Security Analysis of Large Scale IoT Network for Pollution Monitoring in Urban India

by

Ihita G.V., Viswanadh K.S., Sudhansh Y, Sachin Chaudhari, Gaur S

Report No: IIIT/TR/2021/-1

Centre for Communications
International Institute of Information Technology
Hyderabad - 500 032, INDIA
June 2021

Security Analysis of Large Scale IoT Network for Pollution Monitoring in Urban India

G.V. Ihita[1], K.S.Viswanadh[1], Y. Sudhansh[1], S. Chaudhari[1], S. Gaur[2]
[1]*Signal Processing and Communication Research Centre*
International Institute of Information Technology - Hyderabad (IIIT-H), India
[2]*MixORG, Noida, India*
{ihita.g, savitha.viswanadh}@research.iiit.ac.in, sudhansh.yelishetty@students.iiit.ac.in,
sachin.chaudhari@iiit.ac.in, sachin.gaur@mixorg.com

***Abstract*—The surge in the development and adoption of Internet of Things (IoT)-enabled smart city technologies has brought with it a diverse set of critical security challenges. In this paper, protocol and network security threats pertaining to a large-scale IoT-enabled pollution monitoring sensor network, AirIoT, deployed in and around an educational campus in the Indian city of Hyderabad, have been explored. Using the STRIDE methodology, the paper assesses various threat vectors for the deployment. As solutions, the paper proposes an approach for end-to-end encryption, protocol and dashboard security, and a proof of concept deauthentication detector. This baseline threat analysis and risk assessment can provide a foundation for securing Wi-Fi and mobile network-based large-scale IoT deployments.**

***Index Terms*—IoT security, Large-scale deployment, Pollution monitoring, Threat analysis**

I. Introduction

Internet of Things (IoT) has applications in various sectors such as health, education, and energy. An important use case of IoT technology is enabling smart cities initiatives for efficient urban planning and delivery of services, including smart waste management, traffic management, and air quality monitoring.

With the rapid adoption of IoT-enabled smart city technologies, security becomes a critical component to enhance confidence and trust among citizens and the governing bodies. Further, there are challenges of fragmented security standards and the non-homogeneity of IoT use cases. Covering both physical and network security, the security of IoT systems protect against attacks that compromise the confidentiality, integrity, and availability (CIA triad)[1] of data and services. RFC 8576 [1] categorizes the IoT security threats and the risks associated with these threats. The document also describes the challenges with securing IoT devices, the network, and the respective trade-offs of these resource-constrained devices. Further, it briefly explains the role of the global standardization bodies such as the European Union Agency for Network and Information Security (ENISA), Cloud Security Alliance (CSA), and Global System for Mobile Communications Association (GSMA). In [2], ENISA identifies and analyzes existing IoT security practices, security guidelines, relevant industry standards, and research initiatives. Key asset groups and the criticality of these assets are identified through the asset taxonomy. Additionally, threats and the effect of the threats on these assets are mentioned in the threat taxonomy. Finally, based on the security measures developed and challenges identified, security recommendations are proposed.

Air quality monitoring systems monitor particulate matter, humidity, temperature, and the presence of airborne chemicals. For large-scale deployments, the vulnerabilities and risks include eavesdropping, denial-of-service (DoS) attacks, network outages, weak passwords, data tampering, and physical tampering. Establishing resilient, reliable, and secure communication between nodes and the cloud requires analyzing the network for vulnerabilities and work on solutions to secure them, which is the focus of this paper. There have been few works in the literature on IoT security for air pollution networks [3, 4]. In [3], different security vulnerabilities such as unencrypted message communication, inefficient authentication mechanisms, and lack of data integrity verification are demonstrated in "a.com", a low-cost air quality monitoring system. Large scale deployments are exposed to such attacks and to man-in-the-middle (MITM) attacks performed by adversaries to sniff/modify data, thus compromising data privacy and confidentiality. In [4] the authors discuss data security and consistency of data sent in pollution monitoring deployment for smart cities. The criticality of secure communication protocols is mentioned, with a focus on the use of secure MQTT.

This paper presents network and protocol-level vulnerability assessment for an existing large-scale, low-cost pollution monitoring sensor network, AirIoT [5], deployed in urban India. AirIoT is an extended work of [6]. Attempts have been made to capture, analyze, intercept and modify the transmission between the backend and the device to affect the CIA components of data. This includes unauthorized access to resources, DoS, and protocol level breaches. The novelty of the work is brought out through:

- The vulnerability assessment of the three different IoT communication networks, implemented in AirIoT, performed using STRIDE [7]. STRIDE methodology is used to identify potential security threats to a system as

[1]The CIA triad security model. URL: https://www.f5.com/labs/articles/education/what-is-the-cia-triad

TABLE I
STRIDE METHODOLOGY FOR THREAT MODELLING

Threats	Description
Spoofing	Clone/impersonate a sensor node to gain illegitimate access to resources violating authentication
Tampering	Physical tampering of sensor nodes and data tampering to violate data integrity.
Repudiation	Compromising the proof of origin and validity of data violating non-repudiation
Information Disclosure	Disclosing data to unauthorized user violating confidentiality
Denial of Service	Affecting the availability of data and services
Elevation of Privilege	Obtain more privileges by spoofing a user thus violating authorization

described in Table I.

- Analysis of the different communication protocols used in the deployment.
- Threat assessment for the dashboard, which is the only point of access away from the physical deployment.
- Proof-of-concept solutions to the vulnerabilities exposed in AirIoT, which are extensible to general large-scale sensor networks.

The rest of the paper is structured as follows. Section II briefly presents the large-scale AirIoT deployment. Section III presents the approach and the results of the vulnerability assessment of networks. Solutions and recommendations are presented in Section IV followed by the conclusion in Section V.

II. Deployment Scenario

In the process of reaching the milestone of 50 node deployments, there are currently 30 nodes deployed, as shown in fig.1. The nodes are placed in and around an Indian educational institute, International Institute of Information Technology - Hyderabad (IIIT-H). The initial deployment of 10 nodes has been developed further to accommodate new nodes and better network technologies, giving rise to a more robust system. Each sensor node consists of ESP8266 microcontroller (NodeMCU), SDS011 sensor to monitor particulate matters (PM2.5 and PM10) and sensors to monitor temperature and humidity. The sensor nodes send sensed data to the ThingSpeak server, an open source IoT platform, [8] for further analysis via the following networks:

- Network N_1: Nodes connected to IIIT-H campus network via Wi-Fi for communication
- Network N_2: Nodes connected via 4G hotspot JioFi[2]
- Network N_3: Nodes connected via embedded SIM[3]

Fig.2 shows the three types of communication networks sending data to ThingSpeak server.

[2]JioFi 4G hotspot router. URL: https://www.jio.com/shop/en-in/router-jmr540-black-/p/491193576

[3]QoSIM. URL: https://sensorise.net/products/qosim-m2m-connectivity/qosim/

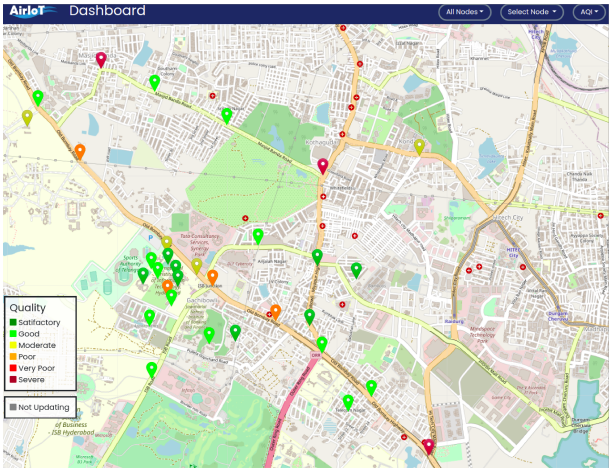

Fig. 1. Deployment of sensor nodes

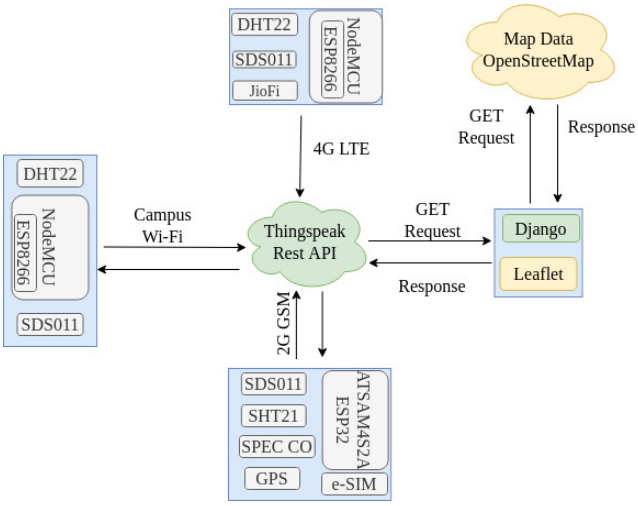

Fig. 2. Data flow diagram of the air pollution monitoring sensor nodes

Of the 30 nodes, ten sensor nodes are placed inside the campus, and 20 are placed outside. Inside the campus, 7 nodes use network N_1 and 3 use network N_2. Of the 20 nodes placed outside, 19 nodes are placed on traffic surveillance towers to avoid physical tampering of the nodes and use network N_3 for communication. One node uses network N_2. All these nodes send data to ThingSpeak.

III. Vulnerability Assessment

This section presents the vulnerability assessment performed on all three networks, the communication protocols used and, the dashboard.

A. Network N_1

Separate access points (AP) have been created for the IoT sensor nodes with low uplink data rate and MAC binding to publish sensor readings to ThingSpeak. The campus network is on a private virtual LAN with internet access allowed only to MAC-bound devices.

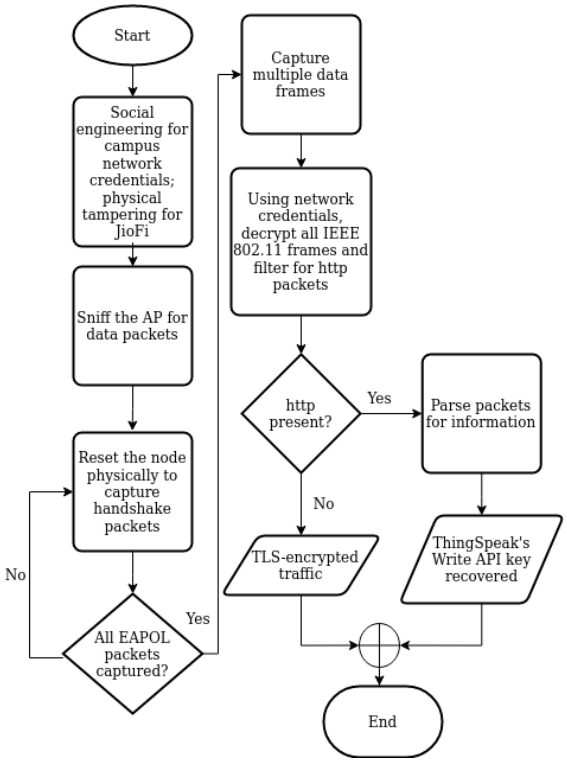

Fig. 3. Packet Pipeline attack

After attempting different attacks, the process narrowed down to two major attack scenarios: packet pipeline, our own creation, and the deauthentication of the node from the network. The packet pipeline is used to recover the Write API key for ThingSpeak to tamper data, thus violating integrity. Deauthentication is a type of DoS attack that disrupts the connection between the client and Wi-Fi AP. It was implemented to impede the availability of data and can be used to tamper data.

1) Packet Pipeline: The packet pipeline involves sniffing communication packets between the sensor node and ThingSpeak. Fig.3 shows the packet pipeline process. The tool airodump-ng [4] is used in proximity to the node to sniff the networks to gather information on the AP, such as its MAC address, basic service set identifiers (BSSID), and the channel ID. It is important to capture the Extensible Authentication Protocol (EAP) over LAN (EAPoL) packets, or the 4-way handshakes, between the node and the AP. All the frames and packets encrypted using the IEEE 802.11 protocol can then be decrypted. The decryption is performed assuming that the network credentials are already available, either by password cracking (for weak passwords) or social engineering. N_1 uses Advanced Encryption Standard (AES)-based WPA2 encryption making it resistant to statistical methods of cracking.

Another vulnerability was discovered in the pipeline. The "proof of origin" of data from sensor nodes is not maintained

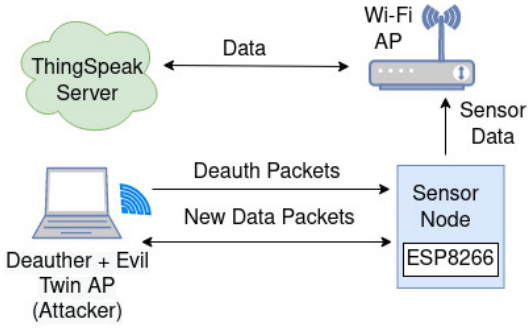

Fig. 4. Setup for the Evil-Twin attack

or verified in any part of the pipeline making the deployment vulnerable. Nodes can be spoofed into a "virtual node" to push garbage values to the server continuously.

2) Deauthentication Attack: Warwalking attack was performed using kismet [5] to find the MAC address of the node connected to the network at the nearest AP, since proximity is a major factor in this attack. Deauthentication attack was performed using this MAC address. It was carried out specifically for the node by inputting a blacklist to the tool MDK3. MDK3 [6], a proof of concept tool used for stress testing 802.11 networks, was used to execute the deauthentication attack. MDK3 uses a blacklist of device MAC addresses to send deauthentication packets to, making it unrecognizable by the network. The node was then bumped off the network and unable to push data during the interval in which the attack was live, hindering data availability.

3) Miscellaneous Attacks: In the evil twin attack, a fake AP is created for the node to connect to, as described in fig.4. This attack makes it possible to read traffic in real-time and gain information about the Write APIs to ThingSpeak. It is essential to note the extent to which such MITM attacks can go. Attacks such as packet injection and replay attacks can be performed on a given network.

Another significant suite of attacks attempted to test the resilience of the network was downgrade attacks. WPA2-secured networks can only be breached using brute-force or dictionary attacks, whereas WEP-secured networks can be cracked through statistical methods. Thus, WPA2 security is recommended. The attempt of degrading WPA2 to WEP is highly impractical in the current situation. In addition, three common vulnerabilities and exposures (CVEs) [9] were explored on the deployment. It is notable that the network is a personal network and not an enterprise one, due to which CVE-2019-12587 and CVE-2019-12586 are not detrimental to the deployment. CVE-2019-12588 was circumvented because of the more secure updated software development kits (SDKs) used in the deployment.

[4]Airodump-ng tool. URL: https://www.aircrack-ng.org/doku.php?id=airodump-ng

[5]Kismet tool. URL: http://www.kismetwireless.net/

[6]MDK3 tool. URL: https://tools.kali.org/wireless-attacks/mdk3

```
00:16:39.168 -> .........WiFi connected
00:16:43.167 -> Attempting MQTT connection...failed, rc=-4 try again in 5 seconds
00:17:04.926 -> Attempting MQTT connection...connected
00:17:14.714 -> Attempting MQTT connection...connected
00:17:31.095 -> Attempting MQTT connection...failed, rc=-4 try again in 5 seconds
00:17:52.425 -> Attempting MQTT connection...connected
```

Fig. 5. Effect of StealCon Attack

B. Network N_2

For this network, spoofing, physical tampering and DoS were performed on the interface between the ESP8266 and JioFi's Wi-Fi module. Like N_1, the deauthentication attack was successful in disconnecting the node, affecting data availability. Additionally, default passwords, a common vulnerability, were exploited to access the credentials to decrypt the captured packets through our packet pipeline. Default passwords to access the JioFi's network are available on the JioFi router. The sniffed packets were TLS-encrypted, and thus the confidentiality and integrity of the messages were maintained.

C. Network N_3

The sensor nodes connected to this network have 2G GSM based embedded SIM cards. Lack of authentication and encryption in 2G network results in attacks such as rogue base station attacks (IMSI catcher)[10]. This will be covered in our future works.

1) Analysis of the MQTT Protocol: In our work, protocol-level security was explored for the GSM based nodes. These sensor nodes use MQTT protocol to send data from the ESP8266 to the ThingSpeak server. MQTT protocol does not provide data integrity thus, attackers can see the payload, topic name, source, and destination IP, and the port number by sniffing the network using tools such as Wireshark. MQTT relies on Transmission Control Protocol (TCP). By default, TCP connections do not use encrypted communication.

2) StealCon Attack: Analysis of the protocol led to the discovery of a vulnerability scenario that we named StealCon. Through StealCon, an attacker can "steal" the connection away from the main node, performing a DoS attack. Each device connected to a host should have a unique 'Username'. When a new device with the same username as an already connected device gets connected, the original device gets disconnected. When the disconnected device tries to reconnect, already connected devices get disconnected. This cycle continues till one of the devices compromises and disconnects. The effect of this attack is shown in fig.5. This attack was tested using 2 clients - Virtual client from PC (MQTTX) and a NodeMCU initialized with the same usernames and host/port with host server mqtt.thingspeak.com configured to port 1883. When it tries to connect to the host, each node disconnects the other node, ultimately disrupting the flow of data to the server from both sources.

D. Analyzing the Dashboard

While in the process of understanding the communication between ThingSpeak and the dashboard, it was discovered that the backend of the dashboard was running on Asynchronous JavaScript and XML (AJAX) [11]. After a simple web crawl on the website directory, the JavaScript scripts used to ping ThingSpeak were discovered. In one of these scripts, the Read API keys of the ThingSpeak server, to which the nodes were pushing data, were clearly visible. From here, the data of one of the nodes was directly obtained in the response of a simple GET request on Postman, as shown in fig.6. The channel ID in the Postman response has been blurred to maintain confidentiality.


```
"channel": {
    "id": [redacted],
    "name": "Library AQ-05-12111",
    "description": "Near Library entrance",
    "latitude": "0.0",
    "longitude": "0.0",
    "field1": "Temperature",
    "field2": "Humidity",
    "field3": "PM2.5",
    "field4": "PM10",
    "field5": "CO",
    "field6": "NO2",
    "field7": "NH3",
    "created_at": "2019-12-23T15:20:16Z",
    "updated_at": "2021-01-25T10:19:50Z",
    "last_entry_id": 4762286
},
"feeds": [
    {
        "created_at": "2021-03-04T03:47:50Z",
        "entry_id": 4762187,
        "field1": "27.60",
        "field2": "76.40",
        "field3": "64.40",
        "field4": "124.50",
        "field5": "nan",
        "field6": "nan",
        "field7": "nan"
    },
```


Fig. 6. Postman Response of the ThingSpeak API retrieved from the AJAX Dashboard

IV. Solutions and Suggestions

Referring to the STRIDE methodology, Table II summarises the threat assessment for the three networks. Based on the analysis performed, this section proposes solutions and suggestions to security challenges for AirIoT and is extensible to other large-scale IoT networks. The section proposes an approach to end to end encryption, proof of concept deauthentication detector, protocol and dashboard security. All the suggestions have been implemented by the authorities concerned.

A. End-to-End Encryption

In the off-chance of a successful attack performed using the packet pipeline, a simple solution is proposed, which involves encryption and optional hashing. Sensed data are encrypted into a single string using an encryption algorithm and pushed to the server in our solution. The dashboard then pings the server and decrypts it at its backend, thus solving the challenge of information disclosure. Since only the two end parties have access to the sensed data, it is end-to-end encrypted. Additionally, hashing can be used to solve the problem against data integrity by appending the hash to the above string, recovering it at the dashboard's backend, and performing a hash check.

TABLE II
STRIDE ANALYSIS OF THE NETWORKS

Type of Network	N_1	N_2	N_3
Spoofing	MAC-spoofing performed to connect an illegitimate device to the network	Spoofing not practical	Spoofing not practical
Tampering	Data tampering to gain access to Write API keys; possibility of packet injection and replay attacks	Physical tampering to gain access to credentials	Physical tampering difficult due to placement on CCTV surveillance poles and use of eSIMs
Repudiation	No proof of origin of data for data authenticity	No proof of origin of data for data authenticity	No proof of origin of data for data authenticity
Information Disclosure	HTTP packets disclose full information about the traffic between the node and ThingSpeak, Write API key retrieved	TLS-encrypted packets after sniffing, no info about any http/https requests disclosed	Possible only after implementing MITM attack such as rogue base station
Denial of Service	Data availability affected through deauthentication	Data availability affected through deauthentication	Can be explored after implementing the rogue base station
Elevation of Privilege	Social engineering it from the deployment team	SSID-password written in the JioFi module	Can be explored after implementing the rogue base station

1) Solution to Repudiation: The issue of proof-of-origin can be solved using this method. Provisioning a unique identifier to each node, such as its channel ID, and incorporating it into the encryption rule can help determine breaches in the deployment. Any malicious attempt can easily be identified by verifying the validity of the channel ID of the received data at the dashboard's backend against a list of valid channel IDs. This method can be reinforced by maintaining a shared secret between the node and the dashboard about dynamically provisioning new unique IDs. Using date-time information can help protect against replay attacks.

2) Trade-Offs: The complexity of the encryption rule, however, directly affects the power consumption of the node. Encryption algorithms with higher time and space complexities result in extremely strong ciphers but are computationally heavier in terms of time and resources. This trade-off is also visible for hashing using state-of-the-art and custom hashing functions.

B. Deauthentication Detector

A proof-of-concept solution was proposed to detect the deauthentication attack and implement a workaround for the same. Although this isn't a system through which such an attack can completely be avoided, it ensures a method to improve the availability of data.

1) Hardware Setup: The solution consists of setting up a second ESP8266 in the deployment node to detect deauthentication packets, and send an alert to the deployment team about the details of the attack. Fig.7 explains the setup. A second ESP8266 is required here because the detector needs to remain in promiscuous mode at all times, staying disconnected from the Internet. The detector filters for packets based on the second half of their frame control field, namely 0xA0 for deauthentication and 0xC0 for disassociation frames.

2) Dynamic Whitelist of MAC Addresses: The vulnerability of MAC-spoofing on MAC-bound networks is exploited here. The MAC address of the sensor node is changed to a different MAC address with MAC-binding on the network. The idea is to set up a dynamic whitelist on the network controller's

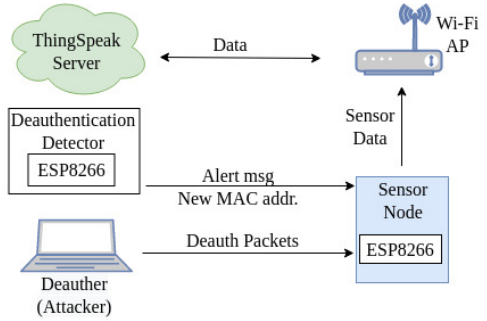

Fig. 7. Deauthentication Detector

end. The controller will update the new valid whitelisted MAC address for the main node to use in a deauthentication attack. In this scenario, since the network recognizes the new MAC address, the main node will be forced to change to this MAC address and then continue to push data to the server. The network controller dynamically changes this predetermined whitelist, and hence, the hacker won't be able to get a list of all the viable MAC addresses on the network.

3) Trade-Offs: The trade-off between response time and power can be balanced by minimizing the power consumption by letting the deauthentication detector run only when the sensor node has been disconnected from the Internet. This is to confirm whether the reason for the disconnection is due to deauthentication packets or not. The overall response time of the detector falls, and data is unavailable for a slightly longer period of time than in the former case.

C. Protocol Security

1) Security Against MITM Attacks: Packet replay attacks and packet injection can be averted if SSL/TLS is used instead, mainly through using HTTPS instead of HTTP. This is applicable to both N_1 and N_2. For N_3, changing the communication protocol from MQTT to secure-mqtt makes it difficult for an attacker to sniff and modify the packets, thus securing communication on a protocol level. Certain MQTT

brokers allow the use of TLS over TCP secure communication. The standardized name at IANA (Internet Assigned Numbers Authority) is "secure-mqtt" and port 8883 is standardized for a secured MQTT connection.

2) Trade-Offs With Moving to More Secure Protocols: There are multiple reasons for not using HTTPS and secure-mqtt protocols in IoT deployments. Setting up the appropriate certificates for such secure communication protocols is tedious and may not even ensure security [12]. HTTPS was designed for the public Internet and not to contain mechanisms, and hence becomes a scalability issue. The two protocols take a lot longer to communicate than their unsecured versions, and it gives rise to a response time versus security trade-off [13]. These two protocols also demand more resources and result in higher average power consumption, and hence a trade-off between security and power consumption arises. A workaround to avoid using HTTPS directly is to bring in the security SSL/TLS provides using the WebSocket protocol, as defined in RFC 6455 [14]. It brings in a more real-time approach to the problem without involving the substantial lag HTTPS provides.

D. Dashboard Security

When it comes to the dashboard, our analysis of the vulnerabilities in the AJAX version prompted the move to a more secure dashboard using a Django backend. To verify this, web-crawling was performed on the new landing site, and the admin page was revealed with a login form exclusively for admins. SQL Injection was performed on this form, and it is safe to say that Django's CSRF protection [15] facility blocked the attempt, and no information was leaked from the admin's database. The only way to break into this is to phish out details from the admin. Although quite difficult, it is not impossible to achieve that. Hence, one suggestion that was proposed was to make this page private in the directory. Another suggestion would be to make the page accessible only within a private network

V. Conclusion

In this paper, a security analysis of an IoT enabled pollution monitoring deployment, AirIoT has been performed. A protocol and network security analysis of the three types of communication networks used by the sensor nodes is presented. The entire analysis is modeled using the STRIDE framework. An original proof-of-concept solution is proposed, which implements a workaround to deauthentication attacks to improve the availability of data during such attacks. Additionally, the security of the dashboard is evaluated, and further measures are suggested. Future works include implementing the rogue base station attack to analyze the mobile network threat landscape. The security framework of OneM2M, a popular open-source implementation of the SmartM2M standard, will also be studied. The approach used to perform a threat assessment on the sensor network can be extended to other use cases that use Wi-Fi and mobile networks for communication based on the proposed threat and risk assessment methodologies.

VI. Acknowledgement

This research was supported partly by National Geospatial Programme (NGP), India, under grant no. 2073 (2020), PRIF Social Incubator Program (2019) and the Ministry of Electronics and Information Technology (MEITY) under grant no. 3070665 (2020), with no conflict of interests.

References

[1] "RFC 8576 - Internet of Things (IoT) Security: State of the Art and Challenges," [Online] Accessed: 11-Mar-2021, https://datatracker.ietf.org/doc/rfc8576/.

[2] "Baseline Security Recommendations for IoT," [Online] Accessed: 11-Mar-2021, https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot/.

[3] Luo et.al., "On the Security and Data Integrity of Low-Cost Sensor Networks for Air Quality Monitoring Sensors," *Sensors 2018*.

[4] Toma C et.al., "IoT Solution for Smart Cities' Pollution Monitoring and the Security Challenges," *Sensors (Basel). 2019*.

[5] *Dashboard-AirIoT IIITH*, [Online] Accessed: 15-Mar-2021, https://spcrc.iiit.ac.in/air/.

[6] C. R. Reddy et al, "Improving Spatio-Temporal Understanding of Particulate Matter using Low-Cost IoT Sensors," *IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications, London, UK, 2020*.

[7] Microsoft Security. 2007. STRIDE chart Microsoft Security, "IoT Solution for Smart Cities' Pollution Monitoring and the Security Challenges," [Online] Accessed: 14-Feb-2021, https://www.microsoft.com/security/blog/2007/09/11/stride-chart/.

[8] *Thingspeak*, [Online] Accessed: 08-Mar-2021, https://thingspeak.com/.

[9] *Matheus-Garbelini: esp32-esp8266-attacks, GitHub*, [Online] Accessed: 15-Mar-2021, https://github.com/Matheus-Garbelini/esp32_esp8266_attacks.

[10] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, and Jean-Pierre Seifert, "Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems," 01 2016.

[11] *Working of AJAX*, [Online] Accessed: 19-Mar-2021, https://www.javatpoint.com/how-ajax-works.

[12] *D. Albuschat, "Where is HTTPS for IoT? (Update)," DEV Community, 22-Oct-2018*, [Online] Accessed: 11-Mar-2021, https://dev.to/danielkun/where-is-https-for-iot-49ao.

[13] *The scalability problem with using HTTPS for M2M, Real Time Logic*, [Online] Accessed: 11-Mar-2021, https://realtimelogic.com/articles/The-scalability-problem-with-using-HTTPS-for-M2M.

[14] *The WebSocket Protocol, IETF Tools*, [Online] Accessed: 11-Mar-2021, https://tools.ietf.org/html/rfc6455.

[15] *Documentation, Django*, [Online] Accessed: 08-Mar-2021, https://docs.djangoproject.com/en/3.1/ref/csrf/.