flagsMF, ip-flagsDF, ip-ttl, ip-fragOffset, tcp-flagsAck, tcp-flagsSyn, tcp-flagsPush, icmp-code**

When a feature is missing in a packet, for example, the `tcp.flag` in a UDP packet, we replace the non-existing value, *None*, with  $-1$ . The idea is to use a unique value for missing features (i.e., not used by existing features) so the model learns the impact of a missing feature as well.

Sarhan et al. focused on feature extraction in machine learning-based IDS more deeply and have concluded that the choice of datasets significantly alters the performance of feature extraction techniques [10].

### 2.4 Sample Size Reduction

According to the scale of our paper, we have selected one malicious and one normal scenario to train and test our models. The benign scenario contains **75356** packets, whereas the malicious scenario contains **83068**.

## 3 Machine Learning Models

The Python library we use for employing machine learning is **scikit-learn**. We use decision trees as our lightweight model [4, 8, 9]. Decision Trees are a supervised learning non-parametric method for classification and regression problems. Their purpose is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

A problem to address when training a machine learning model is overfitting. The model, in order to keep the accuracy as high as possible, may overfit the dataset and lose its ability to generalize. We use validation curves to measure the overfitting of our model. We evaluate this metric with respect to the depth of the decision tree. We can see in Fig. 3 that when the depth excels 10, the model starts to overfit. Hence, we keep the hyperparameter of `max_depth` less than or equal to 10. There are other ways of dealing with overfitting in decision trees, such as ensemble techniques and pruning.

After finding the maximum possible depth for our tree (10), we then instantiate a *DecisionTreeClassifier* class and train it on the dataset. We set the maximum depth

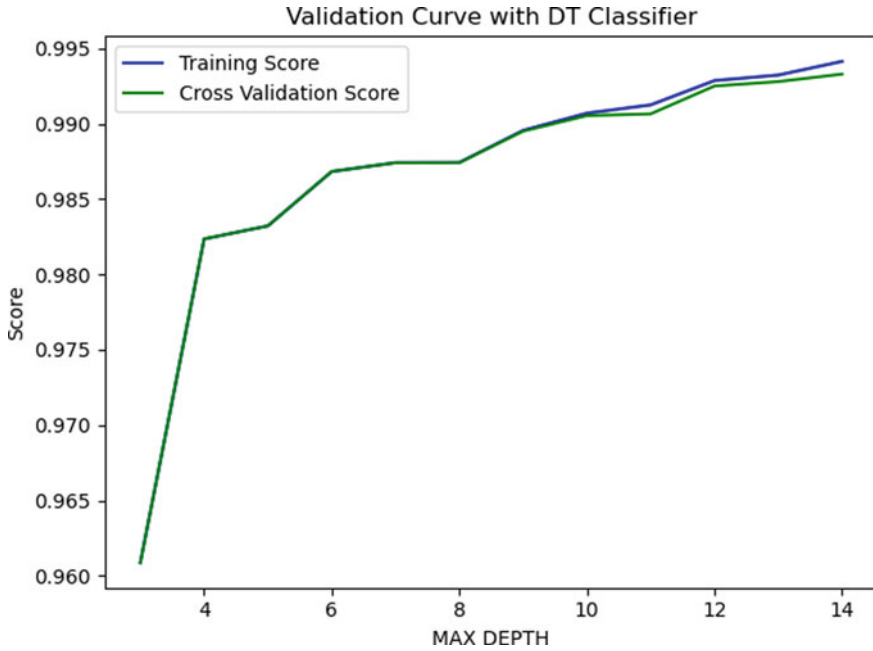


Fig. 3 Validation score with respect to the tree depth

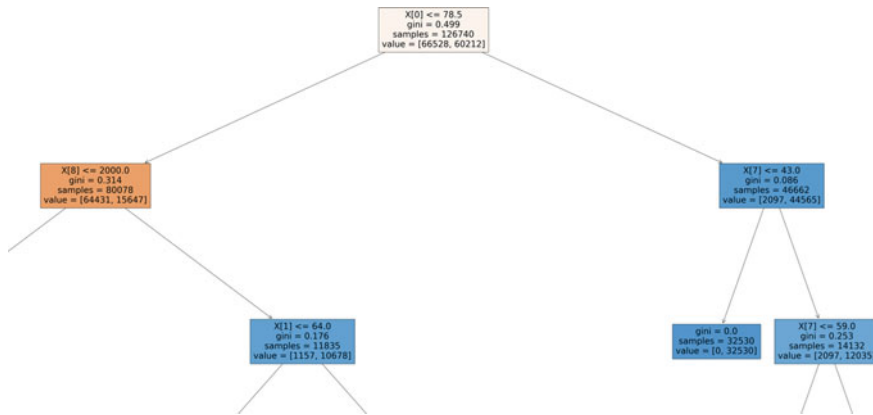


Fig. 4 A subtree of the final decision tree

of the decision tree to 10. To be able to see the decision rules and the number of samples satisfying them more clearly, we can refer to Fig. 4 which depicts a subtree of the final model.

Figure 5 illustrates a bigger subtree of our model. We can see in the subtree that the feature *len* ( $X[0]$ ) appears in many logical rules and is an essential parameter for our decision tree.

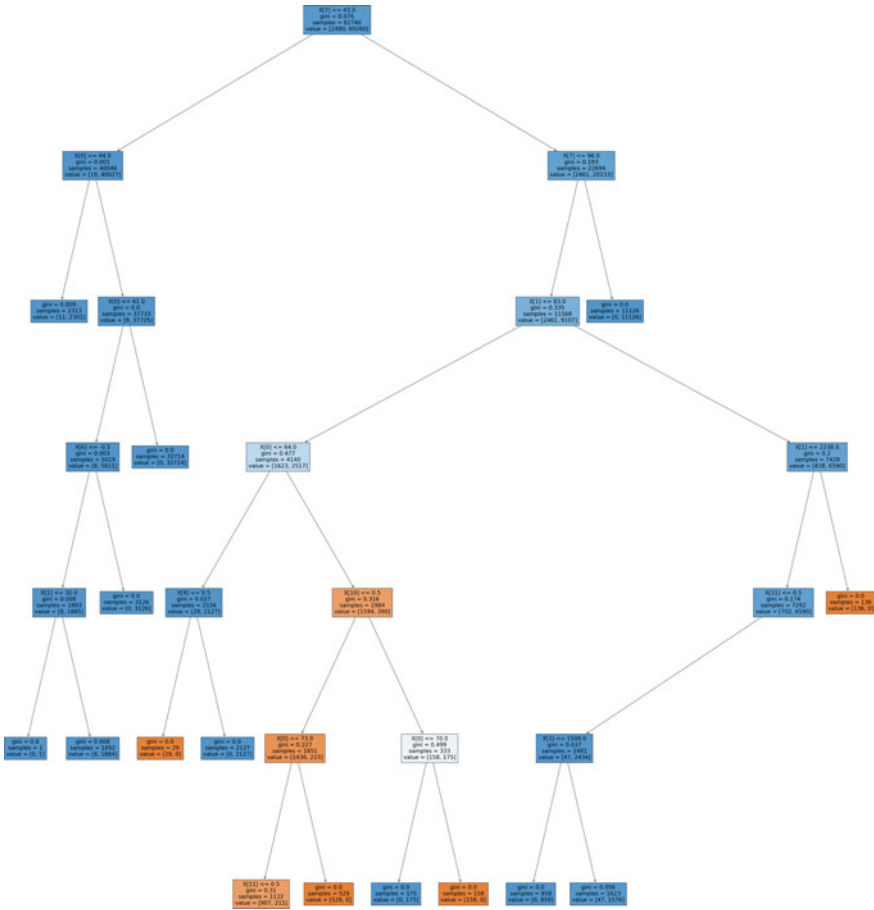


Fig. 5 Detailed subtree

### 4 Evaluation

At first, we describe the evaluation metrics of our decision tree in Table 1. To show that our lightweight model’s performance is satisfactory in this use case, we develop a neural network using the same library and compare its metrics with those of the decision tree. Table 2 briefly describes our neural network.

The neural network is **96.6%** accurate, which is even less than our decision tree. With that in mind that we have avoided overfitting, we can claim that our model outperforms classical neural networks in this scenario. It is important to note that the deep learning approaches may reach higher accuracy (99% and above), especially when the task includes the detection of the attack type [12] (Table 3).

**Table 1** Decision tree evaluation metrics

Metric	Result
Accuracy	0.993
True positive (correctly classified as malicious)	16400 packets
False positive	58 packets
True negative	15087 packets
False negative	142 packets

**Table 2** Neural network parameters

Parameter	Value
Solver	lbfgs
Hidden layers	4
Hidden layer dimensions	(5, 5, 5, 2)
Activation function	tanh

**Table 3** Neural network evaluation metrics

Metric	Result
Accuracy	0.966
True positive (correctly classified as malicious)	15857 packets
False positive	219 packets
True negative	14771 packets
False negative	839 packets

## 5 Conclusion

In this paper, we presented an on-the-fly malicious packet classifier. It employs decision trees to capture IoT network packets and label them as normal or malicious. We demonstrated that our lightweight model outperforms complex neural networks while keeping the processing and storage requirements at a minimum.

## 6 Future Work

For future work, we may integrate this model with a packet capturer to automatically label all (or some) of the traffic in the network.

We selected our features manually. In future research, one might also automate feature selection using statistical or ML methods (intrinsic, wrapper methods, etc.)



One possible contribution to this research would be attack classification. The group of malicious packets found by the decision tree can be fed to a more complex machine learning model to detect the type of attack happening in the network. We may use several IoT attack categories for classification.

- **Denial of Service (DoS):** aims to make IoT devices unavailable by overloading the network and disrupting the services.
- **Distributed Denial of Service (DDoS)/Botnets:** an adversary compromises many IoT devices to employ a significant DoS.
- **Spoofing:** The attacker tries to manipulate an authenticated identity by forging.
- **Man-In-The-Middle:** The communication channel is compromised. The attacker can act after this attack as a proxy to read, write, and modify the packets.
- **Insecure Firmware:** After the control over an IoT device is gained, the device is used to attack other devices.
- **Data Leakage:** If the data is not encrypted, the privacy of the user data is compromised and may be used by an attacker to access the private network.
- **Botnets:** An adversary controls a network of connected IoT devices by which he performs malicious actions.
- **Brute Force Password Attacks:** A potential attacker can gather substantial processing power to try every possible secret a device possesses.

**Acknowledgements** Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-21-1-0264. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

## Appendix

### A Full List of Features

The following table includes all the features that we collected, from which 13 were selected (Table 4).

**Table 4** Appendix: feature list

len	icmp.respin
caplen	icmp.respto
frame.encaptype	data.len
frame.offsetshift	ssl.record.content-type
frame.len	ssl.record.version
frame.cap-len	ssl.record.length
frame.marked	arp.hw.type
frame.ignored	arp.proto.type
eth.lg	arp.hw.size
eth.ig	arp.proto.size
ip.version	arp.opcode
ip.hdr-len	http.response.code
ip.dsfield.dscpl	http.content-length
ip.dsfield.enc	http.response
ip.src	http.response-number
ip.dst	http.request
ip.len	http.request-number
ip.flags	classicstun.type
ip.flags.rb	classicstun.length
ip.flags.df	udp.srcport
ip.flags.mf	udp.dstport
ip.frag-offset	udp.length
ip.ttl	udp.chksum.status
ip.proto	udp.stream
ip.checksum.status	dns.flags.response
tcp.srcport	dns.flags.opcode
tcp.dstport	dns.flags.truncated
tcp.stream	dns.flags.recdesired
tcp.len	dns.flags.z
tcp.seq	dns.flags.checkdisable
tcp.nxtseq	dns.flags.rcode
tcp.ack	dns.flags.queries
tcp.hdr-len	dns.count.answers
tcp.flags.res	dns.count.authr
tcp.flags.ns	dns.qry.name.len
tcp.flags.cwr	dns.count.labels

(continued)

**Table 4** (continued)

len	icmp.resp.in
tcp.flags.ecn	dns.resp.type
tcp.flags.urg	dns.resp.class
tcp.flags.ack	dns.resp.ttl
tcp.flags.push	dns.resp.len
tcp.flags.reset	igmp.version
tcp.flags.syn	igmp.type
tcp.flags.fin	igmp.max-resp
tcp.window-size-value	igmp.checksum.status
tcp.window-size	ntp.flags.li
tcp.window-size-scale-factor	ntp.flags.vn
tcp.checksum.status	ntp.flags.mode
tcp.urgent-pointer	ntp.startum
tcp.options.nop	ntp.ppoll
tcp.options.mss-val	ntp.root-delay
tcp.options.sack-perm	ntp.rootdispersion
tcp.analysis.bytes-in-flight	ntp.precision
tcp.analysis.push-bytes-sent	bootp.type
tcp.payload	bootp.hw.type
icmp.type	bootp.hw.len
icmp.code	bootp.hops
icmp.ident	bootp.secs

## References

1. Amouri A, Alaparthi VT, Morgera SD (2018) Cross layer-based intrusion detection based on network behavior for IoT. In: 2018 IEEE 19th wireless and microwave technology conference (WAMICON), pp 1–4
2. Anthi E, Williams L, Słowińska M, Theodorakopoulos G, Burnap P (2019) A supervised intrusion detection system for smart home IoT devices. *IEEE Internet Things J* 6(5):9042–9053. <https://doi.org/10.1109/JIOT.2019.2926365>
3. Anthi E, Williams L, Burnap P (2018) Pulse: an adaptive intrusion detection for the internet of things. In: *Living in the internet of things: cybersecurity of the IoT*, pp 1–4. <https://doi.org/10.1049/cp.2018.0035>
4. Bilge L, Kirda E, Kruegel C, Balduzzi M (2011) Exposure: finding malicious domains using passive DNS analysis
5. Buczak AL, Guven E (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun Surv Tutor* 18(2):1153–1176. <https://doi.org/10.1109/COMST.2015.2494502>
6. Cao Y, Zhang L, Zhao X, Jin K, Chen Z (2022) An intrusion detection method for industrial control system based on machine learning. *Information* 13(7):322
7. Garcia S, Parmisano A, Erquiaga MJ (2020) Iot\_23: a labeled dataset with malicious and benign IoT network traffic. <https://doi.org/10.5281/zenodo.4743746>

8. Kruegel C, Toth T (2003) Using decision trees to improve signature-based intrusion detection. In: Proceedings of the 6th International workshop on the recent advances in intrusion detection (RAID'2003), LNCS vol 2820. Springer Verlag, pp 173–191
9. Salzberg SL (1994) C4.5: programs for machine learning by J. Ross Quinlan, Morgan Kaufmann Publishers, Inc. 1993. Mach Learn 16(3):235–240. <https://doi.org/10.1007/BF00993309>
10. Sarhan M, Layeghy S, Moustafa N, Gallagher M, Portmann M (2022) Feature extraction for machine learning-based intrusion detection in IoT networks. Digital Commun Netw
11. Shukla P (2017) MI-ids: a machine learning approach to detect wormhole attacks in internet of things. In: 2017 Intelligent systems conference (IntelliSys) pp 234–240
12. Soltani M, Ousat B, Siavoshani MJ, Jahangir AH (2021) An adaptable deep learning-based intrusion detection system to zero-day attacks. arXiv preprint [arXiv:2108.09199](https://arxiv.org/abs/2108.09199)
13. Yu T, Sekar V, Seshan S, Agarwal Y, Xu C (2015) Handling a trillion (unfixable) flaws on a billion devices: rethinking network security for the internet-of-things. In: Proceedings of HotNets, 5p. Philadelphia, PA