IoTDSCreator: A Framework to Create Labeled Datasets for IoT Intrusion Detection Systems

Hyunwoo Lee
Korea Institute of Energy Technology
Department of Engineering
Naju, Jeonnam, Republic of Korea
hwlee@kentech.ac.kr

Charalampos Katsis
Purdue University
Department of Computer Science
West Lafayette, IN, United States
ckatsis@purdue.edu

Alireza Lotfi
Purdue University
Department of Computer Science
West Lafayette, IN, United States
lotfia@purdue.edu

Taejun Choi
Korea Institute of Energy Technology
Department of Engineering
Naju, Jeonnam, Republic of Korea
wns7731@kentech.ac.kr

Soeun Kim
Korea Institute of Energy Technology
Department of Engineering
Naju, Jeonnam, Republic of Korea
kssony@kentech.ac.kr

Ashish Kundu
Cisco
Cisco Research
San Jose, CA, United States
ashkundu@cisco.com

Elisa Bertino
Purdue University
Department of Computer Science
West Lafayette, IN, United States
bertino@purdue.edu

Abstract

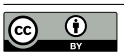
Intrusion detection systems (IDSes) are critical building blocks for securing Internet-of-Things (IoT) devices and networks. Advances in AI techniques are contributing to enhancing the efficiency of IDSes, but the performance of IDSes typically depends on high-quality training datasets. The scarcity of such datasets is a major concern for the effective use of machine learning for IDSes in IoT networks. To address such a need, we present IoTDSCreator– a tool for the automatic generation of labeled datasets able to support various devices, connectivity technologies, and attacks. IoTDSCreator provides a user with DC-API, an API by which the user can describe a target network and an attack scenario against it. Based on the description, the framework configures the network, leveraging virtualization techniques over user-provided physical machines, performs single or multi-step attacks, and finally returns labeled datasets. Thereby, IoTDSCreator dramatically reduces the manual efforts for generating labeled and diverse datasets. We release the source code of IoTDSCreator and 16 generated datasets with 193 features.

CCS Concepts

• **Security and privacy** → Network security; **Intrusion detection systems**.

Keywords

IoT testbed, labeled dataset creation, multi-step attack



CODASPY '25, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-1476-4/2025/06
https://doi.org/10.1145/3714393.3726004

ACM Reference Format:
Hyunwoo Lee, Charalampos Katsis, Alireza Lotfi, Taejun Choi, Soeun Kim, Ashish Kundu, and Elisa Bertino. 2025. IoTDSCreator: A Framework to Create Labeled Datasets for IoT Intrusion Detection Systems. In *Proceedings of the Fifteenth ACM Conference on Data and Application Security and Privacy (CODASPY '25), June 4–6, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3714393.3726004

1 Introduction

The explosive growth of Internet-of-Things (IoT) in many different domains, such as critical infrastructures, is making IoT security a primary requirement [4]. To secure IoT devices and systems, intrusion detection systems (IDSes) are introduced as an important building block [18, 20, 21]. However, the use of ML techniques needs building classifiers to correctly distinguish attack patterns from benign ones, which in turn requires large amounts of training data with high-quality labels [15]. Also, imbalanced, inaccurate, or unstable datasets can lead to significant performance degradation of a classifier (e.g., over-classification) [3]. Furthermore, to keep IDSes up to date, rapidly collecting/generating suitably labeled datasets is required but challenging.

Although there are several IoT datasets [6, 13, 25, 30], typically used for benchmarking, they have limitations. Prior work [22, 28] has shown drawbacks of existing datasets; that is, the datasets contain invalid features or replicated instances. Our focus is, however, on addressing the following drawbacks specific to IoT devices and networks. **(L1) Network specificity:** An IDS that shows high performance on a specific benchmarking dataset may not be effective in different environments as the IDS can be overfitted to a specific network. To train highly effective IDSes, one should use a large number of different datasets with a variety of attacks against the target network. **(L2) Application-based emulation:** Our analysis of existing datasets shows that most of them are generated from IoT applications running on conventional machines. Therefore, those

datasets do not represent the characteristics of IoT devices and networks well. **(L3) Lack of multi-step attacks:** No IoT benchmarking dataset contains data for multi-step attacks; however, actual attacks that have targeted IoT devices usually include multiple attack steps [1, 2, 24]. To capture the advanced attacks, it is necessary to generate datasets that contain data about multiple-step attacks.

To address such limitations, we propose IoTDSCreator- a tool for the automatic generations of labeled datasets, able to support various devices, communication techniques, and attacks. IoTDSCreator offers a user with the dataset creation application programming interface (DC-API) to describe a target network and an attack scenario over it. Based on the description, the framework configures the network, leveraging virtualization techniques over a physical machine provided by the user, performs single or multi-step attacks, and returns labeled datasets. Therefore, IoTDSCreator generates a dataset for a target network specified in the description, addressing (L1), and it leverages the Quick EMUlator (QEMU) to emulate a full system of IoT devices, addressing (L2). IoTDSCreator is able to include labels of attack steps, addressing (L3).

Contributions. We make the following contributions:

- We conduct a scenario analysis and a feature analysis to evaluate existing datasets of IoT networks and elicit requirements for high-quality IoT datasets.
- We design IoTDSCreator, a flexible emulation and simulation-based IoT security testbed that generates labeled datasets of multi-step attacks. We present the detailed design of IoTDSCreator with a running example.
- We implement IoTDSCreator and release it in a public repository[1]. In the current version of IoTDSCreator, we provide 26 types of IoT devices, 2 type of communication technique, and 193 features. Furthermore, we post 16 datasets, including the probing and flooding datasets.

2 Preliminary

2.1 Internet-of-Things

The Internet-of-Things (IoT) is the interconnection of heterogeneous networked objects and the integration between physical things and cyberspace by leveraging the Internet Protocol (IP). IoT aims to connect low-end embedded systems, such as appliances, to the Internet. Hence, IoT networks are different from conventional networks. First, many lightweight protocols, such as Constrained Application Protocol (CoAP) and Message Queuing Telemetry Transport (MQTT), or system architectures, have been designed to suit the low capabilities of IoT devices. Second, behavior patterns of IoT devices are simple, and communications are scarce as IoT devices usually run one or a few applications with specific tasks [21]. Therefore, it is challenging to collect large amounts of data [21].

2.2 IoT Attacks and Cyber Kill-Chain

IoT attacks. Due to their lack of capabilities, IoT devices often represent the weakest security components of networks [8]. Therefore, attack surfaces expand as the number of IoT devices significantly grows [7]. Botnets show a 25% increase in their activity in 2023 compared to the previous year [31], and there was a 500% increase in the number of IoT attacks in 2020 [17]. Recent attacks [1, 2, 24] are designed in a sophisticated manner to exploit vulnerabilities in IoT devices. These attacks often consist of multiple steps, possibly executed over long periods. As they are targeted and persistent, they are referred to as advanced persistent threats (APTs).

Cyber kill-chain. To systematically understand and protect against multi-step attacks, several cyber kill chain frameworks [11, 27] have been introduced to model the steps that an attacker has to execute in order to reach their goal (see Figure 1). In this model, an attacker typically moves forward to the next step only after completing the current step. As the steps in the cyber kill chain frameworks are executed progressively in one direction, detection of anomalies or other activities that can be correlated to a specific step of a multi-step attack can provide hints about possible next steps and previous steps of the attack. Such hints can be helpful to contain the attack [5] or to do threat hunting [26]. Hence, to provide detailed information about multi-step attacks, many APT reports or posts describe the mapping between the adversarial behavior of the attack and the corresponding cyber kill chain step (e.g., [29]).

Threat model. We consider external attackers that aim to exfiltrate confidential data, make a target network inaccessible to legitimate users, or build a botnet with victim machines in a target network. To achieve those goals, the attackers may launch multi-step attacks. We model our network as composed of a victim network and the Internet, like the testbed in [16]. The victim network may include high-end devices (e.g., general-purpose personal computers) and low-end devices (e.g., Raspberry Pi) connected by diverse types of communication links (e.g., Ethernet or WiFi).

3 IDSes Datasets Tailored for IoT

This section analyzes publicly available IDS datasets, eliciting requirements of high-quality IoT datasets. We examine well-known datasets that have been referenced more than 400 times during the last decade, from 2014 to 2023, including 4 IoT datasets and 3 non-IoT datasets. We especially focus on how the IoT datasets are generated compared to non-IoT datasets.

3.1 Network Configuration

The reason why we focus on the environments where the datasets are generated is to understand how researchers configure IoT datasets dealing with the diversity of IoT. We want to see how IoT networks are made. We especially check if and how virtualized techniques are used. To answer those questions, we check how the datasets model IoT devices, how the datasets generate IoT traffic, how many devices are included in the datasets, and what type of communication links are in the datasets.

Compared to non-IoT datasets, the authors of existing IoT datasets mainly focused on how to generate data about IoT services. One can generate such data by either simulating IoT traffic or capturing IoT packets from actual IoT devices. For instance, Bot-IoT and ToN-IoT rely on IoT services simulated with Node-red[2] on virtual machines. On the other hand, in IoTID20, the network packets are captured from a small testbed with real-world devices including a wireless

[1] https://github.com/iotdscreator

[2] https://nodered.org

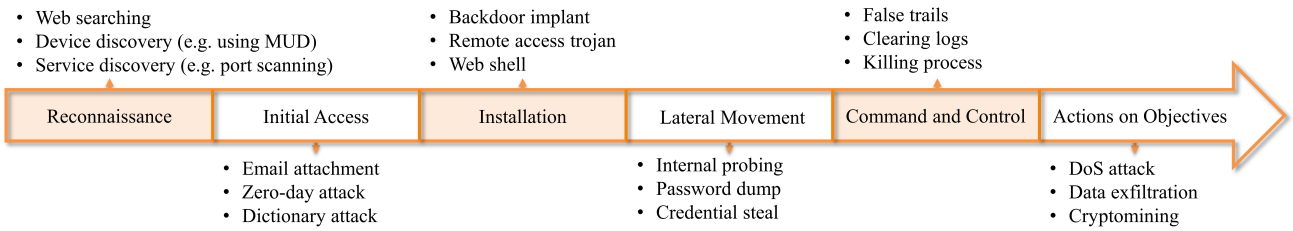


Figure 1: A cyber kill chain. An adversary should complete one step before moving forward to the next step.

Dataset	Referenced*	IoT Device Model	IoT Traffic	Number of Devices	Communication Links	Attack Types	Attack Traffic	Multi-step Attack	Instances	Network Data	Host Data	Period (Collection Time)	Features	Labels	Raw Data
		Network Configuration				Attack Scenario			Dataset Features						
IoT dataset															
Bot-IoT (2019) [13]	2800+	Sim.	Sim.	307	E	7	Sim.	N/A	73.3M	✓	N/A	35d	35	0/1, AC, SC	✓
IoTID20 (2020) [30]	600+	Actual	Actual	58K	U	8	Sim.	N/A	626K	✓	N/A	113d	86	0/1, AC, SC	✓ ([12])
IoT-23 (2020) [25]	700+	Actual	Actual	3	U	15	Actual	N/A	325M	✓	N/A	21d	23	0/1, AC	✓
ToN-IoT (2020) [6]	900+	Actual	Sim.	9	E	9	Sim.	N/A	22.3M	✓	N/A	27d	45	0/1, AC	✓
Non-IoT dataset															
DARPA2000 (2000) [14]	400+	–	–	60	U	4	Sim.	✓	33K	✓	N/A	3h	42	0/1, AC	✓
NSL-KDD (2009) [10]	4100+	–	–	11	U	22	Sim.	N/A	148K	✓	N/A	9w	42	0/1, AC, SC	✓
UNSW-NB15 (2015) [19]	5500+	–	–	45	E	9	Sim.	N/A	2.5M	✓	N/A	31h	49	0/1, AC	✓

Sim.: Simulated, U: Unspecified, E: Ethernet, 0/1: Normal/Abnormal, AC: Attack Categories, SC: Subcategories
* Based on Google Scholar

Table 1: Existing publicly available intrusion detection dataset

router, two smart home devices – SKT NUGU[3] and an EZVIZ WiFi camera[4], and IoT-23 contains packets from the actual IoT devices including a Philips HUE lamp, an Amazon Echo, and a Somfy smart doorlock. Note that traffic is captured at the dedicated testbed.

Note that both simulating IoT traffic and capturing IoT packets from actual IoT devices have their limitations. The Bot-IoT or ToN-IoT datasets, which rely on the former approach, are generated based on applications that simulate IoT networking patterns on general-purpose machines. Therefore, this approach cannot fully reflect patterns due to IoT-specific architectures such as ARM or communication links such as WiFi. On the other hand, the IoTID20 or IoT-23 datasets, which leverage the latter approach, can include such patterns as the datasets rely on traffic from actual IoT devices. However, this approach lacks flexibility because the IoT patterns only reflect hardware devices.

3.2 Attack Scenario

We analyze the attack types in the IoT datasets and see what kinds of IoT attacks are included in the datasets, how the patterns of attacks are generated, and if the datasets contain multi-step attacks. We find that the datasets commonly contain 8 types of attack, including DDoS, DoS, MITM, and Backdoor. Note that the attack types are also included in the non-IoT datasets. Therefore, it is important that the attack types be diversified and the datasets include IoT-specific attacks. Furthermore, we find that there is no IoT dataset that includes multi-step attacks. Only one non-IoT dataset, the DARPA2000 dataset, includes data related to multi-step attacks. Also, a lot of attack packets are generated based on applications that simulate specific attacks over high-end devices (e.g., a laptop).

[3]https://www.nugu.co.kr/
[4]https://www.ezviz.com/category/security-wifi-cameras

For instance, the attack traffic in Bot-IoT is generated from a Kali Linux machine. Furthermore, there is no IoT dataset that contains attack packets from a compromised IoT device or toward an IoT device. Instead, the attack packets are generated and captured from the high-end devices, and the packets are usually combined with the normal packets separately captured from IoT devices. Therefore, there is a gap with respect to realistic patterns if the networks where normal and attack packets are generated are different.

3.3 Dataset Features

We investigate what features the IoT datasets provide. In detail, we focus on the number of samples in the dataset, whether network data is provided or not (indicated by a flag in the table), whether host data is provided or not (indicated by a flag in the table), data collection period, the number of features in the dataset, the types of labels provided by the dataset, and whether raw data (e.g., pcap files) is provided or not (indicated by a flag in the table).

The IDS datasets consist of multiple features, such as the mean of the packets' inter-arrival time as a network feature or a list of open processes as a host feature. Although DARPA2000 [14] contains Windows NT event logs as host features, typically, both the IoT datasets and the non-IoT datasets provide only network features. We find that the time interval between the latest packet and the oldest packet spans from 21 days to 113 days for the IoT datasets (31 hours to 9 weeks for the non-IoT datasets). However, after a detailed analysis, we find that data are not continuously collected. The datasets are generated from multiple pcap files; the time required to collect the packets in each file ranges from 1 minute to 23 minutes. Then, they are combined into scenarios, and features are extracted. The number of network features ranges from 23 to 86. These features are generated based on dedicated tools such as Argus [23]. Finally, we also find that all the datasets release the raw data, such as pcap files, together with the datasets.

3.4 Takeaways

We observe the following three key conclusions:

- IoT datasets adopt two main approaches: traffic simulation (flexible but less realistic) and real device capture (realistic but less scalable). Both approaches show a trade-off between flexibility and reality.
- Existing datasets mainly rely on flow-based features calculated after sessions end, limiting their usefulness for real-time or online detection systems.
- Many datasets combine separately generated benign and attack traffic, often from different environments, which may distort natural traffic patterns and thus may not reflect real-world aspects.

4 The IoTDSCreator Framework

This section presents an overview of how IoTDSCreator generates a labeled dataset.

4.1 Input and Output

IoTDSCreator takes a scenario description and a list of available machines as input and outputs a labeled dataset.

Input. IoTDSCreator provides the dataset creation application programming interface (DC-API), in a YAML [9] format[5], to describe a threat scenario, including a multi-step attack, from an attacker's perspective. The resulting description is called a *scenario description* and describes the network (i.e., IoT devices, routers, and links), the applications running on the devices, the multi-step attack scenario, and the labels (i.e., attack steps).

Output. IoTDSCreator generates the following four types of datasets:

- **Packet dataset:** IoTDSCreator captures network packets from vantage points in the created virtual network. The packet dataset contains features, including the packet length and the transport protocol, about each packet.
- **Flow dataset:** Based on the captured packets, IoTDSCreator extracts features about the relationship between packets, such as the mean of inter-arrival time from a network window that may contain multiple packets within a given period called a window length.
- **Host dataset:** IoTDSCreator collects host logs from available IoT devices. The host dataset contains features, including CPU utilization or a list of open processes, about each device in the network.
- **Transition dataset:** Based on host logs, IoTDSCreator extracts features about transitions between host logs, such as CPU utilization transition rate from a host window that may contain multiple host logs within a given period called a window length.

Note that when a window length is the same as the total length of a scenario, the format of the generated flow dataset is similar to that of the existing datasets. By reducing the length of a window, we address the limitation of existing datasets, discussed in Section 3.

4.2 Workload Overview

With the scenario description, IoTDSCreator runs the following three steps to generate datasets (see Figure 2). First, it sets up an IoT network composed of emulated devices/routers and simulated links (②–④). Second, it performs multi-step attacks and runs IoT applications (⑤–⑦). Finally, it captures network packets at various points, collects host information, and assembles these data into one labeled dataset based on multi-step attacks (⑧–⑩).

Step 1: Preparing an IoT network (②–④). Given a *scenario description* and a list of available physical machines, IoTDSCreator allocates resources to build a victim network and the external Internet. They are implemented as a virtualized network consisting of emulated IoT devices and an attacker node across multiple physical machines to guarantee scalability in the number of devices. To this end, IoTDSCreator collects information about the availability of the physical machines and abstracts the network topology described in the scenario description as a graph. Then, IoTDSCreator splits the graph into several subgraphs based on the available resources in the several physical machines. We devise a greedy algorithm to split the graph. Finally, each of the machines is assigned a subgraph that represents a subnetwork of the entire network.

[5] A list of available keywords for scenario description is described in our public repository (refer to https://github.com/iotdscreator)

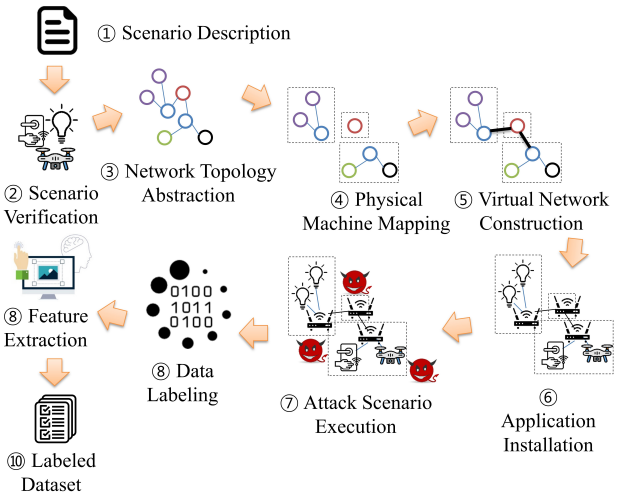


Figure 2: Overview of IoTDSCreator.

Step 2: Running a scenario (⑤–⑦). In this step, IoTDSCreator configures a network and runs a scenario on the network. Each of the physical machines separately builds its assigned subnetwork. On each physical machine, the routers are built first with a sufficient number of network ports. Next, the end devices, the entities on the Internet, and the attackers are created on virtual machines or containers. Then, the links are added to interconnect routers or attach devices to the routers. Finally, the subnetworks are integrated into the entire network by combining the underlying physical machines with the tunneling protocols. Once the network is set, the end devices and the entities on the Internet install their applications including the necessary applications required to collect data, and then start running their applications. Also, the attack is launched according to the scenario description. While running the scenario, the host information is collected at the devices, and the network packets are captured at some vantage points designated by the user and at the attackers' interfaces.

Step 3: Creating a labeled dataset (⑧–⑩). IoTDSCreator summarizes both host and network features into the dataset. For host features, IoTDSCreator lists all the logs reported from devices in chronological order and marks the logs if it is related to specific attacks (e.g., the log before/after the time when a port scanning packet arrived). For network features, IoTDSCreator extracts features from the captured packets. IoTDSCreator labels the attack packets from the captured packets, by referring to the packets captured at the attackers' interfaces. As IoTDSCreator already knows what and when attacks are launched, it can identify attack packets from the packets, providing detailed information about the attack packets, i.e., the corresponding kill chain step (e.g., reconnaissance) and the name of the specific attack vector (e.g., port scanning). In this way, IoTDSCreator is able to precisely label all the packets. Finally, IoTDSCreator returns the labeled dataset of the scenario.

```
1  network_configuration:
2    node:
3      device:
4      - application:
5        - type: iot_hub
6        - type: openssh_server
7        (omit)
8      external:
9      - name: attacker
10       (omit)
11   link:
12   - node1: victim
13     intf1: ext
14     intf2: internet
15     type: ethernet
16   - node1: attacker
17     intf1: ext
18     intf2: internet
19     type: ethernet
```

Figure 3: Example of network configuration.

5 Running Example

This section presents a running example to show how a user can write a scenario description to let IoTDSCreator generate a dataset.

Attack scenario. Our victim device is a Raspberry Pi3 (RPi3) that acts as an IoT hub connected to the Internet. The attacker first sends ping packets to find any victim device that can communicate with, conducts a port scanning to find the open ports on the victim device, and performs TCP SYN flooding. The first two activities are reconnaissance steps, and the final activity is an action step.

Scenario description. That attack scenario description using DC-API is given above. The entities in our virtualized network are described in the network configuration (see Figure 3). The victim, an RPi3, runs the IoT hub application (line 5) and the OpenSSH server (line 6). Both the victim and the attacker are connected to the Internet (lines 11 – 19). The attacks performed by the attacker consist of three steps (see Figure 4). The ping activity is performed at time 10 as reconnaissance (lines 4 – 8). Then, the nmap activity is conducted at time 60 (lines 9 – 12), followed by the TCP SYN flooding (lines 13 – 17) on port number 22, which is the default port of the OpenSSH server. Finally, the dataset is created with the name "ping_nmap_flooding" (line 19).

Dataset generation. IoTDSCreator runs the scenario description and outputs four types of datasets. The packet dataset captures 2,168 packets, each described with 25 packet features at the victim's interface. Each packet is labeled with the corresponding attack step, either of "reconnaissance" or "action" in this scenario, or "benign". Based on the packets, 556 windows are generated, and 51 features per window are extracted. During the attack execution, the victim reports 300 host data. With the host data, IoTDSCreator generates the host dataset that consists of 300 instances with 45 host features. Based on these instances, 40 host windows are generated, and 72 transition features are extracted. Note that the total scenario execution time is 150 seconds, and only 448 seconds are required for all the procedures. The network configuration accounts for 63% (281 seconds) of the total execution in this scenario, mainly due to networking to install necessary applications.

```yaml
attack_scenario:
  scenario_length: 150
  attacks:
  - begin: 10
    attacker: attacker
    type: ping
    target: victim
    step: reconnaissance
  - begin: 60
    attacker: attacker
    type: nmap
    step: reconnaissance
  - begin: 120
    attacker: attacker
    type: flooding
    attack_type: syn_flooding
    port: 22
dataset_feature:
  name: ping_nmap_flooding
```

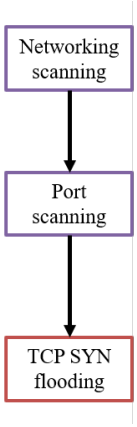


Figure 4: Example of attack scenario.

6 Evaluation

This section evaluates IoTDSCreator by summarizing generated datasets. We also use the datasets to show their effects on augmenting the existing datasets.

6.1 Generated Datasets

We generate 16 datasets through IoTDSCreator. Six datasets contains vulnerabilities in CVEs. For instance, we run an attack exploiting a vulnerability in the log4j library (CVE-2021-44228) on an IoT device. Further, we create a dataset that contains the vulnerability in OpenSSL (CVE-2022-3358). We release the datasets and describe the details of them on our public repository[6].

6.2 Data Augmentation

We experiment to show that the performance of IDS can be improved by adding the dataset of similar scenarios. We fix one scenario as a test set and generate several training sets on similar scenarios, where the training sets contain the same attack patterns, but intervals between patterns are different. We find that recall increases from 0.75 to 1.0 since several training sets can capture diverse patterns. The results show that by generating similar datasets to existing datasets, IoTDSCreator provides robustness of the IDS by adding similar but slightly different patterns.

7 Conclusion

We present IoTDSCreator, a framework for automatically creating the labeled datasets in order to address the issue of the scarcity of datasets in the IoT intrusion detection domain. In the future, we plan to develop a graphic user interface (GUI) tool for ease of describing a scenario and to leverage LLM techniques to automatically generate scenario descriptions from security documents.

Acknowledgments

The work reported in this paper has been funded by the National Science Foundation under grants 2112471 and 2229876, and a grant by Cisco Research.

[6]https://github.com/iotdscreator

References

[1] Netlab 360. 2017. IoT_reaper: A Rappid Spreading New IoT Botnet. https://blog.netlab.360.com/iot_reaper-a-rappid-spreading-new-iot-botnet-en/.
[2] Manos Antonakakis et al. 2017. Understanding the mirai botnet. In *26th USENIX Security Symposium*.
[3] Daniel Arp et al. 2022. Dos and don'ts of machine learning in computer security. In *Proceeding of the USENIX Security Symposium*.
[4] Elisa Bertino. 2016. Data Security and Privacy in the IoT.. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016*. 1–3.
[5] Elisa Bertino and Nayeem Islam. 2017. Botnets and Internet of Things Security. *IEEE Computer* 50, 2 (2017), 76–79.
[6] Tim M Booij et al. 2021. 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* 9, 1 (2021), 485–496.
[7] U Cisco. 2020. Cisco Annual Internet Report (2018–2023) White Paper. *Cisco: San Jose, CA, USA* (2020).
[8] Don Dingee. 2019. IoT, Not People, Now the Weakest Link in Security. https://devops.com/iot-not-people-now-the-weakest-link-in-security/.
[9] Clark Evans et al. 2017. YAML Ain't Markup Language (YAML™) Version 1.2.
[10] Canadian Institute for Cybersecurity. 2009. NSL-KDD dataset. https://www.unb.ca/cic/datasets/nsl.html. (Accessed on 01/08/2024).
[11] Eric M Hutchins et al. 2011. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research* 1, 1 (2011), 80.
[12] Hyunjae Kang et al. 2019. IoT Network Intrusion Dataset. https://ieee-dataport.org/open-access/iot-network-intrusion-dataset. (Accessed on 06/03/2021).
[13] Nickolaos Koroniotis et al. 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. 100 (2019), 779–796.
[14] MIT Lincoln Laboratory. 2000. 2000 Darpa Intrusion Detection Scenario Specific Dataset. https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scenario-specific-datasets. (Accessed on 01/28/2024).
[15] Junjie Liang et al. 2021. FARE: Enabling Fine-grained Attack Categorization under Low-quality Labeled Data. In *Proceeding of Network and Distributed System Security Symposium (NDSS)*.
[16] Richard P Lippmann et al. 2000. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition*, Vol. 2. IEEE, 12–26.
[17] Dave McMillen. 2021. Internet of threats: IoT botnets drive surge in network attacks. https://securityintelligence.com/posts/internet-of-threats-iot-botnets-network-attacks/. (Accessed on 05/23/2024).
[18] Yisroel Mirsky et al. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *Network and Distributed Systems Security (NDSS)*.
[19] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (MilCIS)*. IEEE, 1–6.
[20] Anand Mudgerikar, Puneet Sharma, and Elisa Bertino. 2019. E-spion: A system-level intrusion detection system for iot devices. In *Proceedings of the 2019 ACM Asia conference on computer and communications security*. 493–500.
[21] Thien Duc Nguyen et al. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 756–767.
[22] Jared M Peterson, Joffrey L Leevy, and Taghi M Khoshgoftaar. 2021. A review and analysis of the bot-iot dataset. In *2021 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 20–27.
[23] LLC QoSient. 1980. argus. https://openargus.org/. (Accessed on 05/27/2022).
[24] Check Point Research. 2017. IoTroop Botnet: The Full Investigation. https://research.checkpoint.com/2017/iotroop-botnet-full-investigation/.
[25] Maria Jose Erquiaga Sebastian Garcia, Agustin Parmisano. 2020. IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set]. http://doi.org/10.5281/zenodo.4743746. (Accessed on 05/13/2022).
[26] Inc. Sqrrl Data. 2018. A Framework for Cyber Threat Hunting. https://www.threathunting.net/files/framework-for-threat-hunting-whitepaper.pdf.
[27] Blake E. Storm et al. 2018. MITRE ATT&CK: Design and Philosophy. https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf. (Accessed on 06/03/2021).
[28] Mahbod Tavallaee et al. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 1–6.
[29] CrowdStrike Intelligence Team. 2021. SUNSPOT: An Implant in the Build Process. https://www.crowdstrike.com/blog/sunspot-malware-technical-analysis/.
[30] Imtiaz Ullah and Qusay H Mahmoud. 2020. A scheme for generating a dataset for anomalous activity detection in iot networks. In *Canadian Conference on Artificial Intelligence*. Springer, 508–520.
[31] USTelecom. 2024. Botnet and IoT Security Trends 2024. https://www.ustelecom.org/wp-content/uploads/2024/03/USTelecom-Botnet-and-Security-Trends-2024.pdf. (Accessed on 05/23/2024).