# SATB: A Testbed of IoT-based Smart Agriculture Network for Dataset Generation

Liuhuo Wan[1], Yanjun Zhang[1 2], Ruiqing Li[1],
Ryan Ko[1], Louw Hoffman[1], and Guangdong Bai[1✉]

[1] The University of Queensland, Australia
[2] Deakin University, Australia
g.bai@uq.edu.au

**Abstract.** Agriculture has seen many revolutions since the booming Internet of Things (IoT) was embedded to enable the *smart agriculture* (SA) scenarios. SA integrates end devices, gateways and clouds to digitalize and automate traditional farming methods. Due to the open deployment and wide range accessibility, SA systems face a new attack surface that may lead to security and privacy concerns. It is expected that the cyber security and data science research communities will set off on constructing advanced technologies to safeguard this critical infrastructure, e.g., data-driven protection and AI-enabled defense.
In this work, we set up an SA testbed named SATB that can facilitate SA dataset generation. SATB is designed to be extensible so that it is capable of incorporating sensors (e.g., SenseCAP sensors) and protocols (e.g., LoRaWAN) that are extensively adopted in real-world SA systems. To test the usability of SATB, we use it to create a comprehensive SA network dataset for research use. With SATB, our dataset can capture data that rigorously covers the whole lifecycle of SA scenarios, from the *authentication* stage to the *runtime functioning* stage. We design five typical test cases, and SATB can generate network traces based on them. SATB also supports generating attack traces of network reconnaissance and vulnerability scanning. We show the details of our dataset collection process on SATB and conduct a preliminary statistical analysis, to enlighten potential smart use of our testbed. The collected dataset is released online to facilitate related research: `https://github.com/UQ-Trust-Lab/2022-SATB`.

**Keywords:** Smart Agriculture · LoRaWan · Network · Testbed

## 1 Introduction

The recent booming Internet of Things (IoT) has penetrated many industries, from automotive, home automation, manufacturing to agriculture and cities. Agriculture, one of the most important activities of human civilization, is a typical area that IoT has introduced revolutions. Thanks to the digitalization technologies and innovations, we have entered an era of *smart agriculture* (SA)

or *Agriculture 4.0*. Integrating connected devices with cloud and mobile applications, SA scales up production processes while consuming less natural resources. It has reshaped those traditional processes from managing cattle, monitoring climate conditions, irrigating farms to harvesting crops, greatly enhancing automation, efficiency and sustanability. A recent study reports that the IoT-enabled SA market reached 11.20 billion USD in 2021 and is expected to reach 26.65 billion in 2030 [3].

As one of the emerging paradigms that incorporate IoT into a traditional domain, SA has demonstrated its similarity in the design of the architectures and workflows to other domains. Nonetheless, it has a wider attack surface that leads to new cyber threats that have not been well studied in other IoT systems. First, in order to enable a longer range of wireless communication, SA systems usually support LoRaWAN [4] that enables miles of long-range communications. Unlike in smart home systems where Bluetooth and ZigBee are mainly used for communications and wireless signals are restricted within a narrow and trustworthy range (e.g., 10 meters for Bluetooth), an attacker can easily capture or inject wireless traffic to gain credentials or hijack into established communications. Second, end devices in SA, e.g., sensors and actuators, are mostly deployed in open or rural areas. Unlike in smart city systems where boxes can be installed or camera surveillance can be deployed to protect them, the SA devices are exposed to physical attacks. The attacker may temper with or even remove the sensors. Third, the SA devices are also subject to extreme weather conditions (e.g., rain and lightning) and wild animals, which may render them unavailable.

Due to the rapid growth of SA, several studies have been conducted to analyze the security of SA systems. Various advanced data analysis techniques have been proposed to detect attacks from the huge amount of data [9, 21, 26, 33]. Nonetheless, there is still a lack of SA domain-specific dataset that can be used by these studies for evaluation and benchmarking. Researchers have to turn to those generic IoT or industry IoT (IIoT) datasets, e.g., UNSW-NB15 [24] and TON_IoT [23].

In this work, we set off to build up an SA testbed named SATB for collecting SA domain-specific datasets. SATB follows a widely adopted architecture, which interconnects end devices, gateways and servers with the protocols used in the real-world SA scenarios. Without losing representativeness, our testbed is constructed from equipment and cloud services that have been widely deployed for commercial use. For the communication between sensors and the servers, we deploy LoRaWAN, the *de facto* protocol used in real-world SA systesms.

SATB enables us to construct a preliminary dataset. As a case study, we design five test cases to drive the execution of SATB, and during the execution, we capture the communication traces. Through this, we come up with a dataset that covers the main stages of SA, from *authentication* to *runtime functioning*. This yields 671,239 records of network packets. With SATB, we are also able to conduct attacks and include the attack data into our dataset. As the first step, we have conducted a network reconnaissance with Nmap [5] and a vulnerability scanning with OpenVAS [7]. Both tools are typical attack tools of penetration

testers and hackers in practice. This generates another 167,090 records of attack packets.

To explore the usability of SATB, we explore the quality of the collected dataset with some preliminary studies. With simple preprocessing, the collected traces can be formatted. We can then reveal the protocols and services that packets belong to. The LoRaWAN payloads are kept in the dataset and can be decoded with off-the-shelf parsers. Key information for intrustion detection, such as the used ports and constant strings can be revealed by our dataset.

**Contributions**. In summary, our work makes the following contributions.

- **An extensible and representative testbed**. We construct a testbed named SATB with commercial SA equipment and open-source software stacks. It represents the mainstream architecture and workflow of SA systems. Our testbed is extensible that other sensors or servers can be easily plugged for simulation and dataset collection.
- **Simulation of real-world attacks**. Based on our in-house testbed, we are able to take the first step of generating attack data against SA systems. We conduct network reconnaissance and vulnerability scanning on the testbed, and the attack traces can be captured by our testbed.
- **An SA network dataset**. With SATB, we build a dataset that includes 876,371 records of network packets. It covers the communications between end sensors and the servers, and contains the data of the entire lifecycle from authentication to runtime functioning.

## 2 Background and Related Work

### 2.1 Smart Agriculture

The smart agriculture (SA) technology is a concept that makes use of advanced technologies such as Internet of Things (IoT), robotics and big data, to enhance the traditional agriculture industry. These technologies can form a network of physical objects, including sensors, softwares, or other technologies. Owing to enhanced connectivity, devices become able to communicate with gateways, servers, applications as well as other devices, exchange data and perform the tasks that they are designed for.

With the empowerment of smart agriculture technologies, the farming process has been changed dramatically. In a farming process, the SA can be divided into three stages, i.e., *data gathering*, *data analysis*, and *reaction and control*. In the first stage, the sensors deployed in the farms are used to monitor the animal activities and environment such as temperature, soil moisture and humidity. They upload the data to the cloud in real time. In the second stage, the cloud side employs data-driven techniques to analyze the data, and then corresponding actions can be taken in the reaction and control stage. This can either be a real-time "trigger and action automation", or an action taken after diagnosis. Taking watering as an example, if the sensor detects that the soil moisture level is lower than expected, the water dispensers in the farm can start to work

automatically and water the crops until the soil has reached the desired moisture level. It ensures that the crops will not be underwatered or overwatered. With SA technologies, the farmers are able to monitor the environment remotely and automate the farming process with the minimal use of labour and natural resources such as water and fertilizer.

### 2.2 LoRaWAN

LoRaWAN [4] is a communication protocol at the data link and network layers. It works on top of LoRa radio modulation technique. A LoRaWAN architecture typically consists of four components: *end nodes*, *gateway*, *network server* and *application server*.

- **End nodes.** End nodes are typically sensors and actuators. They communicate with the gateway through the LoRa protocol.
- **Gateway.** The gateway is the bridge between end nodes and the network server. It plays a role of the access point or the relay. The gateway communicates with the end nodes through LoRa radio and gathers data from them. The data is forwarded to the network server through a network protocol compatible with the server, such as Ethernet, WiFi and 3G/4G.
- **Network server.** The network server handles the communication with the gateway and conducts data preprocessing, e.g., deduplication.
- **Application server.** The application server mainly conducts data analysis and interaction with clients. It is usually installed with computational resources to handle massive data and generates intelligent response. It also provides interfaces to the client applications, such as mobile applications and web applications. The raw data and its analysis results are displayed to users, and users' management is forwarded back to the network server and eventually end nodes for actuation. In practice, the network server and application server are mostly merged into one single server.

### 2.3 Related Work

**Cyber Security Issues in SA and IoT**. A broad range of cyber security challenges brought by the interconnectivity of SA and IoT have been noticed and disccussed [8,12,15,19,28–30]. Various threats have been revealed, including hardware/firmware-specific attacks [13, 14], network-layer attacks [11, 20], and cloud-specific attacks [18]. It has been demonstrated that the cyber security issues in SA have attracted attention and posed threat and damages if unsolved. Various studies have also been proposed to migitate the cyber risks at different IoT layers, from device layer [16, 32], edge layer [25] to cloud layer [31].

**IoT-related Testbeds and Datasets**. Sivanathan et al. [27] build up a smart environment with a variety of IoT devices, including spanning cameras, lights, plugs, motion sensors, appliances, and health-monitors. Alsaedi et al [10] construct a TON_IoT dataset, which includes telemetry data of IoT/IIoT services, operating system logs and network traffic of IoT network. Compared to these

studies, SATB mainly focuses on SA scenarios and reflects the representative workflow and working procedures. Specific analysis and data processing are also performed on the traffic. For example, LoRaWAN packets are decoded and attributes are extracted from them.

Recently, Gondchawar et al. [17] proposes a IoT-based SA framework, which consists of three end nodes and a PC to control the system. The end nodes are AVR Microcontroller Atmega 16/32, ZigBee Module, and Temperature Sensor LM35. Their testbed uses Dig Trace, SinaProg and Raspbian Operating System as the software stack. Compared to their work, the devices in our testbed are more up-to-date and can better reflect the current SA systems in practice.

## 3 SATB: a LoRaWAN SA Testbed

### 3.1 Components of SATB

We have built up a LoRaWAN Smart Agriculture Testbed named SATB to facilitate data collection. SATB consists of three major components: *end devices*, *gateway* and *server*, as shown in Fig. 1.

**End devices and gateway**. We deploy two SenseCAP sensors and SenseCAP gateway, as listed in Table 1. The sensors monitor the soil moisture, temperature and light intensity, and report the readings to the gateway through LoRa radio.

**Server**. The server our testbed uses is called ChirpStack [2], which is a LoRaWAN network server stack. It consists of the *gateway bridge*, *network server* and *application server*, which merge the modules discussed in Section 2.2 into one server. The gateway bridge handles the communication with the LoRaWAN gateway and converts the LoRa packets into ChirpStack Network Server compatible data format. The network server then de-duplicates the frames received by the LoRa gateways and forwards them to Application Server. The application server plays the role of the "inventory" of the LoRaWAN infrastructure. It offers a web interface for users to manage, view and organize the stored data.

### 3.2 Functionalities of SATB

Each end devices and the gateway have an identifier called Extended Unique Identifier (EUI), which is delivered to the end user with their packages. For a device to be recognized by ChirpStack server, its EUI has to be registered with the latter. This can be done through the web portal provided by the application server. For a device to be connected to the network server, the correct *app key* should be provided upon authentication.

The testbed is deployed in the Device Test Lab on UQ campus. The sensors are connected to the gateway wirelessly while the gateway is connected to the Internet via Ethernet. A Dell desktop installed with the ChirpStack network server is connected to the same Ethernet. Three Docker containers are deployed in the desktop, each of which hosts one module of the ChirpStack network server. They communicate with each other through Docker's single-host networking. We install a packet logger named Wireshark [1] on the desktop to capture all traffic relayed by the gateway. It also can capture the traffic among the three modules.

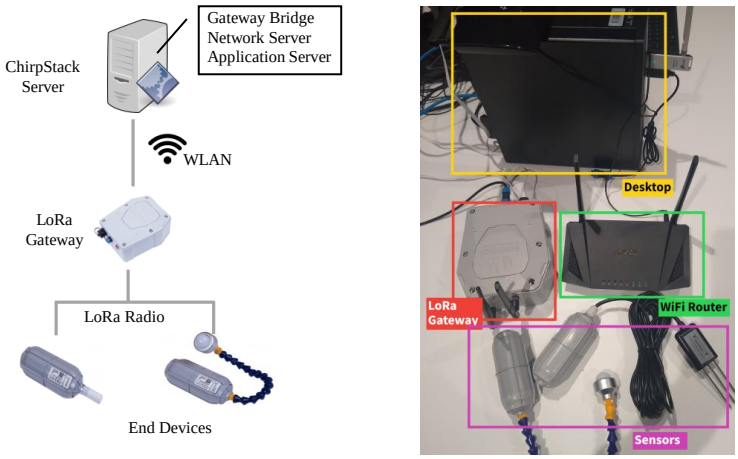


Fig. 1: The architecture and a photo of our testbed

Table 1: The IoT devices in our testbed

| IoT Device | Device Type |
|---|---|
| SenseCAP Outdoor Gateway | Gateway |
| SenseCAP Soil Moisture & Temp Sensor | End device |
| SenseCAP Light Intensity Sensor | End device |

## 4 A Case Study: Constructing an SA Dataset with SATB

To demonstrate the usablity of SATB, we use it to construct a preliminary dataset. It includes SA network traces in normal operation and attack scenarios.

### 4.1 Test Cases and Data Collection

To retain representativeness, we propose five application scenarios to make our dataset. For each scenario, we design a test case to drive the execution and communication of each component in our testbed. Below, we describe our test cases (TCs).

- **TC1.** When the gateway and the sensors are not registered with ChirpStack server, we connect the gateway and the sensors to the network server. In this test case, the gateway and sensors cannot be recognized by the network server.
- **TC2.** When the gateway and the sensors are not registered, we register the gateway, connect the gateway to the network server, and then connect the sensors to the network server. In this test case, the gateway can be connected but the sensors are not recognizable.

- **TC3.** When the gateway is registered and the sensors are not, we connect the gateway to the network server, register the sensor with correct EUI but with incorrect *app key*, and then connect the sensor to the network server. In this test case, the gateway can be connected but the sensors cannot.
- **TC4.** When the gateway is registered and the sensors are not, we register the sensors, connect the gateway to the network server and connect the sensors to the network server. In this test case, both the gateway and the sensors can be connected to the network.
- **TC5.** When the gateway and the sensors are both registered, we connect the gateway and the sensors to the network server. In this test case, both the gateway and the sensors can be connected to the network.

To test SATB comprehensively, we also create attack scenarios on top of the use cases. As the first step, we consider two types of attacks - reconnaissance and vulnerability scanning. For the former, we apply a widely used network scanner called Nmap [5] to scan against our testbed, with different scanning methods such as SYN Scanning and TCP Connect Scanning. For the latter, we use a popular vulnerability assessment tool OpenVAS [7] to conduct a vulnerability scanning. It has the signature of most known software vulnerabilities and checks whether the scanned system is subject to those vulnerabilities. We turn on Wireshark to log the network traffic while we execute the test cases and attacks. Filters are set to ensure that only the relevant packets are collected. The packets are saved as *.pcap* files.

### 4.2 Data Preprocessing

After the data collection, we further process the data. We parse each packet in the *.pcap* file to extract the features, including source ip address, destination ip address, source port, destination port, protocol information and other protocol-specific fields depending on the protocol of the packet. For example, if a packet is an MQTT packet, MQTT-specific fileds, such as the MQTT message type, remaining length, and payload field are included. After the processing, the extracted fields of the packet are exported to a *.csv* file. The main features of our dataset are listed in Table 2.

During the data process, we also apply a tool called Zeek (previously Bro-IDS) [6] on the captured packets. It takes the *.pcap* file as input and turns it into high-fidelity transaction logs. This tool is also able to find unusual and suspicious behaviours, which will be used to confirm the label of the packets captured in the attacking stage.

### 4.3 A Preliminary Study of the Dataset

In this section, we present a preliminary study of the collected dataset and give its basic information.

Table 2: Main features of our dataset

| | Name | Description |
|---|---|---|
| 1 | src_ip | The source ip address of the packet |
| 2 | dst_ip | The destination ip address of the packet |
| 3 | src_port | The source port of the packet |
| 4 | dst_port | The destination port of the packet |
| 5 | protocol | The protocol-specific information of the packet |
| 6 | other | Other information about the packet |

**Packets Distribution**.

Overall, we have collected 876,371 packets, which take around 1.5 GBytes. They mainly cover three working stages, i.e., *authentication stage*, *runtime functioning stage*, and *attacking stage*. The majority of the packets are in runtime use (671,239 packets). The authentication stage and attacking stage have 38,042 and 167,090 packets respectively.

***Authentication.*** There are two types of authentication in LoRaWAN: Over The Air Activation (OTAA) and Activation By Personalisation (ABP). The one used by our testbed is OTAA, which is more extensively used in practice than ABP. In OTAA activation, each time when the end devices intend to join, the session keys are different. When an end device attempts to join the network, it sends a *join-request* message which contains AppEUI, DevEUI, DevNonce and Message Integrity Check (MIC) value. The network server processes this message, and then generates two session keys and a *join-accept* message if the end device is permitted to join. The *join-accept* message is then encrypted using the *app key* and sent back to the end device as a normal downlink. If the join request is not accepted, no response will be replied.

***Runtime functioning.*** After the authentication, the end devices and the gateway are connected to the network. The testbed starts operating. The readings of the sensors are periodically reported to the network server, through the relay of gateway and gateway bridge. The update period can be configured through the web portal of the application server. The network server then dedupliates the received packets and formats them so that they can be usable by the application server.

***Attacking.*** The following scanning traces are included in the collected dataset.

1. Nmap SYN Scan: It scans the ports of a target machine by sending a TCP packet with the SYN flag set.
2. Nmap TCP Connect Scan: It scans the ports of a target machine by establishing a full connection.
3. Nmap Service and Version Scan: It detects more information about the version of the services running on the target machine.
4. Nmap Operating System Scan: It detects the operating system of the target machine by sending a series of TCP, UDP packets and examining the response.

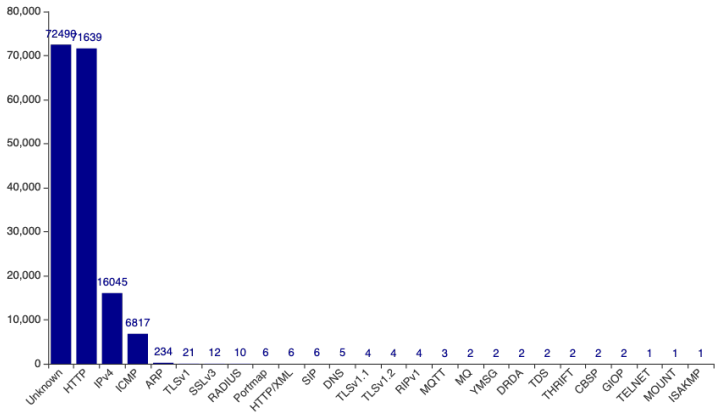


Fig. 2: Protocol types in the authentication stage

Table 3: An example of a decoded LoRaWAN payload

| Feature | Description | Decoded Value |
|---|---|---|
| Message Type | MAC message types | `Join Request` |
| PHYPayload | LoRa Physical Payload | MHDR[1] — MACPayload[..] — MIC[4] |
| MHDR | MAC Header | 02 |
| MACPayload | MAC Payload of Data Messages | 8DC0022CF7F110 |
| MIC | Message Integrity Code, which ensures the integrity of the packet | 24400040 |
| MACPayload | MAC Payload of Data Messages | AppEUI[8] — DevEUI[8] — DevNonce[2] |
| AppEUI | 64-bit unique identifier for Join Server | 2410F1F72C02C08D |
| DevEUI | 64-bit globally-unique Extended Unique Identifier (EUI-64) assigned by the manufacturer | 400040 |
| DevNonce | A random number | NA |

5. OpenVAS full and deep scanning: It performs a full and deep vulnerability scanning against the target machine.

**Packets of Authentication Stage**.

We attempt to recognize the service types of the packets collected at the authentication stage. We focus mainly on the application-layer protocols and the statistics as shown in Fig. 2. The majority of packets in the authentication stage is PGSQL, which is mainly used for PostgreSQL, an open-source relational database management system. It is integrated into the ChirpStack network server stack to store device-related events. Since all events are written into the database, the packets of this type become dominant.

Many packets have unknown types and they are shown as TCP or UDP packets. Almost all UDP packets contain LoRaWAN payloads that are not recognized by Wireshark. However, the LoRa packets can actually be extracted from them. For example, one of the UDP packets that we have captured contains “028dc0022cf7f11024400040” as its payload, which is hex encoded. We use a LoRa packet decoder to decode the payload and the attributes can be extracted as shown in Table 3.


(a) Authentication stage (b) Runtime stage

Fig. 3 stage

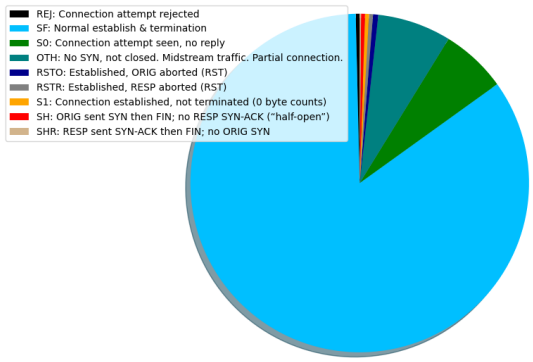


Fig. 4: The Distribution of connection state in authentication stage

We also conduct an analysis on the source ports of the packets. In Fig. 3a, we use a word cloud to present the occurrence of ports in the dataset. If a port appears more frequently, its size in the graph is larger. We can observe that the ports 56062, 56064, and 51620 account for a higher proportion in the captured packets. They are mainly used by TCP.

We also analyze the connection states from the packets. As shown in Fig. 4, around 84% of the packets has a connection state of `SF`, which means the connection is established and terminated normally. `OTH` and `S0` also have a higher percentage than the rest of the packets. The state `OTH` occurs when there is no SYN seen in the connection, it might because it is a midsteam traffic. 6% of the packets have `S0` as their connection state, which means that a connection has been attempted, but no reply is seen.

**Packets of Runtime Functioning Stage**.
Similarly, we attempt to recognize the service types of packets and the ports used in the runtime functioning stage. The results are shown in Fig. 5 and Fig. 3b.

**Packets of Attacking Stage**.
We first attempt to recognize the service types of packets in the attacking stage. The statistics is shown in Fig. 6. We can notice that the service of 43.24% of the packets are unknown (tagged as UDP and TCP packets), while 42.87% of the packets are HTTP packets. After looking into the packets, the high percentage of HTTP packets is due to the vulnerability scanning of OpenVAS. When this

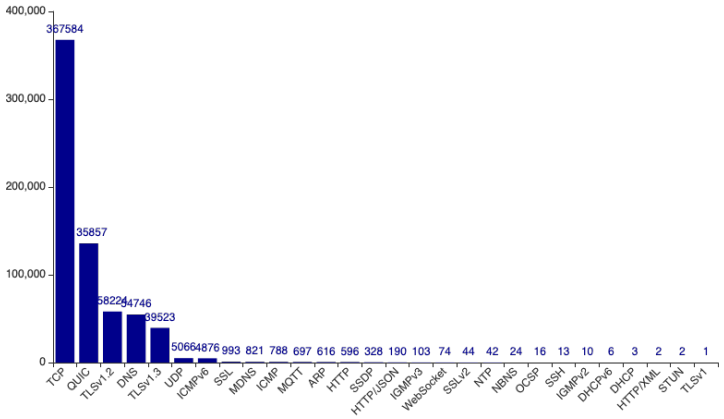


Fig. 5: Protocol types in runtime stage

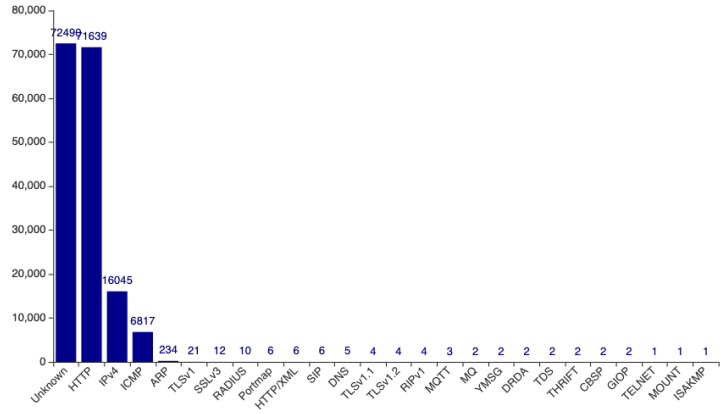


Fig. 6: Protocol Types in the Attacking Stage

tool conducts scanning, it requests a number of web locations that may contain security issues such as SQL injection or buffer overflow. The word cloud of scanned web locations is shown in Fig. 7.

We also use Zeek to process the packets captured in attacking stage. There are 28,806 out of 167,090 attacking packets that are detected unusual or improper behaviors and flagged. The flags can be grouped to a list of 21 flags in total as shown in Table 4. The most frequent flags that are extracted are `possible_split_routing`, `data_before_established` and `inappropriate_FIN`. The `possible_split_routing` happens when the other side of the connection is not seen. The `data_before_established` means before a connection was established, some data is sent by a TCP endpoint. The `inappropriate_FIN` happens when a FIN set in the packet does not follow the RFC for TCP/IP standard.


Fig. 7: Word Cloud of Scanned Web Locations

Table 4: The list of attack flags

| Features | |
|---|---|
| possible_split_routing | DNS_Conn_count_too_large |
| inappropriate_FIN | line_terminated_without_CRLF |
| data_before_established | unknown_HTTP_method |
| bad_HTTP_request | unescaped_%_in_URI |
| empty_http_request | partial_escape_at_end_of_URI |
| line_terminated_with_single_CR | data_after_reset |
| DNS_truncated_len_lt_hdr_len | premature_connection_reuse |
| DNS_label_forward_compress_offset | HTTP_version_mismatch |
| DNS_label_too_long | bad_HTTP_request_with_version |
| DNS_truncated_RR_rdlength_lt_len | double_%_in_URI |
| missing_HTTP_uri | |

## 5 Usage of the SATB Testbed

In this section, we outline a few usage of the SATB Testbed, towards the reliable operation in a smart agriculture ecosystem.

### 5.1 Development of Intrusion Detection Systems for SA

The rapid development and adoption of highly diverse IoT devices in smart agriculture have created increased attack surface. Therefore, building an intrusion detection system (IDS) has become an increasingly urgent issue for detecting network security breaches against SA systems. However, as the performance of an effective IDS largely relies on the availability of representative and real-life data, the lack of SA domain-specific datasets has resulted in a significant gap in the IDS development.

Our high-fidelity SATB Testbed can alleviate the gap by providing intensive sets of traffic traces for training and testing IDSs. SATB covers the complete workflow of smart agriculture system ranging from authentication to runtime usage, which represents the sum of all possible attack vectors that may be exploited. In addition, the construction of the SA dataset has demonstrated the capacity of SATB in incorporating a variety of real network intrusion scenarios against SA systems, not only reflecting the existing attacks of the time, but are also dynamic and extensible as network scenarios evolve.

### 5.2 Preservation of Data Privacy and Integrity for SA

The most significant feature of smart agriculture is its employment of a vast amount of sensors, devices and equipment, from which an enormous amount of data gets generated. The challenge of preserving data privacy and integrity remains open in this area due to the massiveness, complexity and heterogeneity of the generated data [19, 22].

Our SATB Testbed, which serves as the representative digital twins of the counterparts in the real-world production environment, can greatly facilitate the research in the key properties of data protection from the following perspectives: (1) the secure sharing of data with different sensitivity classifications across stakeholders in the entire supply chain, (2) the secure processing of sensitive data (e.g., agriculture purchase information) with provable privacy-preservation and verifiable prevention of information leakage, (3) the provenance based data integrity checking and verification in the smart agriculture environment.

### 5.3 Development of Data-driven Applications for SA

We see the opportunities of SATB contributing to the research area in integrating artificial intelligence and machine learning technologies for the development of data-driven applications in smart agriculture. For example, the massive real-life data generated by our testbed (including time series and spatial data) can be used to train and evaluate real time monitoring systems, which are crucial to modern farms in improving their productivity and security. In addition, despite the advantages of LoRaWAN (such as low-power and long-range transmissions which are typically desirable in the smart agriculture environment), the technology has the disadvantage of low-bandwidth transmissions, which has presented an arising need to integrate edge computing to facilitate the move of data processing and analytics to end devices. Our testbed can be used to construct and test such computing framework with data on edge, extending the functionalites of smart agriculture applications using LoRaWAN for wide area coverage.

## 6 Conclusion

In this work, we have designed and developed an SA testbed named SATB to facilitate simulation, attack experiments and dataset generation. SATB consists of the end devices, gateways, software stacks and protocols that are widely adopted in real-world SA systems. It is designed to be extensible so that new hardware and software modules can be installed. To show SATB's usability, we used it to construct a comprehensive SA network dataset, which rigorously cover the whole lifecyle of SA scenarios. For future works, we will enrich the types of the end sensors to support other protocols. We will also embed an open gateway that enables the interception of the LoRa packets between the end sensors and the gateway.

**Acknowledgment**. This work is supported by The University of Queendland under the Cyber Research Seed Funding.

## References


1. Analysis tool wireshark. https://www.wireshark.org/ (March 2022)
2. Chirpstack. https://www.chirpstack.io (March 2022)
3. Internet of things in agriculture market. https://www.emergenresearch.com/industry-report/iot-in-agriculture-market (March 2022)
4. Network scanner nmap. https://lora-alliance.org/about-lorawan/ (March 2022)
5. Network scanner nmap. https://nmap.org/ (March 2022)
6. Network security monitoring tool zeek. https://zeek.org/ (March 2022)
7. Scanner openvas. https://www.openvas.org/ (March 2022)
8. Abuan, D.D., Abad, A.C., Lazaro Jr, J.B., Dadios, E.P.: Security systems for remote farm. Journal of Automation and Control Engineering Vol **2**(2) (2014)
9. Al-Hawawreh, M.S., Moustafa, N., Sitnikova, E.: Identification of malicious activities in industrial internet of things based on deep learning models. J. Inf. Secur. Appl. **41**, 1–11 (2018)
10. Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., Anwar, A.: Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems. IEEE Access **8**, 165130–165150 (2020)
11. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al.: Understanding the mirai botnet. In: 26th USENIX security symposium (USENIX Security 17). pp. 1093–1110 (2017)
12. Barreto, L., Amaral, A.: Smart farming: Cyber security challenges. In: 2018 International Conference on Intelligent Systems (IS). pp. 870–876. IEEE (2018)
13. Cojocar, L., Zaddach, J., Verdult, R., Bos, H., Francillon, A., Balzarotti, D.: Pie: Parser identification in embedded systems. In: Proceedings of the 31st Annual Computer Security Applications Conference. pp. 251–260 (2015)
14. Costin, A., Zaddach, J., Francillon, A., Balzarotti, D.: A {Large-Scale} analysis of the security of embedded firmwares. In: 23rd USENIX Security Symposium (USENIX Security 14). pp. 95–110 (2014)
15. Demestichas, K., Peppes, N., Alexakis, T.: Survey on security threats in agricultural iot and smart farming. Sensors **20**(22), 6458 (2020)
16. Feng, X., Sun, R., Zhu, X., Xue, M., Wen, S., Liu, D., Nepal, S., Xiang, Y.: Snipuzz: Black-box fuzzing of iot firmware via message snippet inference. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 337–350 (2021)
17. Gondchawar, N., Kawitkar, R., et al.: Iot based smart agriculture. International Journal of advanced research in Computer and Communication Engineering **5**(6), 838–842 (2016)
18. Gruschka, N., Jensen, M.: Attack surfaces: A taxonomy for attacks on cloud services. In: 2010 IEEE 3rd international conference on cloud computing. pp. 276–279. IEEE (2010)
19. Gupta, M., Abdelsalam, M., Khorsandroo, S., Mittal, S.: Security and privacy in smart farming: Challenges and opportunities. IEEE Access **8**, 34564–34584 (2020)
20. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: Ddos in the iot: Mirai and other botnets. Computer **50**(7), 80–84 (2017)
21. Koroniotis, N., Moustafa, N., Sitnikova, E.: A new network forensic framework based on deep learning for internet of things networks: A particle deep framework. Future Gener. Comput. Syst. **110**, 91–106 (2020)

22. Mahadewa, K., Zhang, Y., Bai, G., Bu, L., Zuo, Z., Fernando, D., Liang, Z., Dong, J.S.: Identifying privacy weaknesses from multi-party trigger-action integration platforms. In: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 2–15 (2021)
23. Moustafa, N.: A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. Sustainable Cities and Society **72**, 102994 (2021)
24. Moustafa, N., Slay, J.: 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). pp. 1–6 (2015). https://doi.org/10.1109/MilCIS.2015.7348942
25. Nesarani, A., Ramar, R., Pandian, S.: An efficient approach for rice prediction from authenticated block chain node using machine learning technique. Environmental Technology & Innovation **20**, 101064 (2020)
26. Popoola, S.I., Adebisi, B., Hammoudeh, M., Gui, G., Gacanin, H.: Hybrid deep learning for botnet attack detection in the internet-of-things networks. IEEE Internet of Things Journal **8**(6), 4944–4956 (2021). https://doi.org/10.1109/JIOT.2020.3034156
27. Sivanathan, A., Gharakheili, H.H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., Sivaraman, V.: Classifying iot devices in smart environments using network traffic characteristics. IEEE Transactions on Mobile Computing **18**(8), 1745–1759 (2018)
28. Xie, F., Zhang, Y., Wei, H., Bai, G.: Uq-aas21: A comprehensive dataset of amazon alexa skills. In: International Conference on Advanced Data Mining and Applications. pp. 159–173. Springer (2022)
29. Xie, F., Zhang, Y., Yan, C., Li, S., Bu, L., Chen, K., Huang, Z., Bai, G.: Scrutinizing privacy policy compliance of virtual personal assistant apps. In: 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22) (2022)
30. Yazdinejad, A., Zolfaghari, B., Azmoodeh, A., Dehghantanha, A., Karimipour, H., Fraser, E., Green, A.G., Russell, C., Duncan, E.: A review on security of smart farming and precision agriculture: Security aspects, attacks, threats and countermeasures. Applied Sciences **11**(16), 7518 (2021)
31. Zhang, H., Lu, K., Zhou, X., Yin, Q., Wang, P., Yue, T.: Siotfuzzer: fuzzing web interface in iot firmware via stateful message generation. Applied Sciences **11**(7), 3120 (2021)
32. Zheng, Y., Davanian, A., Yin, H., Song, C., Zhu, H., Sun, L.: {FIRM-AFL}:{High-Throughput} greybox fuzzing of {IoT} firmware via augmented process emulation. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 1099–1114 (2019)
33. Zhou, X., Hu, Y., Liang, W., Ma, J., Jin, Q.: Variational lstm enhanced anomaly detection for industrial big data. IEEE Transactions on Industrial Informatics **17**(5), 3469–3477 (2021). https://doi.org/10.1109/TII.2020.3022432