GothX: a generator of customizable, legitimate and malicious IoT network traffic

Manuel Poisson, Kensuke Fukuda, Rodrigo Carnier

GothX: a generator of customizable, legitimate and malicious IoT network traffic

Manuel Poisson
manuel.poisson@irisa.fr
Amossys/CentraleSupélec/CNRS/Univ. Rennes/IRISA
Rennes, France

Rodrigo Matos Carnier
rodrigo_carnier@nii.ac.jp
NII
Tokyo, Japan

Kensuke Fukuda
kensuke@nii.ac.jp
NII/Sokendai
Tokyo, Japan

ABSTRACT

In recent years, machine learning-based anomaly detection (AD) has become an important measure against security threats from Internet of Things (IoT) networks. Machine learning (ML) models for network traffic AD require datasets to be trained, evaluated and compared. Due to the necessity of realistic and up-to-date representation of IoT security threats, new datasets need to be constantly generated to train relevant AD models. Since most traffic generation setups are developed considering only the author's use, replication of traffic generation becomes an additional challenge to the creation and maintenance of useful datasets. In this work, we propose GothX, a flexible traffic generator to create both legitimate and malicious traffic for IoT datasets. As a fork of Gotham Testbed, GothX is developed with five requirements: 1) easy configuration of network topology, 2) customization of traffic parameters, 3) automatic execution of legitimate and attack scenarios, 4) IoT network heterogeneity (the current iteration supports MQTT, Kafka and SINETStream services), and 5) automatic labeling of generated datasets. GothX is validated by two use cases: a) re-generation and enrichment of traffic from the IoT dataset MQTTset,and b) automatic execution of a new realistic scenario including the exploitation of a CVE specific to the Kafka-MQTT network topology and leading to a DDoS attack. We also contribute with two datasets containing mixed traffic, one made from the enriched MQTTset traffic and another from the attack scenario. We evaluated the scalability of GothX (450 IoT sensors in a single machine), the replication of the use cases and the validity of the generated datasets, confirming the ability of GothX to improve the current state-of-the-art of network traffic generation.

CCS CONCEPTS

• **Computer systems organization** → **Sensor networks**; • **Networks** → Network simulations; **Network security**; Sensor networks; • **Security and privacy** → *Intrusion detection systems*; **Network security**; **Intrusion/anomaly detection and malware mitigation**; • **Computing methodologies** → **Machine learning**.

CSET 2024, August 13, 2024, Philadelphia, PA, USA

ACM ISBN 979-8-4007-0957-9/24/08
https://doi.org/10.1145/3675741.3675753

KEYWORDS

Internet of Things, traffic generator, dataset, network security

ACM Reference Format:
Manuel Poisson, Rodrigo Matos Carnier, and Kensuke Fukuda. 2024. GothX: a generator of customizable, legitimate and malicious IoT network traffic. In *Workshop on Cyber Security Experimentation and Test (CSET 2024), August 13, 2024, Philadelphia, PA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3675741.3675753

1 INTRODUCTION

The deployment of billions of IoT devices has changed the landscape of network security. Their platform heterogeneity creates new surfaces of attack, while their high maintenance cost by sheer volume discourages manufacturers from implementing many security measures [6, 7]. This combination has led to increased security breaches and large-scale DDoS attacks [16]. In this arms race, the increase of malicious activity has also increased their footprint on network traffic, which in turn increased the efficacy of network traffic anomaly detection – particularly ML-based AD [5, 10].

Performances of ML models depend on several factors, but the most important is the quality of the data used in their training. For most studies, datasets are the main source of data [9, 25]. To effectively detect current threats to IoT networks, datasets must contain a diversity of data from recent and realistic attacks. Moreover, to minimize false alerts, datasets should also include diverse and realistic samples of legitimate traffic. Nevertheless, some public datasets include only legitimate or only malicious traffic [27]. Generating only legitimate traffic that properly represents IoT heterogeneity is already challenging, but generating malicious IoT traffic that includes a good representation of IoT threats may not even be feasible. An additional issue is the necessity of correct labeling of data to train the ML model. However, some mixed datasets are just captured traffic, leaving the labeling to be made by the user himself. To compound all this, continuous deployment of new IoT technologies turns IoT datasets outdated fast. As a consequence, mixed and heterogeneous IoT traffic datasets, properly labeled and still relevant, are constantly in need.

To alleviate these problems, the training of ML-based traffic AD may use more than one dataset, but it may be challenging to merge and balance their data. In the worst case, existing datasets cannot match the typical traffic (either legitimate or malicious) of the target network where the AD solution will be deployed. If so, either live data must be collected – which poses the problem of availability of relevant malicious traffic, to prepare the model for future attacks – or a traffic generator must be used to create a dedicated dataset [4].

Preparing a testbed from scratch to generate one's own data requires significant time and expertise. Ideally, existing traffic generators should be adapted to IoT traffic and used instead (Section 2). However, setups used to study traffic and generate public datasets are rarely available themselves. When they are, their implementation may not be flexible enough for the needs of new experiments. Another problem is the generated traffic. The testbed may be public and easy to apply in other studies, but the generated traffic may lack the properties described above. For instance, Gotham [26] produces a static dataset mixing legitimate and malicious traffic. However, the output is hard to modify and Gotham provides no way to automatically assign labels to the traffic.

To overcome these issues, we provide GothX, a traffic generator designed to generate customized and labeled datasets of IoT network traffic. It is an improved fork of the replicable IoT testbed for security experiment Gotham. GothX can automatically execute scenarios in a customized topology (e.g. number of IoT sensors). Simulated scenarios present customizable parameters of operation that generate traffic with more varied features (e.g. volume and frequency of messages customized for each node) (Section 3). We exemplify its capabilities with two use cases (Section 4): a) regeneration and enrichment of mixed IoT traffic from the dataset MQTTset, and b) generation of mixed IoT traffic in a custom multi-step attack scenario that results in a DDoS. This scenario is automatically executed in a realistic heterogeneous IoT network setup configured with MQTT and Kafka services (a real-world configuration found in the Japanese academic network SINET [17, 20]). It consists of the execution of a multi-step attack that compromises a network of sensors and generates a DDoS attack.

Moreover, GothX is open-source and made publicly available on GitHub[1]. A last contribution of our work is the publication of two new datasets (Section 4), containing the traffic of the use-cases above. In the process of creating these datasets, we evaluated and validated the scalability and replicability of our solution (Section 5).

In summary, the main contributions of our work are:

(1) GothX: a public, easy-to-use and highly customizable IoT traffic generator, capable of producing rich IoT network traffic.
(2) Critically desired properties of dataset generation, including mixed legitimate and malicious traffic, customizable traffic properties, heterogeneous network setup with MQTT and Kafka services, and automatic labels generation for datasets.
(3) Two labeled datasets: the replication of MQTTset with some improvements and a new dataset with legitimate and malicious traffic based on our attack scenario.

2 RELATED WORKS

Considering the increasing relevance of IoT networks and the compatibility issues that many applications for conventional networks experience when ported to IoT, many simulation tools have been created to facilitate the development of IoT applications. Consequently, surveys on IoT simulation tools are being steadily published [23, 28, 33]. They demonstrate the current diversification of IoT tools, which can be roughly divided in simulators, emulators and testbeds according to the trade-off between scalability and realism of simulation.

However, surveys of tools dedicated to traffic generation cannot be found in the literature except for the specific case of GAN traffic generators [4]. This is a strong indicative that there are not many available tools dedicated to this task. To better contextualize our contributions to the field, we provide in the section below a short review of existing traffic generators and a description of our design goals with respect to the state-of-the-art.

2.1 Existing traffic generators

Traffic generators can be divided into three main categories [33]: hardware-based, simulation-based, and hybrids. Hardware-based tools are made of real IoT devices. They allow testing interactions between devices in the biggest degree of realism. However, it is difficult and expensive to deploy a real network just for traffic generation, and scalability becomes a prohibitive limitation. FIT IoT-LAB [12] tries to mitigate some of these problems by designing rooms with a rich set of IoT devices having different functionalities and providing remote access to researchers willing to perform experiments in these rooms. However, any significant modification of the experiment by users is not feasible. On the other hand, simulation-based tools like CupCarbon-Lab [8] and Gotham [26] rely solely on the memory and computational power of a simulation setup to represent as many devices as possible, with the trade-off of comparatively reduced realism. Nevertheless, with recent advances in computation and virtualization, simulation-based tools currently offer many benefits in terms of scalability and a much smaller cost of deployment, with the trade-off of smaller realism. However, since the realism of hardware-based tools is high concerning devices but very low with respect to the size and topology of IoT networks, virtualization brings a realistic aspect to the traffic generation with respect to the network topology and deployed services. Finally, hybrid tools like the IoTEP [30], combine both real and virtual devices, trying to balance their advantages and limitations.

Simulation-based traffic generators can also differ regarding the process of packet generation. Leaning toward scalability and simplification of simulation are packet synthesizers, like IoT-Flock [13]. They craft custom packets without a stack of network protocols and simulated devices, directly specifying parameters like headers, payloads and flags. Although it significantly increases the flexibility and volume of generated data, the realism and coherence of crafted data can deteriorate significantly and compromise the utility of the data for training ML-based AD solutions. On the other side of the spectrum are testbeds, which virtualize devices to a big degree, create networks with realistic topologies and traffic controllers like servers and switches, and generate traffic as it would be done by actual hardware. This is the approach chosen for GothX.

Gotham [26] is an open-source traffic generator that leverages the popular open-source network emulator GNS3 [15] to simulate IoT devices and generate datasets in a replicable manner. Virtualization is facilitated and automated with Dockerfiles and virtual machines (VMs), while scripts interact with a GNS3 server using the REST API to execute simulation scenarios. The scenario provided

[1]GothX and the datasets are available at https://github.com/fukuda-lab/GothX

by Gotham generates both legitimate and malicious IoT traffic. Legitimate traffic is created using popular IoT devices like MQTT and CoAP. Malicious traffic is created using the automatic launch of attack scenarios from well-known threats, like the botnet Mirai [16].

To the best of our knowledge, Gotham is the only simulation-based testbed for IoT security that provides open-source and free access to its code. Therefore we chose to develop GothX on top of Gotham functionalities, extending it to produce rich mixed network traffic and automatically label the created dataset.

2.2 Missing features in the state-of-the-art

Table 1: Extended features from Gotham to GothX

Features	Gotham	GothX
Open-source	✓	✓
Legitimate + malicious traffic	✓	✓
Virtualization (Docker + VM)	✓	✓
Automatic network initialization	✓	✓
Replicable results	✓	✓
Labeled data		✓
Customizable node behavior		✓
MQTT service	✓	✓
CoAP service	✓	
MQTT-Kafka service		✓
Accompanying ready-made datasets		✓

Gotham made a big contribution to the coordination of efforts in network traffic dataset generation, providing a replicable environment with an automatic setup of network topology and simulation scenario, packaged in an open-source tool that is well documented. To take the next step in the state of the art of traffic generation, we identified three open problems in Gotham: a) The network simulated scenario is not very customizable and has fixed behavior for all IoT sensors; b) it does not generate a labeled dataset, and c) some network traffic typically found in IoT is not represented in Gotham. Below we present a discussion of the impact of these open problems.

In a traffic generator, low customization of network simulation and device behavior translates directly to the quality of the created dataset. By simulating multiple network topologies and patterns of communication, bigger diversity is introduced in the network traffic (a necessity even bigger for intrinsically heterogeneous IoT networks), which is important to train ML-based AD solutions with high precision and recall. The ability to easily create different datasets using the same tool is beneficial not only for the creation of such AD models, but also for the study of feature selection itself. Explainable artificial intelligence (XAI) [18] is one example of research field that could leverage the abundance of variable data. Another example is the improvement and update of public datasets, which is difficult due to the lack of availability of tools for dataset replication. GothX provides an easy customization in its configuration files of a simulation. Since Gotham hard-codes these aspects of simulation in the testbed itself, it is difficult to generate many different scenarios and consequently different traffic data.

Unlabeled datasets are problematic for numerous reasons. Besides requiring significant additional work of manual labeling by users, errors in labeling are much more likely to occur due to the unfamiliarity with the conditions of simulation that created the dataset. Moreover, unbalanced data is likely to escape the attention of the dataset creators before it goes public, as is the chance of duplicated packets. To provide an internal degree of documentation in the GothX-generated datasets themselves, we devised a method of labeling the datasets in the same workflow of the simulation scenario. More details will be provided in Section 3.

Regarding the representation of different IoT traffic, we highlight the importance of increasing the heterogeneity of simulated traffic in more than one way. Besides the configuration of network topology and nodes' operation, GothX has additional communication services with their own servers in its simulation scenarios: Kafka service, a popular event-streaming platform; a Kafka-MQTT connection server, that consolidates traffic data from multiple MQTT networks; and SINETStream service [29], which is built on top of the services above and is an example of a real-world IoT application.

A final commentary is the absence of different public datasets generated by Gotham. Since Gotham is configured to produce the same type of traffic, it generates the same dataset when executed. We highlight the new features of traffic diversity in GothX by providing two essentially different datasets regarding malicious traffic. Both are automatically labeled. More details on their differences will be presented in Sections 4.1 and 4.2.

Table 1 summarizes the difference in implemented features between Gotham and GothX.

3 ARCHITECTURE DESIGN FOR CUSTOMIZATION AND EXTRA IMPROVEMENTS

This paper presents GothX, an improved fork of Gotham designed to generate customized labeled datasets. Like Gotham, we use a real topology where nodes interact with each other automatically. Agents running on nodes emit legitimate network traffic to other nodes that react accordingly. On top of that, malicious traffic is produced by an attack agent. Usage of such real topology in GothX allows future addition of new nodes interacting with nodes already present. For example, one could integrate to GothX a node with an IDS to evaluate the online performances of the IDS.

One of GothX's advantages is that it adds interactions using Kafka. Furthermore, the labeling process of GothX's datasets is automated. However, the main value of GothX is the easy customization of the generated datasets. Customization leads to a large number of settings' combinations which favors the diversity of generated datasets. Thanks to the easy customization of topology and scenario, GothX can provide multiple datasets, with only one precise, known, setting modified between each variation. Finally, customization capabilities of GothX allow replication and update of existing datasets.

This Section presents GothX's main components, how they interact and how they allow to customize the datasets produced. It presents the customizable settings of the legitimate and malicious traffic contained in the datasets generated by GothX. Finally, the benefits of GothX are not solely about customization. This Section's

end presents extra improvements fixing performance and realism issues.

3.1 Architecture

To generate customized labeled datasets, GothX follows the workflow depicted in Figure 1. At first, it is necessary to create templates of the future topology's nodes. Then, GothX's scripts use configuration files to influence the data they generate. In the end, this data is labeled to form the final dataset. Figure 2 explains how GothX interacts with GNS3 and leverages CICFlowMeter [11]. Basically, GothX's role is to automatically generate a topology in GNS3 and execute actions in this topology. CICFlowMeter is used to extract network flows. GothX's GitHub repository provides documentation about installation and usage to create the custom topology and run a configured scenario as well as the labeling process.

The configuration file `config_topology` defines different topology's configurations. A configuration influences how many nodes of each type will be generated in the topology. The second configuration file `config_scenario` references a previously created topology. The scenario defines which nodes will be started and how they will behave. It also describes which network traffic will be captured.

`create_topology` creates a topology in GNS3 based on the topology's configuration file (step 1 on Figure 2). Then, `run_scenario` uses `config_scenario` to automatically perform legitimate and malicious actions in the topology (step 2 on Figure 2). It creates some `pcap` files containing raw network packets generated during the scenario execution.

Labeling is required to allow usage of the dataset for supervised machine learning because `pcap` files mix legitimate and malicious traffic. The labeling is a two-steps process. At first (step 3 on Figure 2), CICFlowMeter extracts, in a `csv` file, network flows from each `pcap` file, alongside different features. Then a labeling script, part of GothX, automatically labels each network flow (step 4 on Figure 1). It allows anyone to distinguish legitimate traffic among different types of malicious traffic. For example, this is useful in the case of training a model performing multi-class classification to differentiate each step of a long attack like discriminating DDoS from ports scan.

3.2 Customization

Table 2: Customizable topology and scenario parameters

Legitimate traffic	Malicious traffic
Sensors count	Parameters of attack tool
Messages rate* (periodic/random)	Strength of DDoS attack (e.g. payload size)
(In)activity duration*	% of compromised sensors
Which data, from a dataset of real sensors, is sent*	Sleep time between attack steps
Traffic volume (MQTT/Kafka)*	

*customizable for each sensor independently

With GothX, both the topology and the scenario can be easily customized using the configuration files described in Section 3.1. It is very helpful to generate datasets dedicated to the need. Customization of the scenario is useful in the context of anomaly detection to analyze the efficiency of the detector when legitimate traffic varies but the attack is the same, or vice versa. For example, it can be useful in AI and XAI because it helps to study the impact of a specific parameter on a machine learning model for anomaly detection.

Table 2 details parameters that can be customized concerning the malicious and legitimate traffic generated by GothX. The malicious traffic comes from the automatic execution of an attack scenario. This scenario is implemented and published jointly with GothX. Modifying the parameters, one can define a waiting time of three hours between payload transfer and DDoS start, for example. Varying the arguments of the attack tools will impact the strength, speed and therefore stealthiness of the attack. Attackers tend to increase their stealthiness by increasing the time between attack stages or by making slower scans or brute-force. GothX's scenario configuration file allows anyone to study the impact of such variations on the ability to detect the attack.

3.3 Solved performances and realism issues

GothX includes various improvements and extra functionalities to state-of-the-art as detailed below.

No graphical user interface is necessary to use GothX, which enables its usage on a remote server. Embedded install scripts allow to install and setup all requirements. Similarly, usage only consists of optionally modifying configuration files and executing scripts. It is therefore possible to go from an empty computer to the full execution of a scenario using a command line interface only.

When deploying a topology in GNS3, configuration of VM nodes (i.e. install image disk, set network configuration) is time-consuming. With GothX, all VM nodes are configured in parallel and in the background. It significantly reduces the time required to deploy a topology, making it more convenient to quickly experiment with new topologies.

In GothX, the realism of the data contained in MQTT messages is assured by reading line by line publicly available datasets containing data collected from real sensors. Furthermore, it is possible to choose which columns of the dataset is read by a simulated sensor. This way, multiple simulated sensors can read from the same dataset and still send different data, which improves data diversity. Moreover, this new feature allows the definition of network packet size by modifying the number of columns contained in the published MQTT message.

4 USE CASES

4.1 MQTTset replication

GothX allows to replicate MQTTset [32], a public dataset containing network packets involving MQTT. This validates GothX's ability to generate good-quality data. It also demonstrates that thanks to customization, GothX can generate data following precise characteristics previously defined. Indeed, we could check that our dataset presents the same characteristics as those of MQTTset. For example, we have the same number of packets/sec. and Bits/sec. in TCP flows.

MQTTset was produced to train machine learning models to detect security issues in MQTT exchanges. It is composed of six IoT

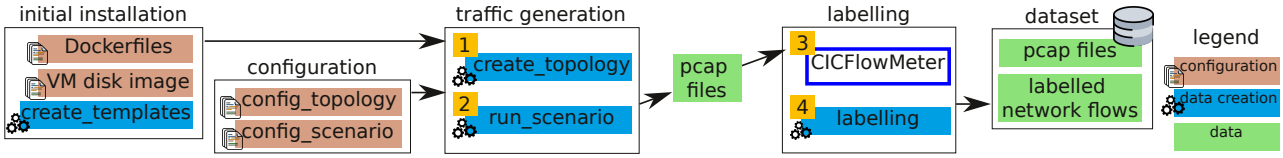

Figure 1: GothX's workflow

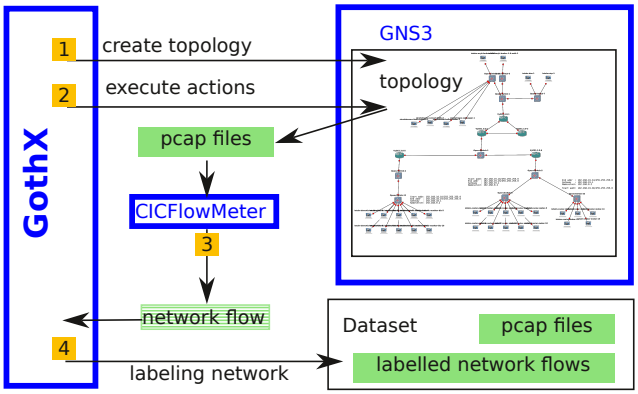

Figure 2: GothX's interaction with other tools

traffic traces. One trace contains only legitimate traffic. It represents 10 distinct IP addresses publishing messages to 1 MQTT broker. 5 IP addresses send data periodically, every 60, 120 or 180 seconds. 5 IP addresses send data at a random interval, with a mean time of one second between two messages. This file was generated using the traffic generator IoT-Flock [13]. This tool generates a pcap file with network packets as they would appear in real communications between IoT devices. Each of the five other traffic traces in MQTTset contains network packets about a different type of attack against MQTT. Table 3 details the type of attack and the tool used to generate the file.

Table 3: Attack types and corresponding tools in MQTTset

Attack type	Tool
MQTT publish flood, CVE-2018-1684	IoT-Flock [13]
Flood DoS	MQTT-malaria [1]
SlowITe	SlowTT [31]
Malformed data	MQTTSA [22]
Authentication bruteforce	MQTTSA [22]

With GothX, we produced a dataset of six traffic traces with network traffic having the same characteristics as MQTTset. Files containing an attack were replicated using the same attack tools, except for the one generated by IoT-Flock (first line in Table 3). The latter was replicated using MQTT-malaria [1], a tool for testing scalability in MQTT environments, with some arguments leading to the same effect. (namely, `malaria publish -P 1 -n 265 -H 192.168.2.1 -s 30700`). Each of the six traffic traces in our dataset has the same number of distinct IP addresses as in their corresponding traffic trace in MQTTset. We confirmed that our datasets have similar statistics in each TCP conversation regarding the number of packets, the number of bytes, the bits rate (bits/sec.). For example, the flood DoS (second line in Table 3) in MQTTset consist of 44M packets received by an MQTT broker at the bit rate of 2.8K bits/sec. during 126 seconds. In our dataset, there are 62M packets received at the bit rate of 4.5K bits/sec. during 110 seconds. Finally, we could confirm that ML models for anomaly detection had similar results on MQTTset and our replication.

Our dataset contains some differences with MQTTset, which makes it better than the original dataset. Our legitimate traffic was generated in GothX with one node being the MQTT broker and ten nodes the sensors. Real MQTT messages are exchanged between the broker and the sensors. In MQTTset on the contrary, legitimate traffic was synthesized with IoT-Flock, which only crafts network packets. Real interactions allow anyone to study how the attack interferes with legitimate actions. This is particularly interesting for DoS which goal is precisely to disrupt legitimate usage. In MQTTset, legitimate traffic is fully separated from the malicious one while GothX can mix both. A mix is more realistic because real network captures of an attack usually also contain legitimate traffic and the main difficulty of intrusion detection systems (IDS) is to distinguish normal and malicious traffic. Furthermore, our dataset is more realistic than MQTTset because it is more diverse. It contains not only network data involving MQTT, but also DNS, NTP, ARP and ICMP exchanges. This is the diversity that exists in real network traffic and IDS must be trained to deal with it to be efficient in real networks.

4.2 New realistic complete attack scenario

As a second use case, we used GothX to generate a dataset from the automatic execution of a scenario mixing legitimate and malicious actions between IoT devices. This section presents the topology where the scenario takes place. It details the different steps of the complete *kill-chain* followed by an attacker, from the initial compromision by exploiting a recent critical vulnerability (CVE-2023-25194 [2]) to a DDoS. Analysis of the generated network traffic validates the smooth execution of the scenario, both for legitimate and malicious actions.

4.2.1 Topology. The topology used in the scenario at the origin of our dataset is depicted in Figure 3. It contains:

- 450 nodes simulating IoT sensors sending MQTT messages
- 3 MQTT brokers
- 1 Kafka broker
- 1 node «*kafka-connect*», executing Kafka-connect
- 1 node «*client-connect*», the client of «*kafka-connect*»

Using the topology configuration file described in Section 3.1, a user of GothX who wants to perform a small-scale experiment can

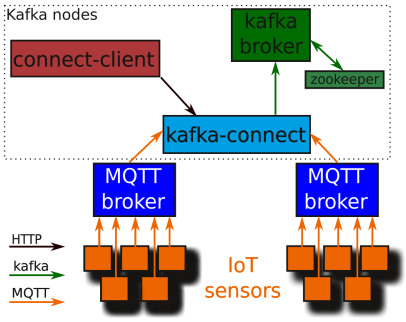

Figure 3: Overview of the simulated network

easily generate a smaller topology with fewer than 450 IoT devices or fewer than three MQTT brokers or no node related to Kafka.

Legitimate traffic is generated by IoT sensors sending messages to MQTT brokers. Then, Kafka-connect is used to read topics in MQTT brokers and transfer the messages to a Kafka broker. The simulated environment is similar to existing IoT architectures mixing MQTT and Kafka traffic, like SINETStream [19, 29]. Two MQTT brokers receive clear text messages, with one of them requiring device authentication with a login and password before publishing. The third receives TLS encrypted messages, without prior authentication.

Some sensitive, but plausible, settings have been defined to introduce vulnerabilities allowing execution of a realistic attack scenario from the initial compromission of a node to a DDoS by multiple IoT sensors. On IoT devices, the package `ssh-server` is available so that it's possible to activate a remote SSH connection. Also, the Kafka-connect component runs version 7.3.1 from confluent. This version was released in December 2022 and is the last version vulnerable to the CVE-2023-25194 [2]. Moreover, Kafka-connect is a Java program for which we intentionally set the system property `UnableUnsafeSerialization` to `true`. Therefore, an attacker able to send `POST` requests to Kafka-connect can exploit the CVE-2023-25194 and get a remote code execution (RCE) on the node executing Kafka-connect.

4.2.2 Steps of complete attack scenario. Based on what can be seen in recent attacks launched against IoT networks [3, 14, 24], we introduce a new realistic attack scenario. Our scenario considers an internal threat with the attacker initially controlling the node «*client-connect*». He begins by sending legitimate `POST` requests to «*kafka-connect*» and then starts executing malicious actions remotely. The scenario is divided into six steps.

(1) From the node «*client-connect*», the attacker exploits the CVE-2023-25194 affecting Kafka-connect. It allows him to gain a remote code execution (RCE) on the node «*kafka-connect*».
(2) The attacker uses the RCE to transfer attack tools from «*client-connect*» to «*kafka-connect*» with the utility `wget`. Then, he configures the attack tools on «*kafka-connect*» and he opens a reverse shell using `netcat`.
(3) From «*kafka-connect*», the attacker scans the network, searching for devices with port 22 listening, which is the usual port for SSH. The command executed is `./nmap -Pn -oG ips.txt 192.168.18-20.10-150 –max-rate 0.7 -p 22`.
(4) For nodes detected as listening to SSH communication, the attacker searches for the right SSH credentials by trying multiple combinations of username/password using dictionaries of commonly used credentials. This is done from «*kafka-connect*», with the command `hydra/hydra -o success.txt -M ssh_ips.txt ssh -f -L u.txt -P p.txt -t 2` executed.
(5) For each node where he found the right SSH credentials, the attacker transfers his payload from «*kafka-connect*» to the compromised node using `scp`, which is dedicated to transferring files over SSH. Here, the payload is the tool `mqttsa`.
(6) From «*kafka-connect*» and using SSH, the payload is triggered simultaneously on all compromised nodes which all execute the command `mqttsa -fc 100 -fcsize 10 -sc 2400 192.168.2.1`. This is the beginning of the DDoS targeting the MQTT broker at the IP address `192.168.2.1`.

When a user of GothX wants to execute this scenario, he can choose the arguments of attack commands with the scenario configuration file presented in Section 3.1. Moreover, a user only interested in studying the DDoS can skip all steps except for the last one. In this case, desired nodes, simulated by docker-containers, are considered as already compromised. `mqttsa` is installed on the compromised nodes by copying the file from the machine hosting GothX to the containers, without generating network traffic. Similarly, the payload is triggered by executing a command on the containers from the host, without generating network traffic. This use case with DDoS only was used to evaluate in-kernel anomaly detection using eBPF [21] with various proportions of compromised nodes.

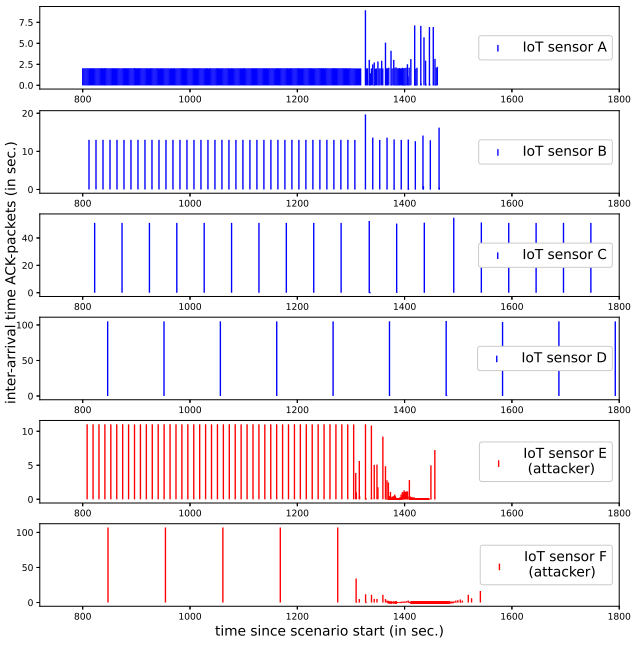

Figure 4: Inter-arrival time of ACK-packets during scenario

4.2.3 Validation of generated network traffic. To validate that the generated traffic was the one expected, we analyzed the traffic traces resulting from the execution of our scenario. In Figure 4, each plot shows, for a given destination IoT sensor, the inter-arrival time of some specific packets with respect to the time since the scenario's beginning in seconds. We focus on packets emitted by MQTT brokers sending acknowledgments to IoT sensors, the ones with the TCP flag ACK. A first ACK-packet is sent when an IoT sensor establishes an MQTT connection. Almost immediately after (the inter-arrival time is near zero), a second ACK-packet is sent when the broker correctly receives an MQTT message. These plots allow to verify that (before being disrupted by the attack) MQTT brokers correctly receive messages at the rate defined in the file `config_scenario`, even when some sensors publish messages faster than one message per second.

When everything works fine, the cycle repeats itself periodically. This can be seen on the left of the plots in Figure 4, before 1200 seconds. For example, the second plot shows an MQTT broker acknowledging MQTT connections and publications every 12 seconds for the IoT sensor B.

During the DDoS, MQTT brokers acknowledge, to the nodes launching the attack, many MQTT connections and publications in a very short time. The inter-arrival time between two ACK-packets sent to attacking nodes is very small. This is represented after about 1200 seconds of execution, by the drop near zero of plots E and F related to the nodes representing compromised IoT sensors responsible for the DDoS. For non-attacking nodes, acknowledgments of some MQTT connections are missed or delayed. This is shown by the increase of plots A and B, related to non-attacking IoT sensors. This phenomenon is more noticeable for nodes with a high publication frequency (e.g. IoT sensor A) but appears for almost all nodes.

We notice that after DDoS, most plots stop before the end of the scenario. IoT sensors were unable to contact their MQTT broker. They considered the broker as unreachable and stopped sending connection requests. Before the DDoS, 225 IoT nodes received ACK-packets from the broker targeted by the attack. The scenario last about 1800 seconds and after 1700 seconds, the victim MQTT broker stopped sending ACK-packet to 82% of these IoT nodes. This analysis shows that the DDoS is efficient because it definitely disrupts the normal behavior of nodes.

4.2.4 Provided dataset based on our attack scenario. The dataset produced is composed of a text file detailing settings of the scenario execution, three `pcap` files containing raw network packets, associated with three label files. The text file gives, among others, the commands executed by the attacker at each attack step and the IP address of nodes involved in the DDoS.

Our goal was to have the minimum number of network probes to capture all the network traffic generated during the scenario. It leads to three probes recording three traffic traces in three `pcap` files. One `pcap` file contains only legitimate traffic about the communication between Zookeeper and the Kafka broker. Another `pcap` file contains all traffic related to the node «*kafka-connect*». In particular, there is the legitimate transfer of messages from MQTT brokers to the Kafka broker. This is mixed with the malicious traffic related to the exploitation of the CVE [2], the attack tool transfer on node «*kafka-connect*» and the commands sent through the `netcat` reverse shell. The third and last `pcap` file contains all legitimate traffic about IoT sensors sending MQTT messages to MQTT brokers. This is mixed with the following malicious activity: ports scan, credential brute force, payload transfer on compromised devices via `scp` and DDoS of the MQTT broker.

Since `pcap` files mix legitimate and malicious traffic, labeling is required to allow the usage of the dataset for supervised machine learning. Furthermore, our labeling is useful to distinguish different types of malicious traffic in case of training an IDS to differentiate each step of the attack like discriminating DDoS from ports scan for example. It is possible to know, for every single packet, whether it corresponds to benign traffic (labeled as *normal*) or if it is part of the attack process. Labels about the attack are *cve_exploitation*, *reverse_shell*, *scan_ports*, *credentials_bruteforce*, *transfer_payload_to_iot* and *mqttsa_slowite*. They detail to which attack step the packet is related to. Our labeling script is adaptive to the various settings applied when the scenario was executed. Using GothX, everyone can generate his custom dataset fitting his need and label it using our labeling script.

GothX's customization capabilities, detailed in Table 2, allow anyone to generate variations of our dataset. For example, one can easily change one line in the scenario configuration file to consider a DDoS with 25% of compromised nodes, instead of 50%. With this new dataset, he can see if the disruption of legitimate traffic is reduced and if an IDS can still detect the attack. Another interesting experiment would be to evaluate the evolution of an IDS performance when the proportion of legitimate over malicious traffic increases. GothX's customization makes this possible by allowing an easy variation in the size and frequency of legitimate messages while keeping the same attack.

5 DISCUSSION

5.1 Identification of the scalability bottleneck

The scalability of GothX is defined as the number of simulated devices that can easily run simultaneously. To be the most realistic possible, scalability is required because real IoT networks and DDoS involve many devices. We want to easily increase the number of IoT sensors in the topology. In particular, we identified the bottleneck to the maximum number of IoT sensors that can be simulated in GothX. We assessed two challenges to overcome to achieve scalability: hardware resources and data realism. Finally, we measured execution time.

Hardware resources. In terms of memory (RAM) consumption, Gotham and GothX behave the same way. Based on measurements of Gotham's paper, we can infer that R, the amount of RAM in megabytes necessary to run nodes, is about $R = q * 470 + d * 37$, where d, respectively q, is the number of nodes based on docker, respectively based on qemu VMs, running simultaneously. In GothX, we could execute a scenario involving 450 IoT devices. Such topology is made of 498 docker nodes for IoT devices, switches, brokers, and four qemu VMs nodes for routers. It uses 20.4GB of RAM, which is not a strong limitation, even for some laptops.

Another hardware resource we considered that could limit scalability is the CPU usage. Our measurements showed that the number

of running nodes does not strongly impact CPU usage. However, depending on what a node is doing, it may use a lot of CPU power. In particular, a scenario with some settings leading to a strong DDoS creates many processes on the machine running the topology and CPU consumption is high. Even with only two IoT devices set to launch the DDoS, this can lead a laptop to freeze if settings are dis-proportioned. It can be noted that, since GothX adds the ability to be executed on a remote server, it eases access to computers with a lot of memory and CPU power.

Realism. The realism of the simulated network must be kept in mind while increasing the number of devices. In particular, duplicating dozens of times devices with the same behavior easily increases the number of simulated nodes but is only minimally representative of a real IoT network. GothX realism lies in its ability to generate network traffic similar to the one in a real IoT network with multiple different sensors sending their data to some brokers. Data sent by IoT devices simulated in GothX is read from public datasets where each column contains data registered by a real sensor. The cumulated number of real sensors represented in the public datasets we use does not reach 450. Therefore, with 450 simulated sensors in the topology, not all simulated sensors can publish data from a different real sensor. We lose realism with multiple simulated sensors publishing the same data from the same real sensor. However, configuring simulated sensors to publish different combinations of data from multiple real sensors mitigates the realism loss.

Execution time. Two different execution times can be distinguished. The execution time for data generation is the same as the scenario duration. This time is fully customizable using the scenario configuration file and depends on the temporal scope of the dataset desired by GothX's user. Concerning the time needed to deploy a topology, prior to a scenario execution, it depends on the number of VM nodes and docker nodes in the topology but remains reasonable. Docker nodes are fast to configure. Creating and configuring one VM takes a median time of 278 seconds based on our multiple measurements. Thanks to the parallel configuration of VMs in the background, multiple VMs can be configured simultaneously and during the creation of docker nodes. Following our multiple executions on a laptop, GothX takes about 26 minutes to create a large topology composed of four VM nodes and 498 docker nodes. The topology we used to replicate MQTTset contains 5 VM nodes and 29 docker nodes and is created in about less than 3 minutes.

5.2 Replicability

Replicability is the recreation of the same experimental apparatus, and using it to perform exactly the same experiment. We paid particular attention to replicability at different levels.

At first, the traffic generator itself is replicable. The code is open-source, and available on GitHub. We checked that GothX's documentation allowed peers to use our traffic generator without external help. Validation of the portability was done by verifying GothX's ability to be installed and generate the same topology on different laptops and on a remote server with the operating systems Fedora and Ubuntu. This verifies that GothX can be used on different host computers and is not dependent on a specific operating system.

Secondly, we assessed the replicability of the generated dataset. We could verify that executing GothX twice with the same settings generates the same dataset in the end. One of GothX's strengths is its ability to replicate existing datasets. Specifically, users can take a public dataset and generate a new one with similar characteristics. This is one of the use cases detailed in Section 4.1 for MQTTset. Furthermore, GothX is useful for generating variations of an existing dataset. A user can replicate a famous dataset and make sure he gets similar results with the copy. Then he can make new experiments to understand what would have changed if a given setting had been different in the original dataset.

6 CONCLUSION

GothX is an open-source, replicable and highly customizable traffic generator for heterogeneous IoT networks. It sets up virtual devices, network topologies and specific simulation scenarios in a completely automatic fashion. Parameters of simulation are easily modifiable through configuration files that are well documented in the GitHub repository. From its diverse simulation scenarios, GothX captures rich mixed IoT traffic and creates traffic datasets automatically. The following new features in the state-of-the-art of traffic generation are implemented in GothX: highly customizable node behavior, specific mechanisms of traffic diversification for both legitimate and malicious traffic, automatic labeling of generated dataset, and increased heterogeneity of network topology through automatic configuration of MQTT and Kafka networks.

We validated GothX with two use cases: the replication and enrichment of the simulation that generated the mixed traffic dataset MQTTset, emphasizing the diversification of legitimate traffic parameters; and the simulation of a six-step scenario involving CVE-2023-25194 exploit, nodes infection, and the subsequent DDoS attack, emphasizing the diversification of malicious traffic parameters. Through these two use cases, Gotham generated the mixed IoT traffic used to create two new datasets, both of which we make available publicly. The current scalability of the solution allows up to 450 nodes simulated at a cost of 20.4GB of RAM. CPU cost is also feasible for single machines even during the simulation of weaker DDoS attacks.

ACKNOWLEDGMENTS

Manuel Poisson is supported by the NII internship program, the Collège doctoral de Bretagne, and the research team PIRAT\'); This work is partly supported by JST CREST JPMJCR21M3.

REFERENCES

[1] 2024. etactica/mqtt-malaria. https://github.com/etactica/mqtt-malaria (Accessed on May 13th, 2024).
[2] 2024. NVD - CVE-2023-25194. https://nvd.nist.gov/vuln/detail/CVE-2023-25194 (Accessed on May 13th, 2024).
[3] Mahmoud Abbasi, Marta Plaza-Hernandez, Javier Prieto, and Juan M. Corchado. 2022. Security in the Internet of Things Application Layer: Requirements, Threats, and Solutions. *IEEE Access* 10 (2022), 97197–97216. https://doi.org/10.1109/ACCESS.2022.3205351
[4] Tertsegha Anande and Mark Leeson. 2023. WRAP-Generative-adversarial-networks-GANs-survey-network-traffic-generation-2022. *International Journal of Machine Learning and Computing* 12 (10 2023), 333 – 343. https://doi.org/10.18178/ijmlc.2022.12.6.1120
[5] The UCI KDD Archive. 1999. KDD Cup 1999 Data. https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (Accessed on May 13th, 2024).

[6] Mikael Asplund and Simin Nadjm-Tehrani. 2016. Attitudes and Perceptions of IoT Security in Critical Societal Services. *IEEE Access* 4 (2016), 2130–2138. https://doi.org/10.1109/ACCESS.2016.2560919

[7] Saurabh Bagchi, Tarek F. Abdelzaher, Ramesh Govindan, Prashant Shenoy, Akanksha Atrey, Pradipta Ghosh, and Ran Xu. 2020. New Frontiers in IoT: Networking, Systems, Reliability, and Security Challenges. *IEEE Internet of Things Journal* 7, 12 (2020), 11330–11346. https://doi.org/10.1109/JIoT.2020.3007690

[8] Ahcene Bounceur, Olivier Marc, Massinissa Lounis, Julien Soler, Laurent Clavier, Pierre Combeau, Rodolphe Vauzelle, Loic Lagadec, Reinhardt Euler, Madani Bezoui, and Pietro Manzoni. 2018. CupCarbon-Lab: An IoT emulator. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, Las Vegas, NV, 1–2. https://doi.org/10.1109/CCNC.2018.8319313

[9] François De Keersmaeker, Yinan Cao, Gorby Kabasele Ndonda, and Ramin Sadre. 2023. A Survey of Public IoT Datasets for Network Security Research. *IEEE Communications Surveys & Tutorials* 25, 3 (2023), 1808–1840. https://doi.org/10.1109/COMST.2023.3288942

[10] Sohaila Eltanbouly, May Bashendy, Noora AlNaimi, Zina Chkirbene, and Aiman Erbad. 2020. Machine Learning Techniques for Network Anomaly Detection: A Survey. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*. 156–162. https://doi.org/10.1109/ICIoT48696.2020.9089465

[11] Gints Engelen, Vera Rimmer, and Wouter Joosen. 2021. Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study. In *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 7–12.

[12] Eric Fleury, Nathalie Mitton, Thomas Noel, and Cédric Adjih. 2015. FIT IoT-LAB: The Largest IoT Open Experimental Testbed. *ERCIM News* 101 (April 2015), 4. https://inria.hal.science/hal-01138038

[13] Syed Ghazanfar, Faisal Hussain, Atiq Ur Rehman, Ubaid U. Fayyaz, Farrukh Shahzad, and Ghalib A. Shah. 2020. IoT-Flock: An Open-source Framework for IoT Traffic Generation. *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)* (March 2020), 1–6. https://doi.org/10.1109/ICETST49965.2020.9080732 Conference Name: 2020 International Conference on Emerging Trends in Smart Technologies (ICETST).

[14] Junaid Haseeb, Masood Mansoori, and Ian Welch. 2020. A Measurement Study of IoT-Based Attacks Using IoT Kill Chain. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (Dec. 2020), 557–567. https://doi.org/10.1109/TrustCom50675.2020.00080 Conference Name: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).

[15] Saida Helali. 2020. *Simulating Network Architectures with GNS3*. 9–25. https://doi.org/10.1002/9781119779964.ch2

[16] Georgios Kambourakis, Constantinos Kolias, and Angelos Stavrou. 2017. The Mirai botnet and the IoT Zombie Armies. In *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*. 267–272. https://doi.org/10.1109/MILCOM.2017.8170867 ISSN: 2155-7586.

[17] Takashi Kurimoto, Koji Sasayama, Osamu Akashi, and Shigeo Urushidani. 2023. SINET6: Nationwide 400GE-Based Academic Backbone Network in Japan. 1–3. https://doi.org/10.23919/OFC49934.2023.10117428

[18] Nour Moustafa, Nickolaos Koroniotis, Marwa Keshk, Albert Y. Zomaya, and Zahir Tari. 2023. Explainable Intrusion Detection for Cyber Defences in the Internet of Things: Opportunities and Solutions. *IEEE Communications Surveys & Tutorials* 25, 3 (2023), 1775–1807. https://doi.org/10.1109/COMST.2023.3280465 Conference Name: IEEE Communications Surveys & Tutorials.

[19] National Institute of Informatics. 2021. SINETStream. https://www.sinetstream.net/index.en.html Japan. (Accessed on May 13th, 2024).

[20] National Institute of Informatics. 2024. SINET6. https://www.sinet.ad.jp Japan. (Accessed on May 13th, 2024).

[21] Atsuya Osaki, Manuel Poisson, Seiki Makino, Shiiba Ryusei, Kensuke Fukuda Fukuda, Okoshi Tadashi, and Jin Nakazawa. 2024. Dynamic Fixed-point Values in eBPF: a Case for Fully In-kernel Anomaly Detection. *Proceedings of the 19th Asian Internet Engineering Conference* (August 2024), 9 pages.

[22] Andrea Palmieri, Paolo Prem, Silvio Ranise, Umberto Morelli, and Tahir Ahmad. 2019. MQTTSA: A Tool for Automatically Assisting the Secure Deployments of MQTT Brokers. In *2019 IEEE World Congress on Services (SERVICES)*, Vol. 2642-939X. 47–53. https://doi.org/10.1109/SERVICES.2019.00023 ISSN: 2642-939X.

[23] N D Patel, B M Mehtre, and Rajeev Wankar. 2019. Simulators, Emulators, and Test-beds for Internet of Things: A Comparison. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 139–145. https://doi.org/10.1109/I-SMAC47947.2019.9032519

[24] Rachit, Shobha Bhatt, and Prakash Rao Ragiri. 2021. Security trends in Internet of Things: a survey. *SN Applied Sciences* 3, 1 (Jan. 2021), 121. https://doi.org/10.1007/s42452-021-04156-9

[25] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. 2019. A survey of network-based intrusion detection data sets. *Computers & Security* 86 (2019), 147–167. https://doi.org/10.1016/j.cose.2019.06.005

[26] Xabier Saez-de Camara, Jose Luis Flores, Cristóbal Arellano, Aitor Urbieta, and Urko Zurutuza. 2023. Gotham Testbed: A Reproducible IoT Testbed for Security Experiments and Dataset Generation. *IEEE Transactions on Dependable and Secure Computing* PP (Jan. 2023), 1–18. https://doi.org/10.1109/TDSC.2023.3247166

[27] Vaishali Shirsath. 2023. CAIDA UCSD DDoS 2007 Attack Dataset. https://ieee-dataport.org/documents/caida-ucsd-ddos-2007-attack-dataset (Accessed on May 13th, 2024).

[28] Abhishek Singh, Himanshu Nandanwar, and Anamika Chauhan. 2022. Simulation Tools and Testbeds for Internet of Things(IoT): "Comparative Insight". In *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*. 1–7. https://doi.org/10.1109/ICCSEA54677.2022.9936302

[29] Atsuko Takefusa, Jingtao Sun, Ikki Fujiwara, Hiroshi Yoshida, Kento Aida, and Calton Pu. 2021. SINETStream: Enabling Research IoT Applications with Portability, Security and Performance Requirements. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. 482–492. https://doi.org/10.1109/COMPSAC51774.2021.00073 ISSN: 0730-3157.

[30] Fernando Terroso-Saenz, Aurora González-Vidal, Alfonso P. Ramallo-González, and Antonio F. Skarmeta. 2019. An open IoT platform for the management and analysis of energy data. *Future Generation Computer Systems* 92 (March 2019), 1066–1079. https://doi.org/10.1016/j.future.2017.08.046

[31] Ivan Vaccari, Maurizio Aiello, and Enrico Cambiaso. 2020. SlowITe, a Novel Denial of Service Attack Affecting MQTT. *Sensors* 20 (May 2020), 2932. https://doi.org/10.3390/s20102932

[32] Ivan Vaccari, Giovanni Chiola, Maurizio Aiello, Maurizio Mongelli, and Enrico Cambiaso. 2020. MQTTset, a New Dataset for Machine Learning Techniques on MQTT. *Sensors* 20, 22 (Jan. 2020), 6578. https://doi.org/10.3390/s20226578 Number: 22 Publisher: Multidisciplinary Digital Publishing Institute.

[33] Shicheng Zhu, Shunkun Yang, Xiaodong Gou, Yang Xu, Tao Zhang, and Yueliang Wan. 2022. Survey of Testing Methods and Testbed Development Concerning Internet of Things. *Wireless Personal Communications* 123, 1 (March 2022), 165–194. https://doi.org/10.1007/s11277-021-09124-5