Security for IoT Enabled Smart Cities

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering
by Research

by

Vigneswara Ihita Gangavarapu
2021701007
`ihita.g@research.iiit.ac.in`

International Institute of Information Technology
Hyderabad - 500 032, INDIA
June 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Security for IoT Enabled Smart Cities" by Vigneswara Ihita Gangavarapu, has been carried out under our supervision and is not submitted elsewhere for a degree.

Date	Advisor: Dr. Sachin Chaudhari

To

My Family and Friends

Acknowledgments

I want to thank my advisor, Dr. Sachin Chaudhari, for his constant guidance in my research and for providing me multiple opportunities to explore the depth of this field. His expertise and insightful feedback have been invaluable in shaping my work. I had the opportunity to work on challenging research problems and much to learn from my co-authors: Sachin Gaur, Dr. Thierry Monteil, Sudhansh Yelishetty, Viswanadh Kandala, Vybhav Acharya, and Likhith Kanigolla. I thank them for their contributions to the research. My gratitude also goes to Mrs. Anuradha Vattem and the Smart Cities Living Lab of IIIT-H for enabling my work. I am grateful to I-Hub Data for their fellowship that supported my research. The on-ground experiments conducted as part of the IIIT-H smart city security research were performed with the support from the IT team of IIIT-H. I thank the IT team, especially Ms. K.Sriranjani (Head, IT Services) and Mr. Madhu Babu Dontineni (System Administrator) for trusting the testing approach, granting the necessary permissions and providing a sandbox for the experiments whenever needed.

I appreciate my lab mates Ayush, Ruchi, Ayu, Spandhana, David, Nikhil, Sasanka, Viswanadh, and others for keeping the energy high at SPCRC. During my master's at IIIT-H, I came across incredible people who have been a strong support system. I couldn't have done without Sudhansh, Prajneya, Pranav, Eshika, Pratik, and the "Ashwathama band" members. Thank you for the amazing memories. The covid-19 period was full of ups and downs. I value the help and company of Harshit, Shanthika, Kinal, Anushka, Arpita, Shantanu, and Mounika for making the period go by happily and seamlessly. I met many people through my participation in music events, fests, and sports on campus. I acknowledge their presence in making my stay here so wonderful.

I am eternally grateful to my parents for their unwavering support, encouragement, and love throughout my academic journey. Their belief in me has been my driving force and motivation.

Abstract

Smart cities leverage IoT to improve citizens' quality of life by providing better infrastructure, enhanced transportation systems, and efficient public services. With IoT-enabled smart city applications receiving traction, mechanisms must be implemented to cater to the diverse requirements of the use cases. The increasing number of connected devices increases the risk of cyber attacks and breaches. Further, smart cities rely heavily on integrating critical infrastructure such as power grids and water treatment plants. Securing these systems is crucial to protect citizens and institutions from potential harm caused by attacks. However, securing the IoT ecosystem is a challenging task. There are constraints on processing, memory, and energy consumption in addition to interoperability challenges, cost, and complexity. Designing and implementing the appropriate security controls for the devices and services requires on-ground testing and analysis.

This thesis explores and analyzes the security of IoT-enabled smart cities. The work consists of two main contributions. First, IoT security requirements, potential threats, and vulnerabilities to the various layers of IoT are analyzed. Vulnerability assessment and modeling of threats are performed on the air quality monitoring vertical of the smart city deployment of IIIT-H to gain visibility into the baseline security requirements. The recommendations and state-of-the-art solutions are extensible to applications that require Wi-Fi and mobile network-based security for large-scale IoT deployments. Second, the entire smart city deployment was examined, covering applications such as air quality monitoring, water quantity management, weather monitoring, and energy monitoring. With each application operating using different hardware components, communication technologies, and software dependencies, requirements such as interoperability, scalability, resiliency, and security are essential. This study focused on the provisions of the oneM2M standard in catering to these requirements and its role in smart cities. Experiments were conducted on the OM2M platform, an implementation of oneM2M, used in the smart city deployment of IIIT-H. The recommendations are a foundation for the security of oneM2M-based smart city applications.

Contents

List of Figures

List of Tables

List of Abbreviations

ACOR	Access Control Originator
ACP	Access Control Policies
AE	Application Entity
API	Application Program Interface
CIA	Confidentiality-Integrity-Availability
CRUD	Create-Read-Update-Delete
CSE	Common Service Entity
CSRF	Cross-Site Request Forgery
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial-of-Service
DNS	Domain Name System
ENISA	European Union Agency for Cybersecurity
Es-Prim	End-to-end security of primitives
ETSI	European Telecommunications Standards Institute
GBA	Generic Bootstrapping Architecture
GSMA	Groupe Speciale Mobile Association
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IIIT-H	International Institute of Information Technology, Hyderabad
IoT	Internet of Things
IPE	Interworking Proxy Entities
ITU	International Telecommunications Union
IUDX	Indian Urban Data Exchange
LoraWAN	Low-Power Wide Area Network
M2M	Machine-to-Machine
MITM	Man-in-the-Middle
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NSE	Network Service Entities
OM2M	Open Machine-to-Machine

RBAC	Role Based Access Control
REST	Representational State Transfer
SAEF	Security Association Establishment Framework
SCL	Service Capability Layers
SQL	Structured Query Language
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS/SSL	Transport Layer Security/Secure Sockets Layer
URN	Uniform Resource Name
WiFi	Wireless Fidelity

Chapter 1

Introduction

1.1 Motivation

Countries are working towards making smart cities a reality. A smart city uses cutting-edge technology to improve sustainability, optimize urban services, and increase the quality of life for its residents. To gather and process data, automate processes, and offer insights that aid city administrators in making better decisions, it integrates a variety of technical systems, including sensors, communication networks, and data analytics. A smart city aims to employ technology to boost public safety, infrastructure, and economic development while making cities more effective, livable, and sustainable. Even in India, 100 cities have been identified by the Smart Cities Mission (SCM) with projects to improve core infrastructure and services, making cities more livable, economically diverse, and environmentally sustainable. The smart city application pipeline is complex, with hardware platforms, communication technologies, and interfaces. Several components from different vendors make interoperability, scalability, security, and resiliency essential requirements for smart city applications.

With the various advantages that IoT provides, we see a surge in the adoption of IoT-enabled applications. Books [1, 2, 3, 4] are good reads on IoT that discuss the technology, its architecture, and its potential to solve real issues. The use of IoT in the supply chain industry reduces costs by automating processes, optimizing resource utilization, and improves supply chain management. It provides real-time data for businesses, enabling them to deliver personalized customer services. IoT plays a major role in effective smart city governance. For smart city applications such as waste management, traffic management, air pollution monitoring, and energy management, IoT devices help monitor and regulate various parameters. An example is the use of IoT for smart traffic management systems, which can reduce congestion and improve safety on the roads. Similarly, smart lighting systems can save energy and reduce light pollution. The sensors and cameras can monitor public spaces and detect unusual activity or potential safety hazards.

Incorporating security to protect confidentiality, integrity, and availability in smart cities is critical and challenging. IoT devices in smart city applications can be inexpensive sensors with memory, power, and computational limitations. Security typically suffers due to trade-offs between cost, deployment situation, computing power, and IoT device security. With manufacturers and consumers prioritizing functional and operations requirements over security, a new dimension of challenges arises. These devices are exposed to threats, such as physical tampering leading to side-channel attacks and man-in-the-middle attacks. Vulnerabilities in the communication network can be exploited, leading to a breach of privacy. Weak application programming interfaces (APIs) can affect the integrity and authenticity of users and data. All these threats make incorporating security even more critical, especially when dealing with smart cities. To implement the right security controls, on-ground analysis and verification on real large-scale smart city deployments are required. The overall work presented in this thesis explores specific security vulnerabilities and threats to smart city applications such as air and water quality monitoring systems. The security analysis includes analyzing the physical, network, and application layer security, including authentication mechanisms for nodes, data confidentiality, and the use of secure APIs.

With the fragmentation of standards being one of the challenges with IoT security, oneM2M [5] proposes a common middleware technology in a horizontal layer covering the smart city requirements. This makes the standard a de facto for a smart city as it reduces fragmentation, facilitates large amounts of data sharing, increases the reusability of underlying technologies, and optimizes costs. Various implementations of the standard are being developed and implemented globally. Recently, India has adopted the oneM2M standard as the national standard. The oneM2M standard includes many security measures. Access control policies, dynamic authorization, application/device impersonation prevention, and privacy protection are only a few security requirements and practices. This thesis presents a detailed analysis of the security specifications of the oneM2M standard and its implementation, OM2M [6], used by the smart city deployments of the International Institute of Information Technology (IIIT-H), Hyderabad [7]. This includes modeling the threats using security frameworks and incorporating appropriate security features for our use cases.

There is an urgent need for security analysis and verification on IoT/M2M-enabled smart city pilots and testbeds to develop proof-of-concept solutions. This has motivated my research of carrying out the security analysis on the actual IoT-enabled smart city deployment and providing baseline security requirements for future developments of smart cities.

1.2 Summary of contributions

The main contributions of this thesis are split into two chapters:

- **Chapter 4**
  - The sensor network for air quality monitoring, AirIoT [8], is examined.
  - The vulnerability assessment and threat modeling of the three IoT communication networks, implemented in AirIoT, is performed using the STRIDE threat modeling framework [9]. Packet pipeline, deauthentication, and stealcon attacks are some attacks performed as part of the vulnerability assessment. [10]

  - An analysis of the different communication protocols, such as MQTT, and HTTP used in the deployment is conducted. Threat assessment for the remotely accessible dashboard is also performed.
  - Solutions are presented to the vulnerabilities exposed in AirIoT, which can be extensible to general large-scale sensor networks. Solutions include a proof-of-concept deauthentication detector, a mechanism for end-to-end security of communication, mitigation against man-in-the-middle (MITM), and recommendations on dashboard security.

- **Chapter 5**
  - This chapter presents a security analysis of oneM2M-based implementation called OM2M deployed as part of the smart city network of IIIT-H.
  - Security provisions of the oneM2M standard are studied. Potential threats to oneM2M implementations with existing standard provisions are mapped to the STRIDE framework.
  - Security analysis of OM2M-based smart city deployment is conducted via various attacks, including eavesdropping, brute force, and packet replay attack.
  - Recommendations are proposed based oneM2M provisions and the observations from the assessments. These recommendations are also mapped to each of the STRIDE elements. [11]

1.3 Organization of thesis

The remainder of this thesis is organized as follows:

- *Chapter 2* presents an overview of IoT, and its fundamental architecture, with applications in various sectors, followed by the limitations.
- *Chapter 3* covers the key challenges when creating a secure IoT ecosystem, common threats, global efforts on IoT security requirements, and security for IoT-enabled smart cities.
- *Chapter 4* provides threat modeling and risk assessment of a large-scale IoT-enabled pollution monitoring sensor network, AirIoT.
- *Chapter 5* deals with the potential threats to oneM2M standard-based implementations and presents recommendations based on actual on-ground analysis conducted on OM2M-based smart city deployment.
- *Chapter 6* serves as the conclusion of the thesis.

Chapter 2

An overview on IoT

This chapter first presents some global definitions of IoT. This is followed by its architecture, numerous applications, and challenges. The chapter covers the topics only briefly. For a more in-depth understanding of IoT, you may refer to these: [1, 2, 12, 3, 4]

2.1 Definitions

Broadly, IoT is a network of physical devices that sense the environment, talk to each other, and interact with humans by bringing together technologies such as sensors, actuators, high-performance computing, wireless connectivity, and cloud storage/internet. Over the years, multiple definitions of IoT have evolved. As per the United Nations International Telecommunication Union [13], IoT is defined as *a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.* The Internet Engineering Task Force (IETF) [14] defines it as the *network of physical objects or "things" embedded with electronics, software, sensors, actuators, and connectivity to enable objects to exchange data with the manufacturer, operator, and/or other connected devices.* In [15], Gartner Research defines it as the *network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment.*

2.2 Architecture

The architecture of IoT depends upon its usability, functionality, and implementation in different domains. The architecture typically includes several layers, each serving a different purpose in the overall system. An overview [16, 17] of the standard IoT architecture is provided below and as shown in Fig. 2.1:

- **Devices/Things:** Sensors, actuators, and other smart objects connected to the internet make up the initial layer of IoT architecture. These devices typically have embedded microprocessors, sensors,

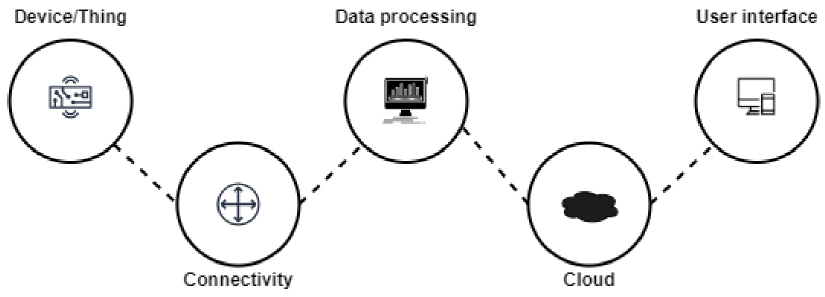


Figure 2.1: Layers in IoT architecture

and communication modules that enable them to collect and transmit data. They continuously collect environmental information and transmit it to the following layer. Some examples of common sensors are temperature sensors and thermostats, pressure sensors, humidity and moisture sensors, light intensity detectors, proximity detection, and RFID tags.

- **Connectivity:** The second layer of the IoT architecture contains the networks and protocols that enable devices to connect. Depending on the use case and requirements, these may include wired or wireless networks, such as Wi-Fi, Bluetooth, Zigbee, cellular, or satellite communication. IoT gateways also aid with connectivity. Gateways are employed to translate various network protocols to ensure that the linked devices and sensors can communicate with one another. They control the two-way data flow among various networks and protocols. It also serves as a middle layer between the cloud and the devices to guard against malicious intrusion and unauthorized access.

- **Data processing:** The edge devices or gateways that gather and preprocess data from the sensors and devices before delivering it to the cloud or a central server are included in the third layer of IoT architecture. This layer may also include edge computing or fog computing nodes that perform real-time analytics, ML, or AI algorithms on the data. Processing helps businesses capture and apply insights to their upcoming business possibilities.

- **Cloud:** The cloud-based platform or server makes up the fourth layer of the architecture. IoT generates enormous amounts of data from users, applications, and devices that must be effectively handled. IoT cloud provides capabilities to quickly gather, manage, process, and store massive amounts of data. It receives and stores data from edge devices and offers a range of services, including data management, analytics, visualization, security, and integration with other programs or systems. A cloud-based system involving billions of devices, sensors, gateways, protocols, and data storage is used to give predictive analytics. Businesses utilize analytics to improve their products and services and precisely build new business models.

- **User interface:** The fifth and final layer of IoT architecture includes the end-user applications or interfaces that allow users to interact with the IoT system, visualize the data, and control the devices. These can be web or mobile apps, dashboards, or APIs that provide access to the IoT platform and the data. User interface design is important because it frequently influences the user's decision to select a certain product or appliance.

Ultimately, the IoT architecture is built to provide for the secure, scalable, and reliable operation of the system while enabling the seamless integration and communication of various devices and services across various layers.

2.3 Applications:

IoT has revolutionized how we live our lives, and there are multiple areas where it has had a significant impact. The applications of IoT are vast and growing rapidly and here are some examples of how IoT is being used today: [18, 19]

- **Smart cities:** IoT-based smart city applications offer a wide range of benefits, including improved energy efficiency, optimized city planning, enhanced public safety, and more. Smart waste management detects the level of waste in bins and containers. The collection routes and schedules are then optimized using this data, which lowers costs and boosts productivity. Smart lighting systems are being used in many cities around the world. Sensors are used by these systems to detect the presence of people and vehicles and to change the lighting accordingly. By moderating the use, it can improve vision in dimly lit places, enhance public safety and reduce energy consumption. Other applications of smart cities include smart traffic management to improve the flow of traffic, smart parking to detect the availability of parking spaces and help reduce traffic congestion and improve the overall parking experience and smart public safety to monitor public spaces and detect potential security threats.
- **Smart home**: IoT-enabled devices and systems allow homeowners to monitor and control various aspects of their homes from anywhere at any time. Some of the IoT technologies that are commonly used in smart homes include smart thermostat devices allowing users to control the temperature of their homes remotely, smart lighting using a smartphone app or voice commands, and smart security systems containing a combination of cameras, sensors, and alarms to monitor the home and alert the homeowner of any unusual activity. Other applications include smart locks and smart entertainment systems.
- **Industrial IoT (IIoT)**: IoT sensors can be used to track and enhance industrial processes, such as those in factories or power grids, in order to increase productivity and decrease downtime. It uses interconnected sensors, instruments, and other devices in the industrial sector to collect and exchange data. The applications of IIoT are diverse and rapidly expanding. For predictive maintenance, IoT continuously monitors industrial machinery and equipment. Smart devices help with quality control and supply chain management that tracks inventory levels, monitors shipping

conditions, and optimizes supply chain logistics, reducing costs and increasing efficiency [20, 21]. They also help with asset tracking, energy, and environment management. Overall, IIoT is helping reduce downtime, improve reliability, reduce costs, and increase efficiency.

- **Health care:** Smart devices are changing how we receive healthcare, empowering not just regular individuals to monitor and track their health in daily life but also centers that serve physicians and patients with cutting-edge insights and analytics. IoT deployment in healthcare has risen recently due to the COVID-19 outbreak. Remotely monitoring patients is a key application in healthcare [22]. Here, IoT-enabled devices collect vital signs such as blood pressure, heart rate, glucose levels, and oxygen saturation, enabling healthcare professionals to monitor patients' health in real-time remotely. IoT-enabled wearables aid with remote monitoring. Wearable devices such as smartwatches, fitness trackers, and smart clothing can help individuals track their physical activity, sleep patterns, and overall health status. These devices also help with medication management where they remind patients to take their medication at the appropriate time and can alert healthcare professionals. Other critical applications are in telemedicine where IoT devices connect patients with healthcare professionals via video conferencing and smart hospital systems that help hospitals optimize their operations by monitoring the use of medical equipment, tracking inventory, and ensuring that rooms are cleaned and sanitized appropriately.

- **Smart agriculture:** IoT in agriculture helps improve efficiency, reduce waste, and increase yields. Smart irrigation systems are where IoT sensors measure soil moisture, temperature, and humidity to determine the optimal water needed for crops. This data can be used to control irrigation systems, reduce water waste and improve crop yields. Livestock monitoring to monitor the health and well-being of livestock help farmers to identify and treat health issues early, reducing the risk of disease outbreaks and increasing productivity. Soil-related information through precision agriculture creates detailed field maps, guiding precision farming techniques such as variable rate fertilization and targeted pest control. Crop monitoring is another IoT application to monitor the growth of crops in real time, collecting data on factors such as temperature, humidity, and light levels. In [23], the authors present a study on the status of rural connectivity, including potential use cases. They focus on the state-of-the-art initiatives, challenges, and technologies to improve digital connectivity in the Indian scenario. Robust connectivity will help leverage IoT for agricultural applications in rural sections.

- **Smart grids:** The grid refers to the electric grid, a network of transmission lines, substations, and transformers that deliver electricity from the power plant to your home or business. [24] Utility businesses may more efficiently monitor and control energy usage with the help of IoT-enabled smart meters that can provide real-time data on energy consumption. They can also help consumers track their energy usage and make informed decisions about their consumption. The efficient storage of energy via IoT and automation of its distribution help to balance the supply and demand of electricity on the grid and lessen the need for fossil fuel-based power plants. In addition, this enables utilities to quickly identify and respond to outages, reduce strain on the grid, and prevent blackouts, improving the reliability and efficiency of the grid. In smart grids, IoT can be used to monitor the health of equipment and predict when maintenance will be required.

2.4 Challenges in IoT

IoT devices are typically referred to as resource-constrained devices since they have fewer computation, storage, energy, and networking resources than regular computer devices. To fully fulfill the future potential of IoT, numerous more limitations in addition to resource-based ones must be addressed. Some key limitations that need addressing are as below, [25, 26]

- **Low-cost requirements:** Conventional sensing technologies have high acquisition costs and installation complexity, frequently requiring substantial resources, infrastructure, and knowledge. However, with increased demand for smart devices, the cost of IoT has been reduced. IoT comprises low-cost sensors. The cost constraints can lead to limiting other features, such as security, to ensure a low cost.

- **Deployment and scalability:** The deployment phase is the culmination of the preparation for security and data privacy, testing and enabling interoperability, establishing backups, customer support, and technical support for networks, gateways, web interfaces, apps, and cloud platforms. It is crucial to acquire assurances that the system can be supported and serviced during the crucial periods of initial implementation, deployment, and subsequent maintenance for all infrastructure. There may be several problems, including getting the necessary permissions from the relevant authorities and choosing the right deployment area. The scalability of IoT systems can also be a challenge, mainly when dealing with large numbers of devices. As more devices connect, managing them and guaranteeing seamless interoperability becomes difficult.

- **Connectivity:** It is essential to offer dependable and seamless connectivity to the numerous devices connected to the internet. Connectivity is one of the fundamental components of IoT, as data transportation depends upon reliable connectivity. Connectivity is made possible via communication protocols like 802.15.4, Bluetooth, ZigBee, etc.

- **Power management:** Several IoT devices are made to be low-power and operate for extended periods of time without needing maintenance or battery replacement. The functionality and performance of IoT devices as well as the IoT system as a whole may be significantly impacted by these power limitations. A key problem in extending a sensor node's lifetime is when the node is difficult to access or requires a lot of maintenance.

- **Privacy and security:** The security of data is a major problem as more devices are deployed in hostile environments in an unregulated and remote manner [10]. Resource constraints make it difficult to carry out complex cryptographic procedures for secure communication, over-the-air upgrades, and data protection. The use of hard-coded weak passwords, a lack of reliable patches, a lack of secure interfaces, fragmented security standards, and skill gaps all make it more difficult to ensure the security of IoT devices.

- **Interoperability:** The capacity of two or more devices, systems, platforms, or networks to cooperate is known as interoperability. IoT devices may employ many communication protocols and use diverse hardware components, making integrating them into a single system difficult. Additionally, combining and analyzing data from many sources might be challenging since the data supplied by IoT devices may be in different formats.

Chapter 3

Security for IoT: A literature survey

Security is a critical component to consider when dealing with IoT. IoT applications are intertwined with our lives, and security breaches will have a detrimental impact on all. Smart devices range from home appliances and wearable devices to industrial systems and medical devices. The data they collect includes personal, location, and financial data. Malicious actors can use these devices to launch large-scale attacks, steal personal data, or compromise critical infrastructure.

With identifying the best practices for the IoT application, one usually first understands the threats, and threat actors, then the security requirements of the IoT use case, and lastly, assesses vulnerability to gain insights on the appropriate mitigation mechanism. The sections in this chapter focus on threats to IoT systems and modeling methods, the global contributions made around the security requirements, and vulnerability assessment using penetration testing followed by IoT security challenges. The last section is more specific to the smart city use case and provides an overview of smart city security.

3.1 Threats and modeling

The threats are dynamic and ever-evolving. The survey [27] provides a comprehensive list of threats, including attacks on physical aspects, encryption, network, and application. Threats to the node and the device's sensors include node tampering, outage attack, and sleep deprivation attack. Cryptanalysis and side-channel attacks are a threat to encryption. The networks are vulnerable to a MITM attack, denial of service (DoS), Sybil, and replay attacks. Malicious scripts, botnets, and phishing attacks affect the application layer. Traditional cryptographic solutions, physical unclonable functions (PUF) and blockchain-based solutions are also presented. In [28] the authors identify major threats in the hardware, communication network, and smart application environment. Modern IoT security threats and vulnerabilities have been investigated for applications such as the smart environment, intelligent transportation, smart grid, and healthcare system. Based on these, the most commonly used security techniques with use cases are described. The authors in [29] categorize threats into three sections: hardware, software, and data in transit. The paper focuses on the growing vulnerabilities and threats due to integrating IoT and emerging technologies such as blockchain, machine learning, and fog computing. RFC 8576 [30] 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 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). A recent report by ENISA [31] identified ransomware, malware, social engineering attacks, threats against data, and threats against availability as some of the prime threats. Securing IoT devices requires a multi-layered approach that includes both hardware and software solutions to address the various challenges and threats. IoT devices can benefit from hardware-based security features like Trusted Platform Modules (TPMs) and secure enclaves, as well as software-based security measures like secure boot and secure firmware updates that can help shield against malware and other threats.

Designing and implementing mitigation strategy first requires understanding the threat landscape. Threat modeling identifies potential security threats and vulnerabilities in a system or application. Several frameworks can be used for threat modeling, some of which are [32]:

- **STRIDE:** It is a framework that assists in recognizing and classifying potential risks based on six different types of attacks: spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege.

- **PASTA:** It stands for Process for Attack Simulation and Threat Analysis (PASTA). The first six steps in the framework consist of objectives, system description, threat analysis, vulnerability analysis, attack modeling, and risk analysis.

- **VAST:** A lightweight and agile threat modeling paradigm called Visual, Agile, and Simple Threat Modeling (VAST) focuses on a simple and clear display of threats. There are three phases to it: identify, map, and assess.

- **OCTAVE:** Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) is used to identify potential threats to an organization's assets. It combines organizational and technological viewpoints to produce a thorough threat model that can be used to guide risk management choices.

3.2 Security requirements

It is important to emphasize the necessity for appropriate security controls to mitigate threats. However, security requirements for IoT are not straightforward. The IoT ecosystem is complex, with a variety of components. The security requirements depend on the assets' criticality, known vulnerabilities, potential threats, identification of threat boundaries, and the IoT application itself. In addition, each layer of the IoT architecture will have a unique set of security requirements.

Table 3.1: Broad IoT security requirements

Requirements	**Description**
Confidentiality	Ensures that information is accessible by legitimate and authorized entities only
Integrity	Ensures reliability and accuracy of information
Availability	Ensures the availability of services and information and that it is not denied to authorized entities
Non-repudiation	Ensures proof of origin of information and entities' non-deniability in a transaction.
Authenticity	Ensures system, resource, or communication is genuine, has not been tampered with by an unauthorized party and can be verified.
Trustworthiness	Ensures the quality and behaviour of a system, product, or process meets the intended security objectives by being reliable, secure, and resilient.
Privacy	Ensures protection of personal information and data collected by devices from unauthorized access, use, and disclosure.
Accountability	Ensures tractability and assigning responsibility for actions or events that impact the security and privacy of data and services

Authors in [33] offer recommendations for improving security using Blockchain, fog computing, and ML-based approaches. Numerous research [28, 27, 29] cover the vulnerabilities and potential threats to IoT. The works categorize the threats based on the architecture and components of IoT. Global standardization efforts are evolving to ensure the security of the IoT ecosystem. In [26], ENISA identifies and analyzes existing IoT security practices, guidelines, relevant industry standards, and research initiatives. Key asset groups and the criticality of these assets are identified in the asset taxonomy. Additionally, threats and the effect of the threats on these assets are mentioned in the threat taxonomy. Finally, security recommendations are proposed based on the security measures developed and the challenges identified. ISO/IEC 27001 [34] outlines the requirements for information security management systems. The NIST Cybersecurity Framework [35] is for improving critical infrastructure cybersecurity, which includes guidelines for securing IoT devices. ETSI [36] focuses on the minimum requirements for IoT device security. It covers a range of areas, such as passwords, firmware updates, and data encryption. Similarly, the GSMA IoT Security Guidelines [37] provides IoT security guidelines for services, endpoint ecosystems, and network operators. Report on recommendations for IoT / M2M security from TEC [38] focuses on the potential vulnerabilities, IoT threat landscape, and security requirements for IoT and M2M devices. ETSI's 'Consumer IoT security' [36] standard presents baseline requirements for consumer IoT devices that include the security of personal data, secure software updates, and secure communication. Fundamentally, requirements of confidentiality, integrity, and availability (also known as the CIA triad) [39] must be satisfied to ensure the security of information and services. Table 3.1 describes the CIA triad and other general requirements [27] for IoT security.

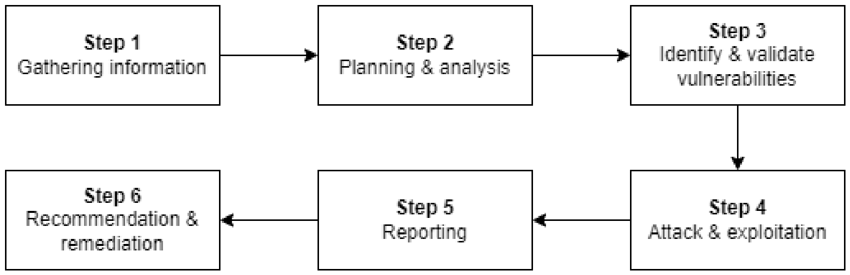


Figure 3.1: Steps for penetration testing

3.3 Vulnerability assessment

Identifying and evaluating potential vulnerabilities in IoT systems, networks, and devices constitute vulnerability assessment. This is specific to the threat actors, use case, and assets involved. Penetration testing, commonly called ethical hacking, simulates an assault on an IoT system to find weaknesses that malicious actors could use to their advantage. Penetration testing can be done manually or automatically, frequently combining the two. Figure 3.1 depicts the steps involved in penetration testing to identify and report vulnerabilities.

1. **Gathering information:** This involves gathering information about the target systems such as user accounts, system configurations, hardware components, and network topology.

2. **Planning and analysis:** This involves defining the scope of the test, and identifying the target IoT system, the assets, and the threats.

3. **Identify and validate vulnerabilities:** Manual methods or automated tools can scan the target systems to identify open ports, services running, and potential vulnerabilities.

4. **Attack and exploitation:** Attempts are made to exploit the identified vulnerabilities to gain access to the target system. This includes escalating privileges, maintaining access, and gathering sensitive information.

5. **Reporting and recommendations:** Documenting the penetration test findings and providing recommendations for mitigating the identified vulnerabilities.

6. **Recommendation and remediation:** Fixing the identified vulnerabilities and retesting the system to ensure that the vulnerabilities have been effectively mitigated.

3.4 IoT security challenges

Security requirements for IoT systems need to be a focus area starting from the manufacturing of IoT devices till their disposal, covering an entire lifecycle of IoT. However, incorporating security becomes a challenge due to IoT's various limitations. Some common yet key reasons why securing IoT is challenging are as follows, [26]

- **Heterogeneity and complexity of ecosystem:** Security is made more difficult by the diverse nature of the physical equipment, communication technologies, and applications, as well as the requirements for interoperability. Further. IoT security integration is becoming challenging due to the potential for conflicting demands and opinions from all stakeholders involved.
- **Low cost:** With the extensive penetration of IoT and its widespread deployment, manufacturers and consumers prioritize functional and operational requirements over cost. It frequently happens that the cheap cost affects the addition of important features such as security.
- **IoT resource limitations:** Because most IoT devices have inadequate processing, memory, and power capabilities, it isn't easy to properly apply even baseline security controls.
- **Vast attack surface:** Attack surfaces in IoT systems and applications are where threats and vulnerabilities may exist. The overall threat landscape of IoT is vast, comprising devices, communication channels, the cloud, software, and applications. The data (such as credentials, sensed data, and sensitive data) also fall under the threat landscape.
- **Fragmentation of standards and regulations:** Security concerns are exacerbated due to fragmented standards, slow adoption of regulations, and the introduction of newer technologies.
- **Lack of clear liabilities:** The absence of a clear assignment of liability could cause confusion and disputes, especially in light of the extensive and intricate supply chain involved in IoT. Furthermore, there is still no solution to the problem of managing security when several parties share a single component. Another important problem is enforcing culpability.
- **Lack of expertise:** Although initiatives are working towards building the necessary cybersecurity skill sets in organizations and individuals, there is still a long way to go.

3.5 Security for IoT-enabled smart cities

Smart cities employ cutting-edge technology and data analysis to raise their citizens' living standards, promote sustainability, and increase the effectiveness of city services. Technologies such as the IoT, sensors, big data analytics, and artificial intelligence (AI) collect and analyze data in real time, providing insights that aid in effective governance. Singapore is frequently mentioned as one of the top smart cities in the world [40]. The city-state has implemented a comprehensive smart city program that includes initiatives in areas such as transportation, energy, and public services. For example, Singapore's intelligent transport system uses real-time data to manage traffic flow, reduce congestion, and improve public transportation. Barcelona is another leading smart city known for its innovative use of technology to improve urban life. The city has implemented a range of smart city initiatives, including a sensor network that monitors air quality and noise levels, smart parking systems, and a bike-sharing program. Similarly, Amsterdam, Dubai, and Helsinki have also focused on meaningfully utilizing technology to make their cities smarter. In India, smart cities are part of the Smart Cities Mission [41] launched by the Indian government in 2015 to transform 100 cities across the country into smart cities. The mission focuses on improving citizens' quality of life, promoting sustainable and inclusive growth, and using technology and innovation to improve urban services and infrastructure. Work is underway to implement smart city projects in each of these cities.

IIIT-H [42] has also collaborated with the government of Telangana to promote smart city initiatives in Hyderabad. The initiatives include air and water quality management systems, smart campus facilities, and installing occupancy sensors to control lighting and air conditioning for energy management.

3.5.1 Securing smart cities

Authors in [43, 44] provide a comprehensive overview of the dimensions of smart cities and the associated risks. The study explores smart cities' technology, governance, social inclusion, environmental sustainability dimensions, and potential risks surrounding cybersecurity. In [45, 46, 47], a survey on the applications and security threats to smart cities is covered. Various deliberations are taking place globally, and study groups [48, 49] of ITU focus on smart cities and security, respectively. Present below are some of the popular cyber attacks experienced by IoT-enabled smart cities [45, 50, 47, 46]

- **Physical attacks:** Physical attacks on smart cities can include vandalism, theft, or destruction of physical infrastructure, such as sensors, cameras, or control systems. Physical attacks can disrupt the functioning of the smart city and cause significant damage.

- **Man-in-the-middle attacks:** MITM attacks occur when an attacker intercepts the communication between two systems and manipulates the data. This type of attack can be used to steal sensitive data or take control of critical infrastructure.

- **Denial-of-service attacks:** DoS attacks occur when an attacker floods a system with traffic, rendering it unable to function properly. This type of attack can be used to disrupt critical services such as emergency response or transportation systems.

- **Insider attacks:** Insider attacks occur when an authorized person with access to the smart city's systems misuses their access to cause harm to the system. This attack can be difficult to detect and can cause significant damage.

- **Social engineering attacks:** Social engineering attacks are a type of cyberattack that uses manipulation techniques to trick people into divulging sensitive information or performing actions that could harm the smart city's infrastructure.

- **Advanced persistent threats:** Advanced persistent threats are a type of cyberattack that involves a prolonged and targeted effort by a group of attackers to gain access to the smart city's systems. These attacks can be difficult to detect and cause significant damage if successful.

There are certain must-haves for the security of IoT-enabled smart cities. It is essential to secure communications to prevent unauthorized access and ensure the integrity and confidentiality of data. Next, access control mechanisms must be implemented and achieved through strong authentication mechanisms like two-factor authentication and biometric authentication. Since IoT devices collect vast amounts of data from different sources, data collected should be salted/hashed before storing. It is crucial to continuously monitor IoT devices and sensors to detect and respond to real-time security incidents. This includes intrusion detection, threat intelligence, and behavioral analytics. Globally, the privacy of users and their data is gaining much-needed traction. The appropriate privacy rules and regulations must be complied with when collecting and processing personal data. Smart cities should also have plans to recover from security incidents quickly, address security vulnerabilities and ensure the continuity of critical services.

Chapter 4

Security analysis of large scale IoT network for pollution monitoring in urban India

In this chapter, 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. STRIDE methodology is used to analyze the various threat vectors in the deployment. An approach for end-to-end encryption, protocol, and dashboard security and a proof of concept deauthentication detector are presented as solutions.

4.1 Introduction

With the rapid adoption of IoT-enabled smart city technologies, security becomes critical 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 CIA triad of network and information.

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, 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 work. There have been few works in the literature on IoT security for air pollution networks [51, 52]. In [51], 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 MITM attacks performed by adversaries to sniff/modify data, thus compromising data privacy and confidentiality. In [52] the authors discuss data security and consistency of data sent in pollution monitoring deployment for

Table 4.1: 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

smart cities. The criticality of secure communication protocols is mentioned, with a focus on the use of secure MQTT.

This work presents network and protocol-level vulnerability assessment for an existing large-scale, low-cost pollution monitoring sensor network, AirIoT [8], deployed in urban India. AirIoT is an extended work of [53]. 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 key contributions present in this chapter are as follows:

- The vulnerability assessment of the three different IoT communication networks, implemented in AirIoT, was performed using STRIDE [9]. STRIDE methodology is used to identify potential security threats to a system as described in Table 4.1.
- 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.

4.2 Deployment scenario

In reaching the milestone of 50 node deployments, there are currently 30 nodes deployed, as shown in fig.4.1. The nodes are placed in and around an Indian educational institute, IIIT-H. The initial deployment of 30 nodes is being developed further to accommodate new nodes and better network technologies, giving rise to a more robust system. Each sensor node consists of ESP8266 micro-controller (NodeMCU), SDS011 sensor to monitor particulate matters (PM2.5 and PM10) and sensors to monitor

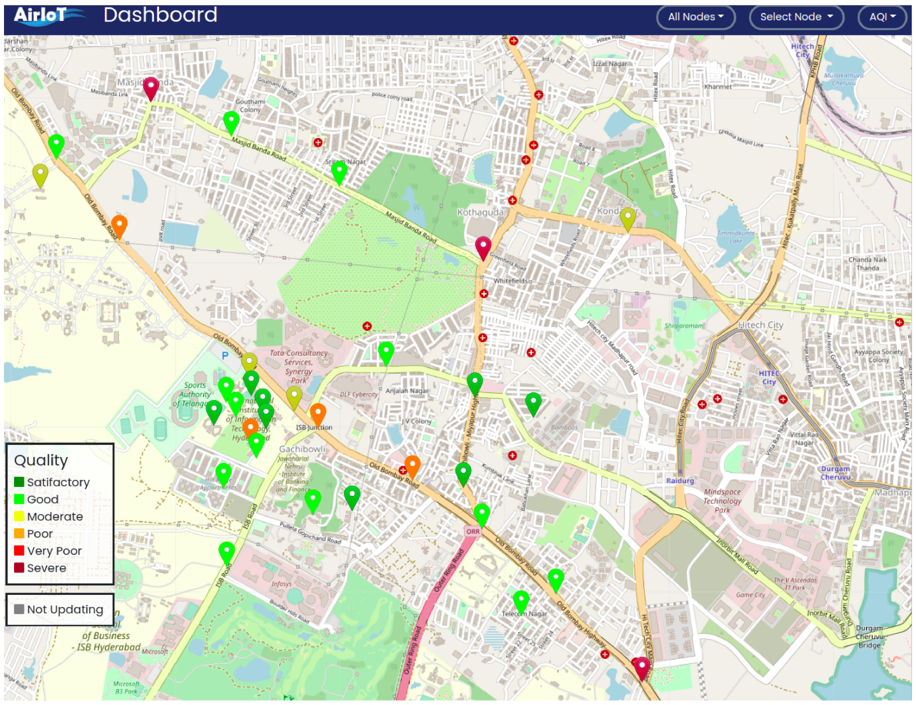


Figure 4.1: Deployment of sensor nodes

temperature and humidity. The sensor nodes send sensed data to the ThingSpeak server, an open-source IoT platform, [54] 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
- **Network N_3:** Nodes connected via embedded SIM

Fig.4.2 shows the three types of communication networks sending data to ThingSpeak server. 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 three 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.

4.3 Vulnerability assessment

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

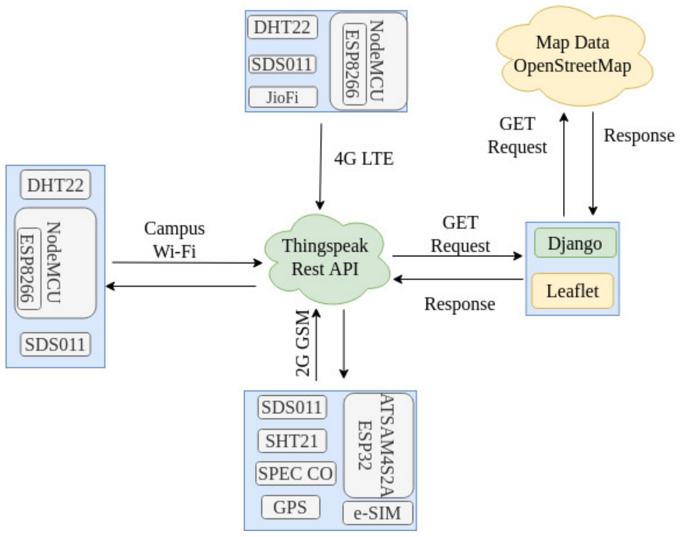


Figure 4.2: Data flow diagram of the air pollution monitoring sensor nodes

4.3.1 Network N_1

For the AirIoT nodes to publish sensor readings to ThingSpeak, the access points (AP) within the campus have separate wireless networks with SSID “esw19@iiith”. This network is created through a private virtual LAN granting low uplink data rate and access only to MAC bound devices. After attempting different attacks, the process narrowed down to two major attack scenarios: packet pipeline, a novel 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 DoS attack that disrupts the connection between the client and the Wi-Fi AP. It was implemented to impede data availability and can be used to tamper data.

4.3.1.1 Packet pipeline

The packet pipeline involves sniffing communication packets between the sensor node and ThingSpeak. Fig.4.3 shows the packet pipeline process. The tool airodump-ng [55] 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 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 cracking methods. Another vulnerability was discovered in the pipeline. The “proof of origin” of data from sensor nodes is not maintained or verified in any part

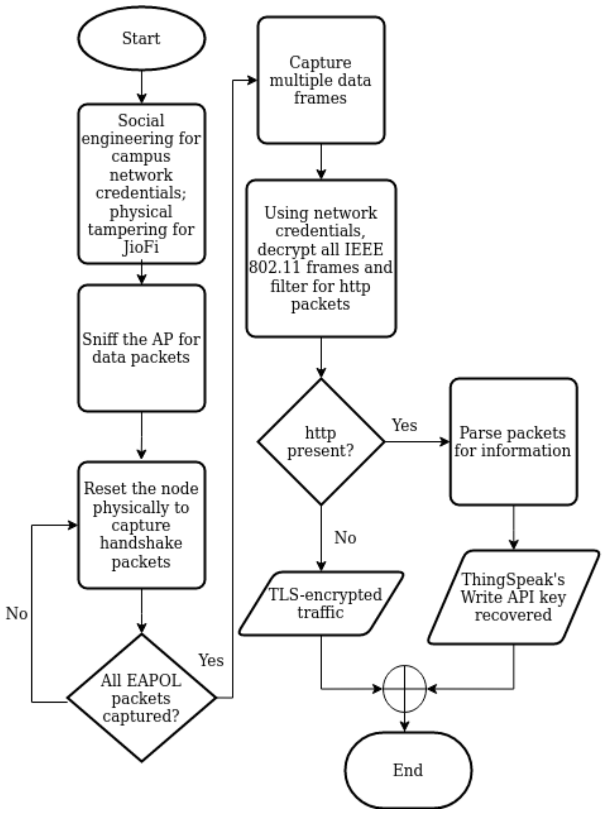

Start
Social engineering for campus network credentials; physical tampering for JioFi
Sniff the AP for data packets
Reset the node physically to capture handshake packets
All EAPOL packets captured?
No
Yes
Capture multiple data frames
Using network credentials, decrypt all IEEE 802.11 frames and filter for http packets
http present?
Yes
No
Parse packets for information
TLS-encrypted traffic
ThingSpeak's Write API key recovered
End

Figure 4.3: Packet pipeline attack

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

4.3.1.2 Deauthentication attack

Warwalking attack was performed using Kismet [56] 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. The node placed near the library within the IIIT-H campus was the target node and deauthentication attack was performed using its MAC address. The deauthentication considered the MAC address of only that particular node to prevent the disruption of network of other connected devices. This MAC address was given as an input to the tool MDK3. MDK3 [57] , 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.

4.3.1.3 Miscellaneous attacks

In the evil twin attack, a fake AP is created for the node to connect to, as described in fig.4.4. 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. One of the primary weaknesses of WEP is its reliance on a relatively short encryption key (usually 64 bits) and a flawed encryption algorithm RC4. These vulnerabilities make it possible to recover the encryption key by analyzing captured packets. Whereas, brute force attacks on WPA2 are considered computationally infeasible due to the strength of the encryption algorithm used. WPA2 employs a strong encryption method called AES (Advanced Encryption Standard) to secure wireless networks. The widespread recognition of WEP's weaknesses and its deprecated status further reduce the likelihood of devices supporting it by default. Executing a downgrade attack from WPA2 to WEP typically requires knowledge of the encryption keys used in the network. The attempt of degrading WPA2 to WEP was considered unsuccessful [58] in the current situation given the time it takes to crack WPA2 using brute force. In addition, three common vulnerabilities and exposures (CVEs) [59] 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.

4.3.2 Network N_2

For this network, spoofing of MAC addresses, physical tampering of nodes for access to credentials and deauthentication for DoS were performed on the interface between the ESP8266 and JioFi's Wi-Fi module. The experiment was conducted on an actual deployed live node placed within the IIIT campus after attaining the necessary permissions. This node was placed near the sports ground of the campus and required the attacking system to be in the proximity of the node. Like N_1, the deauthentication attack successfully disconnected the node, affecting data availability. Additionally, default passwords, a common vulnerability, were exploited to access the credentials to decrypt the captured packets through the 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.

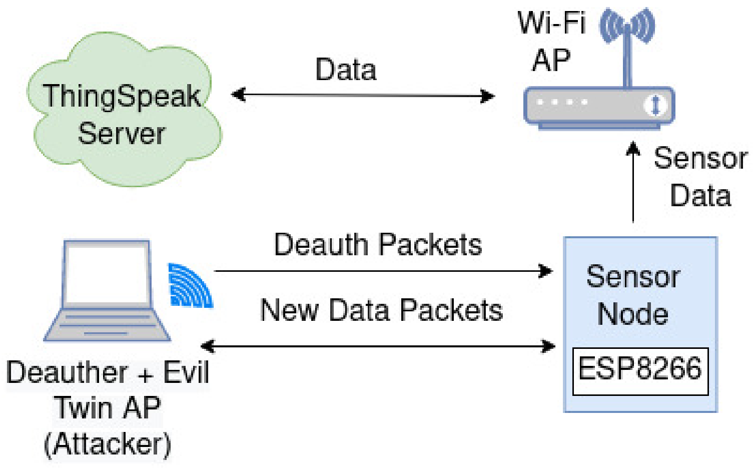
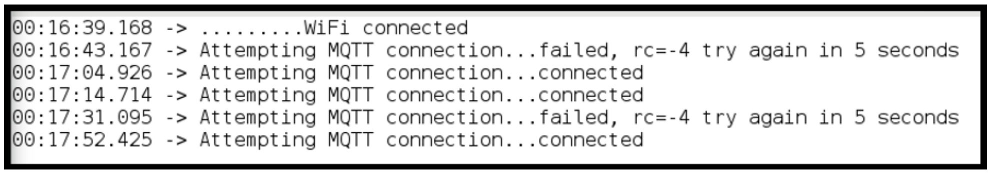


Figure 4.4: Setup for the evil-twin attack



```
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
```

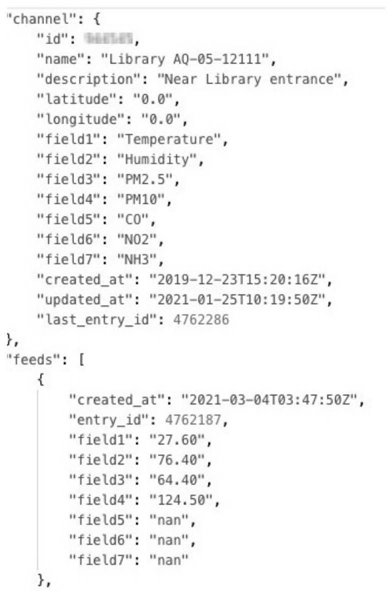
Figure 4.5: Effect of StealCon attack

4.3.3 Network N_3

The sensor nodes connected to this network have 2G GSM-based embedded SIM cards. Lack of authentication and encryption in 2G networks results in rogue base station attacks (IMSI catcher)[60]. However, the focus here was on the MQTT protocol that was being used by the nodes for communication with Thingspeak. It is to note that these node are placed outside the campus.

4.3.3.1 Analysis of the MQTT protocol

Protocol-level security was explored for the GSM-based nodes. One 2G GSM based node from the actual deployment was brought in and placed in close proximity to the PC running Airmon-ng to conduct packet analysis using packet pipeline approach. It was found that the 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 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.



```
"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"
    },
```

Figure 4.6: Postman response of the ThingSpeak API retrieved from the AJAX dashboard

4.3.3.2 StealCon attack

The protocol analysis 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.4.5. This attack was tested using two clients - A 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 data flow to the server from both sources.

4.3.4 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) [61]. 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 response to a simple GET request on Postman, as shown in fig.4.6. The channel ID in the Postman response has been blurred to maintain confidentiality.

4.4 Solutions and suggestions

Referring to the STRIDE methodology, Table 4.2 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.

4.4.1 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 this 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 can access the sensed data, it is end-to-end encrypted. Additionally, hashing can 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.

4.4.1.1 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.

4.4.1.2 Trade-offs

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

4.4.2 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 be completely avoided, it ensures a method to improve data availability.

Table 4.2: STRIDE analysis of the networks

Type of Network	$\mathbf{N_1}$	$\mathbf{N_2}$	$\mathbf{N_3}$
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; the 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

4.4.2.1 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.4.7 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.

4.4.2.2 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 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. It is to note that dynamic whitelisting of MAC addresses only adds a level of challenge to the attacker and may delay the attacker. This solution is not a mitigation against deauthentication attack.

4.4.2.3 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.

4.4.3 Protocol security

4.4.3.1 Security against MITM attacks

Packet replay attacks and packet injection can be averted if SSL/TLS is used instead, mainly through HTTPS instead of HTTP. This applies 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.

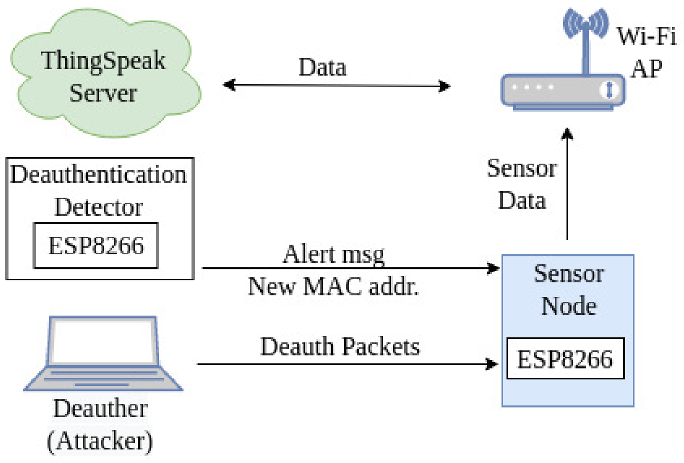


Figure 4.7: Deauthentication detector

4.4.3.2 Trade-offs with moving to more secure protocols

Multiple reasons exist 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 [62]. HTTPS was designed for the public internet and not to contain mechanisms, which becomes a scalability issue. The two protocols take much longer to communicate than their unsecured versions, giving rise to a response time versus security trade-off [63]. These two protocols also demand more resources and result in higher average power consumption; 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 [64]. It brings in a more real-time approach to the problem without involving the substantial lag HTTPS provides.

4.4.4 Dashboard security

Regarding the dashboard, the 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 [65] 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 proposed suggestion was to make this page private in the directory. Another suggestion would be to make the page accessible only within a private network.

4.5 Summary

This chapter covers the security analysis of pollution monitoring sensor network, AirIoT. With nodes placed both in and around the IIIT-H campus, connected via three different communication networks and using different communication protocols, the security assessments wary. WiFi, 4G and 2G GSM based networks are assessed and the potential threat are mapped to the STRIDE threat modeling framework. Two communication protocols, HTTP and MQTT are also analysed. Web crawling on the dashboard of AirIoT running on AJAX is conducted that exposes the ReadAPI keys of Thingspeak. Proof of concept solutions based on security assessments and recommendations are proposed to mitigate the revealed vulnerabilities.

Chapter 5

Security for oneM2M based smart city network

The oneM2M provides promising technical specifications for an interoperable and secure IoT/M2M system. This chapter focuses on the potential threats and their impact on oneM2M standard-based smart city deployments. Further, configurations for baseline security of the oneM2M open-source implementation, called eclipse OM2M, are presented. Recommendations are proposed based on actual on-ground tests conducted on OM2M-based smart city deployment of IIIT-H in India. The tests cover passive eavesdropping, performing replay attacks, brute force on credentials, denial of service, and analysis of access control policies.

5.1 Introduction

IoT-enabled smart city requirements include heterogeneity, interoperability, scalability, mobility, connectivity, and security. To satisfy the need for a common platform supporting these requirements, oneM2M [5] proposes a common middleware technology in a horizontal layer covering architecture, API specifications, security solutions, and interoperability for M2M/IoT technologies. There are multiple oneM2M implementations [66] such as Mobius, Open-source Architecture Semantic IoT Service-platform project (OASIS), C-DOT Common Service Platform (CCSP), and eclipse OM2M. The eclipse OM2M [6] implements oneM2M and the smartM2M standard. It is an open-source project under the eclipse technology project.

Much work has been done on IoT/M2M systems and their security. Authors in [67] present a survey on the security requirements of mission-critical IoT applications, vulnerabilities, and sources of threats, culminating work on mitigation strategies for emerging applications. The work also presents the integration of blockchain with IoT for enhancing security. Proposing recommendations through consultations from stakeholders, the work in [26] focuses on challenges with IoT, threats, attack scenarios and possible mitigation strategies. Further ETSI standards on consumer IoT security [36] and [68] cover security and privacy best practices for consumers and manufacturers of IoT devices. Particularly in the case of IoT-enabled smart cities, [10] provides a detailed analysis of the threats and vulnerabilities to AirIoT, an air quality monitoring smart city set up. They model the threats using STRIDE modelling framework followed by solutions with respective trade-offs. In the context of M2M/IoT security, the oneM2M technical specifications document, TS-0003, [69] mentions security provisions and procedures on access control policies (ACP), dynamic authorization, application/device impersonation prevention, and privacy protection. The paper [70] implements security in the OS stack, Mbed OS. Their implementation to enable secure end-to-end communications for IoT devices involves realizing the oneM2M specified Security Association Establishment Framework (SAEF). On the same MbedOS, the authors [71] show the implementation of secure MQTT binding as per [72] of the oneM2M technical specification. Both these works incorporate key security provisions from the standard into their implementations. To address the privacy and resource access management requirements, [73] proposes a "Privacy_Enforcement" plugin.

This work focuses on the security analysis of OM2M based implementation of IoT network deployed at IIIT-H having more than 200 nodes for several smart city applications. First, the potential threats and attacks are modeled on oneM2M standard compliant implementations using STRIDE methodology [9]. Second, five security analyses are performed on this network: eavesdropping, brute force attack for OM2M credentials, authorization via access control policies, scalability requirements, and a packet replay attack. Third, solutions are recommended based on the above analysis comprising of configuring OM2M provisions to ensure baseline security of smart cities. To the best of my knowledge, this kind of security analysis has not been done for an actual OM2M-based smart city deployment to date.

The work is structured as follows: Section 2 describes the oneM2M standard, followed by Section 3 that focuses on modeling the major potential threats to any oneM2M standard-based system. Section 4 describes the OM2M platform and its IIIT-H smart city implementation. Next, Section 5 presents a security analysis of OM2M followed by recommendations for baseline security in OM2M-based smart city proposed in Section 6. Finally, Section 7 concludes the work and mentions the future scope of research in this work.

5.2 oneM2M standard

The oneM2M is a global standard initiative led by eight national standardization bodies and various industries aiming to provide an interoperable horizontal platform for building vertically scalable applications in the IoT paradigm. It provides the technical specifications which cater to the requirements needed by a common M2M service layer. The common service layer is incorporated into varied hardware and software, interconnecting all types of devices in the field with the M2M application servers worldwide. All IoT components are brought together in a solution stack using oneM2M standard.

5.2.1 Architecture

The functional architecture of oneM2M comprises application entities (AEs), interworking proxy entities (IPEs), common service entities (CSEs), and network service entities (NSEs) as shown in Fig.

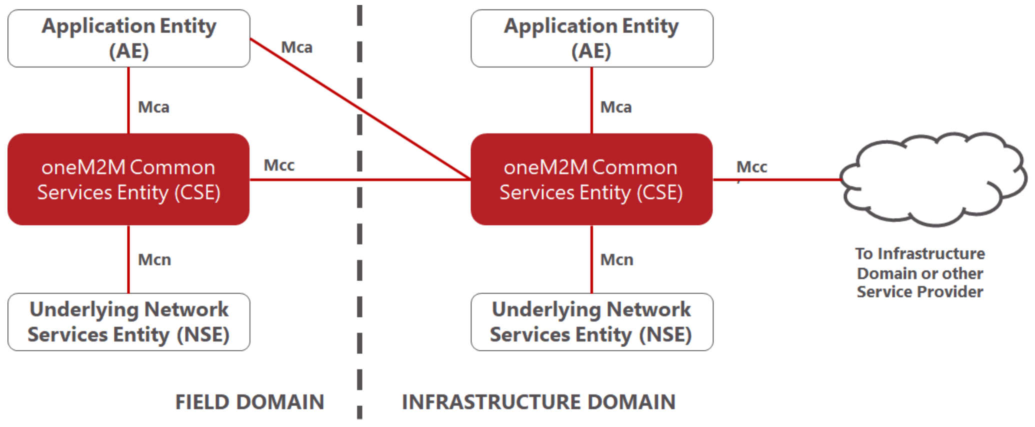

Application Entity (AE)
Mca
Mca
oneM2M Common Services Entity (CSE)
Mcc
Mcn
Underlying Network Services Entity (NSE)
Application Entity (AE)
Mca
oneM2M Common Services Entity (CSE)
Mcc
Mcn
Underlying Network Services Entity (NSE)
To Infrastructure Domain or other Service Provider
FIELD DOMAIN
INFRASTRUCTURE DOMAIN

Figure 5.1: oneM2M architecture

5.1 [74]. Application entities deal with the application layer in the IoT setup, residing in the sensors and communicating with the M2M service layer using RESTful (REST) APIs. IPE aims to provide an interface for non-oneM2M devices to communicate with the service layer. The common service entities provide the common service functions (CSFs). These include registration, discovery, data management, and security. Network service entities manage communications for services such as device triggering, small data transmission, location notification, and location queries. The oneM2M System has logical entities called nodes which typically contain CSEs and/or AEs. The nodes mainly have two categories, the field domain, and the infrastructure domain. The "Field Domain" comprises sensors, actuators, and gateways, while the "Infrastructure Domain" handles all the servers and applications on larger computers. The entities communicate using OneM2M reference points such as Mca (AE-CSE communication), Mcc (inter-CSE communication), Mcn (CSE-NSE communication).

5.3 oneM2M for smart city: Threats and vulnerabilities

Technical report [75] gives a high-level overview of security threats and countermeasures for oneM2M-based systems. The document briefly explains the security services of oneM2M, security requirements, threats to oneM2M systems, and suitable mitigation recommendations. This section presents Table 5.1 describing the potential security threats and vulnerabilities in an oneM2M based smart city-centric implementation. Additionally, security provisions of [69], and technical specifications [76] are modeled using the STRIDE threat modeling framework. STRIDE is an acronym for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. STRIDE is preferred over other frameworks such as PASTA, OWASP, and MITRE Att&ck as it is a product and development-centric method of assessing the threats. Each STRIDE element with implications on oneM2M systems is defined below.

- **Spoofing:** It is a technique where cyber attackers impersonate legitimate sources to manipulate communications, access sensitive personal data, and modify policies. In the context of oneM2M, spoofing leads to compromising passwords, cryptographic keys, and CSE and AE node identifiers. The secure association establishment procedure of oneM2M provides mutual authentication mechanisms to counter these threats.

- **Tampering:** This is a process by which an attacker violates integrity and authorization. With tampering, the attacker can modify a system, component, intended function, or data as a consequence of intentional but unlawful conduct. This includes physical tampering of sensor nodes, communication channels, primitives, the oneM2M service capability layer, and oneM2M system dependencies for a smart city setup.

- **Repudiation:** This occurs when an application or system fails to provide facilities to monitor and log user activities, allowing for malicious modification or falsifying the identification of new actions. Similar to how spoofing mail messages are used, its use may be expanded to include generic data manipulation under others' names. In the event of this assault, the information recorded in log files may be deemed false or deceptive. Logging of user activity and system procedures can prevent violation of non-repudiation. oneM2M systems can log primitives, messages between entities, and user profiles.

- **Information disclosure:** When an application or website makes sensitive information available to unauthorized users, it is called information disclosure (also known as information leaking). Websites may reveal any information to a prospective attacker depending on the context, which includes information about other users, including their usernames or financial data, sensitive business or commercial data, the architecture of the website, and its technical specifications. Further, unsecured communication protocols, such as HTTP and MQTT, can reveal sensitive information. Secure environment proposed by oneM2M assists with sensitive data storage and sensitive function execution.

- **Denial of service:** An attacker aims to bring down a computer system or network so its intended users cannot access it. DoS attacks achieve this by providing the victim with excessive traffic or information that causes a crash. Both times, the DoS attack denies the service or resource that legitimate users expected. Various attacks can violate the availability of data and services, such as buffer overflow attacks, ICMP floods, and SYN floods. DoS is possible in oneM2M systems by overwriting the limits of buffers, multiple REST API requests to resource trees, and sending unsupported data formats.

- **Elevation of privilege:** With privilege escalation, an attacker gets the system's administrative, root, or higher privileged rights. It leads to an authorization violation and impacts the security of the common service functions on oneM2M. Misconfigurations, unnecessary open ports, and weak authentication processes can result in this threat.

Table 5.1: STRIDE analysis of oneM2M

STRIDE	Potential threats to oneM2M standard based implementations	Existing security provisions in oneM2M
Spoofing	• AE impersonation • Broken authentication • Session hijacking	• AE impersonation prevention • Authentication mechanisms: Symmetric key-based security, Certificate-based security, Generic Bootstrapping Architecture (GBA) framework
Tampering	• Corrupted service layer software • Unauthorized access to oneM2M software dependencies • Alteration of primitives transmitted over the Mca/Mcc/Mcc' reference points - • Physical tampering of nodes	• "m2m:software update" provides consistent updates to the end nodes to patch the security vulnerabilities • "m2m:DeviceID" uniquely identifies a device using a URN • Device certificates to authenticate the AEs or CSEs
Repudiation	• Unavailability of access logs • Log injection/tampering/forging	• Token based authorization • Role based access control • "m2m:logStatus", "m2m:logTypeId" to check the log status of the event management resource
Information Disclosure	• Insecure communication protocols • Replay of M2M primitives between entities • Device hijacking • Network manipulation attacks • Exposed sensitive data in AE or M2M gateways	• oneM2M protocol bindings • ESPrim • Secure environment plug-in
Denial of Service	• Buffer overflow • Flooding of RestAPI requests	• "m2m:accessControlRule" (with assigning access using m2m:ipv4, m2m:ipv6 and m2m:locationRegion) • Mutual authentication through SAEF
Elevation of Privilege	• Privileged insider attack • Access mechanism violation • Insecure cryptographic storage • Exposed Long-Term Service-Layer Keys	• "m2m:authorizationStatus" provides status of access control policies • Dynamic authorization for token based temporary permissions

5.4 OM2M implementation at IIIT-H

This section studies the OM2M platform and its integration with the smart city setup. Various experiments are conducted, and observations are made to understand the platform's security provisions and effectiveness in catering to the security requirements. The experiments are conducted on an actual OM2M-based dense IoT deployment of IIIT-H in India.

5.4.1 About OM2M

OM2M [9] stands for open-source M2M service platform, which is in line with ETSI M2M and oneM2M standard. OM2M, a part of the eclipse IoT working group, consists of service capability layers (SCL) which are highly extensible via plugins that provide various functionalities. OM2M is built on top of modular OSGi architecture and provides RESTful APIs for all the services on its platform. It enables multiple communication protocols binding (HTTP, HTTPS, COAP, MQTT), reuse of existing remote devices management mechanisms, and inter-working with existing legacy devices. OM2M is extensible via plugins. For example, the Jetty plugin may be activated to offer HTTPS as an additional layer of protection. Similarly, to use the MongoDB database, the home persistence MongoDB plugin can be activated.

5.4.2 Smart city at IIIT-H

Fig. 5.2 shows the current smart city deployment based on OM2M. There are currently more than 200 nodes deployed, covering an area of 66 acres in and around IIIT-H. The applications covered include air and water quality, energy and weather monitoring, smart room (air conditioning, occupancy, air quality, energy monitoring), and smart campus applications (smart street lamps). Each of these nodes uses a different network, namely Wi-Fi, 4G, Wi-Sun, and LoraWAN, to send data to the OM2M server, which is then used to send the data to the data warehouse using the subscription method. The data stored in the data warehouse is utilized for various purposes depending on the application type. The smart city dashboard [77], smartphone applications, Alexa interface, and home automation access the data from the warehouse. The general user must utilize the Indian Urban Data Exchange (IUDX)[78] to view the data. IUDX requires user self-registration to obtain a token. With the token, the user can view data from the OM2M server.

5.5 Security analysis of OM2M

This section present observations made on the default security settings of OM2M at IIIT-H based on five experiments. Each of these experiments provides visibility into the communication protocol used, data formats in place, configured authorization and authentication mechanisms, and the overall OM2M

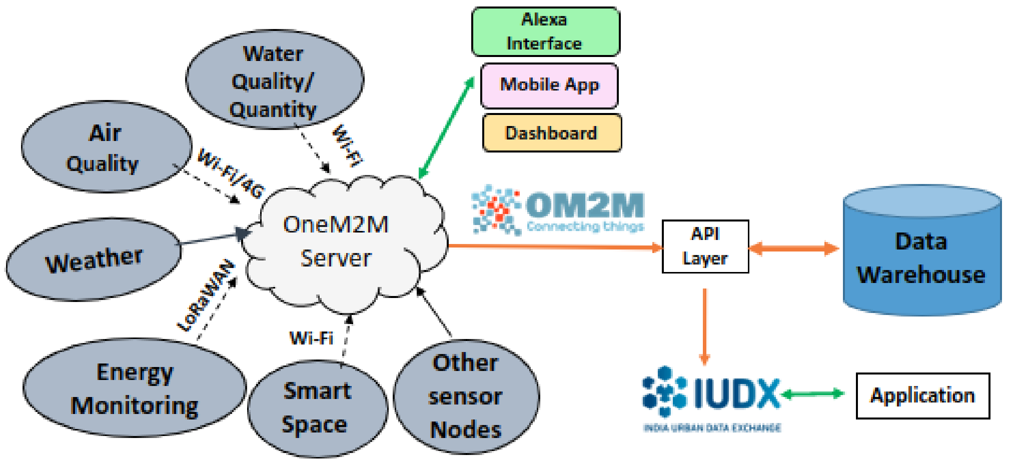


Figure 5.2: OM2M based smart city deployment

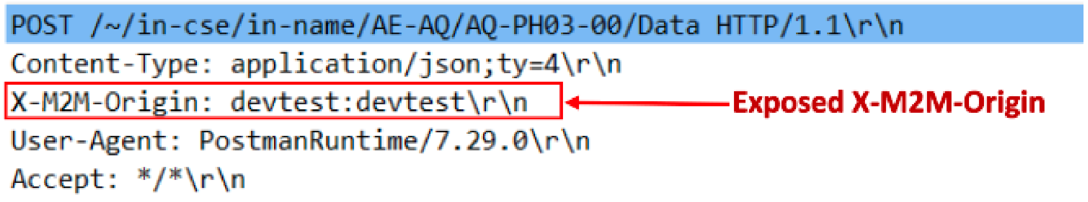
architecture. As a fundamental step before starting the experiments, a dedicated Linux based PC was set up as the OM2M server following the steps mentioned by the Eclipse foundation.[79]

5.5.1 Eavesdropping attack

Eavesdropping was performed using Wireshark to intercept the communication between the AE and OM2M server. For the experiment, two PCs, one with Windows OS and the other with Linux OS, were used. The Windows OS-based PC had two software running. Postman, which is the API-building software, created OM2M-specific APIs and pushed fabricated sensor readings onto the OM2M server for testing. The second software was Wireshark, used for packet analysis. On the Linux OS-based PC, OM2M server was set up. Wireshark was run in promiscuous mode in the system, pushing the data. Multiple packets appeared after capturing packets using Wireshark and filtering for HTTP request and response packets. Fig. 5.3 shows the one of the HTTP based packets and its contents after performing a passive eavesdropping on the network. The following observations were made:

- The standard provides plugin support for communication via COAP, HTTP, MQTT, and Websocket. By default, OM2M uses HTTP as the communication protocol. This default configuration is not secure and thus exposes the header and payload.
- X-M2M-Origin: This username:password pair is used as the authenticator to manage the resource tree. The X-M2M credentials are used across all the nodes in the network and need to be added to all the API requests made in OM2M setup. This parameter is assigned by the request originator which maybe a CSE or AE.

The attacker can easily create custom requests with this header value, which can severely impact the entire infrastructure.



```
POST /~/in-cse/in-name/AE-AQ/AQ-PH03-00/Data HTTP/1.1\r\n
Content-Type: application/json;ty=4\r\n
X-M2M-Origin: devtest:devtest\r\n
User-Agent: PostmanRuntime/7.29.0\r\n
Accept: */*\r\n
```

Figure 5.3: Confidential X-M2M-Origin visible in HTTP packet on Wireshark

5.5.2 Brute force attack for OM2M credentials

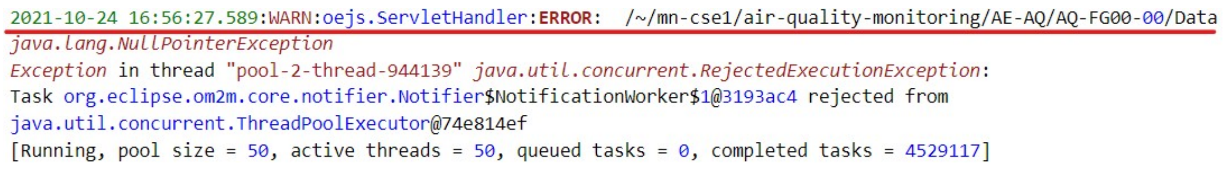
In order to access the entries of the resource tree, users must login to the OM2M login page using credentials assigned to them by the admin. Users and roles are authorized through ACPs in OM2M. With access to those credentials, an attacker can manipulate the entire resource tree and change the admin credentials, resulting in a DoS on OM2M administrators. Brute force, dictionary attacks and social engineering can be used to obtain the credentials. It is found that there is a possibility of launching wide-scale attacks by several thousand devices (botnets) or attempting many passwords on the single OM2M server. There is no mitigation mechanism such as blocking the attacker's IP address. Further, there is a limit on the type of special characters allowed for the passwords making the brute force attack easier. Passwords with characters such as '(', ')', '[', ']' were not allowed.

5.5.3 Authorization via access control policies

To access the setup, the configuration file in the default OM2M server stores two kinds of user profiles, admin and guest. These two profiles can log into the OM2M setup, with the admin capable of performing CRUD operations on all the entities in the setup, while the guest can only view the resource tree. The resource tree stores different entities, such as the AE, IPE, and ACP. ACP in the OM2M setup issues different application entities with certain Access Control Operations (ACORs), such as create-1, retrieve-2, update-4, delete-8, notify-16, and, discover-34. Another user with "devtest" as the username and password was created for the experiment. For the user, CRUD operation 34 was set as its ACOR. With these new credentials, any user could view the resource tree on the OM2M platform. It was noticed that the admin credentials could no longer be used to view the resource tree, but the credentials continue to control all the CRUD operations.

5.5.4 Scalability requirements

Scalability is an essential requirement for smart cities. OM2M comes with the H2 database acting as the default database to store records collected in the resource tree. It was discovered that the server started logging null point exceptions and locking objects upon an overload of requests from IoT nodes. Therefore to handle the scalability issues, the database was migrated from the high-speed in-memory H2 database to MongoDB, an auto-scalable production development NOSQL database. This database



```
2021-10-24 16:56:27.589:WARN:oejs.ServletHandler:ERROR:  /~/mn-cse1/air-quality-monitoring/AE-AQ/AQ-FG00-00/Data
java.lang.NullPointerException
Exception in thread "pool-2-thread-944139" java.util.concurrent.RejectedExecutionException:
Task org.eclipse.om2m.core.notifier.Notifier$NotificationWorker$1@3193ac4 rejected from
java.util.concurrent.ThreadPoolExecutor@74e814ef
[Running, pool size = 50, active threads = 50, queued tasks = 0, completed tasks = 4529117]
```

Figure 5.4: H2 database server crash - a scalability issue

change helped solve the server crash issues, thus preventing a potential DoS attack. Fig. 5.4 shows 944139 threads. With 200+ nodes deployed, each node tries to send the data to the server for which it creates threads and keeps them in the thread pool. The exception arises when the pool size and the number of active threads are equal. At this point, the writelock waits for a period of 600 seconds, exceeding which the thread gets locked with all its upcoming container entities. The server throws a ServletHandler Error which cause the NullPointerException and RejectedExecutionException cases to be resolved.

5.5.5 Packet replay attack

A replay attack is when an attacker intercepts (sniffing, eavesdropping) the packets from the client to the server, delaying or resending the packets at different intervals. This attack is successful on systems that do not have the means to differentiate between the source of the various API requests the servers receive.

For this demo experiment, MITM Proxy [80], a Kali Linux-based cybersecurity tool for penetration testing and replaying web traffic was used. The tool helps intercept HTTP and HTTPS requests and replay messages.It is important to note that all nodes within the campus network use HTTP connections, while the nodes outside the campus network require HTTPS to connect to the server. Before the experiment, ESP8266 with SDS011 sensor sends sensed data to the OM2M server. The sensed data is sent every 5 seconds, thus creating multiple content instances in the resource tree. This resource tree is used as a reference for observing the experiment's results. The experiment proceeds as follows:

- Postman created a post request and sent a random value (in this case, value 30 was sent) to the OM2M server
- This request gets intercepted by the MITM proxy
- MITM proxy keeps track of other requests from the client side (Postman), and replays the captured packet at a different time interval.

The creation of a new content instance with the value defined by the postman was observed in the reference resource tree, indicating the server has no provisions to check the authenticity of any API request. In the case of an HTTPS connection, CA certificates need to be added to the client nodes. Since the smart city nodes are deployed around the city, physical tampering is feasible. It was observed that the OM2M server could not identify the fake CA certificate installed on the client end. The same fake

CA certificates issued by the MITM proxy were immediately flagged as fake in other popular websites. OneM2M security solutions document provides provisions such as X-M2M-RT (request timestamp) to mitigate such attacks. Other provisions provided in the standards, such as establishing sessions with a fixed session time, can also be used.

5.6 OM2M security recommendations

Following the observations based security experiments, this section presents recommendations which are mapped to each STRIDE element. Appropriate formats from oneM2M specification on service layer core protocol [76] are also recommended for mitigating STRIDE. This document specifies a list of common data formats, interfaces, and message sequences to be used by developers.

5.6.1 Spoofing

In the case of spoofing or impersonation, oneM2M technical report[75] mentions ways to mitigate this threat using a secure communication link or Role Based Access Control (RBAC). In OM2M, HTTPS binding encrypts the headers, node identifiers, and timestamps. M2M node identifier (m2m:NodeID) data format, AE-ID and CSE-ID mitigates impersonation and replay attacks. Using these data formats provides baseline security and does not require rebuilding the entire authentication mechanism of the deployment.

5.6.2 Tampering

Illegitimate modification of the ContentInstance is detected by observing the discrepancy in the resource tree's creation time (CT) and last modified time (LT) indicate. Data format X-M2M-OT, a HTTP header parameter, can be used to indicate originating timestamp of request and response. It helps mitigate possible packet injection and similar man-in-the-middle attacks. ACPs prevent intentional or unintentional tampering of the resource tree. Based on the analysis of the implemented ACPs, it was observed that the attacker can never obtain the admin credentials by guessing them on the server login page. Thus as a solution, access to view the resource tree and use of CRUD operations are decoupled. This helps mitigate against the brute force attack on the server login page. As shown in Fig. 5.5, the attacker must perform two steps to obtain all the previous privileges.

- In step 1, the attacker would first need to obtain the devtest credentials to view the resource tree with all the required information to launch the attack on specific targets can be obtained.
- In step 2, the attacker would need to obtain the admin credentials, with which the CRUD operations would be granted. At this stage, the attacker has both view access to the resources and the CRUD operations needed to modify any resource. Obtaining the credentials via standard techniques like brute force and dictionary attacks is indeterminate and tedious.

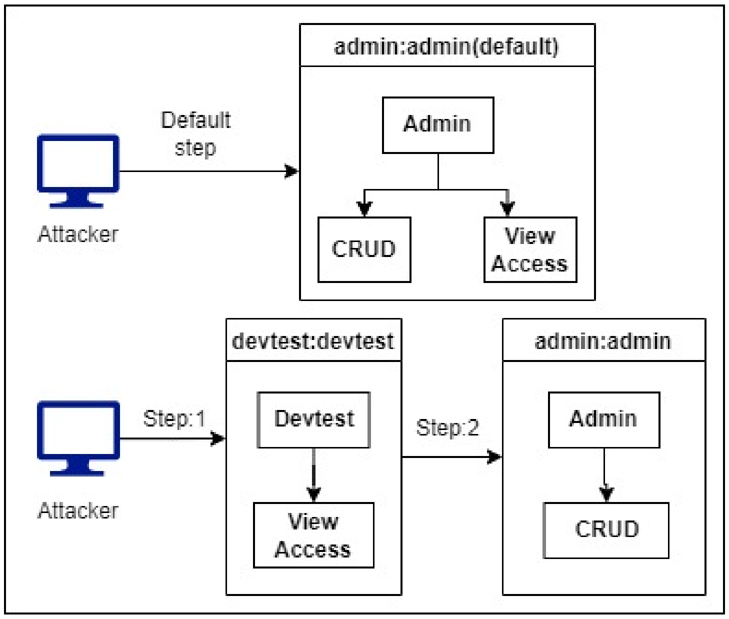


Figure 5.5: Mitigation against brute-force attack

5.6.3 Repudiation

In the case of repudiation, provisions for tracking and logging users' actions have been implemented by OM2M server. The "m2m:logStatus" can be utilised for additional update on the logging activity.

5.6.4 Information disclosure

ACPs are implemented to authorize entities for data retrieval and manipulation. In addition to enabling HTTPS to mitigate this scenario, separate X-M2M credentials were created for all the verticals present such as air, water, and energy, thereby restricting the scope of the attack. SAEF can be established to mutually authenticate the MQTT Client and MQTT Server when using the MQTT protocol. When dealing with brute force attacks, locking of user account after failed attempts is a must.

5.6.5 Denial of service

In an attempt to mitigate the DoS attack, modifications were made both at the node end and the server end. At the node end, two buffers were created: primary and secondary to handle the issues of network failure ensuring resiliency. On network restoration the requests would be sent to the server. Additionally, the nodes are MAC, and IP bound, thus a white list of permitted IP and MAC addresses is implemented. Dynamically allocating IP and MAC addresses can make it more challenging for an attacker to specifically target that device for deauthentication. However, skilled attackers can still scan the network and identify the new MAC address associated with the device. This method only creates a short delay in the attack but it is not a foolproof defense against deauthentication attacks. Based on a scalibility assessment at the OM2M server end, MongoDb has been implemented to prevent a potential DoS attack.

5.6.6 Elevation of privilege

The status of the resource authorization can be known using provisions such as “m2m:authorizationStatus”, “m2m:operationMonitor” and “m2m:accessControlRule”.

5.7 Summary

The oneM2M standard for IoT/M2M has multiple implementations. OM2M, an open source M2M platform is an implementation of the oneM2M standard and has functions extensible via plugins. This chapter presents an analyses of the security provisions of oneM2M and OM2M. Initially, numerous potential threats and vulnerabilities to any oneM2M implementation and the existing provisions within the standard are modeled to each element of the STRIDE threat modeling framework. Experiments such as passive eavesdropping, packet replay and, brute force are conducted on an actual OM2M-based smart city deployment to assess the existing default configurations of the OM2M platform. To help enhance the baseline security of oneM2M-based implementations for smart cities, configurations and provisions from the oneM2M standard are presented.

Chapter 6

Concluding remarks

6.1 Conclusions

The research carried out in this thesis highlights the security requirements for IoT-enabled smart cities. IoT plays a major role in smart city solutions. Securing the IoT ecosystem is complex and challenging as requirements depend on various factors. The requirements were explored by taking a deep dive into the work done in the domain of IoT security, smart cities and by conducting on-ground analysis with verification of an actual smart city deployment of IIIT-H.

Chapters 4 and 5 provide the procedures, experiments, analysis, and solutions. A security analysis of the AirIoT deployment using IoT to monitor pollution was done in 4. The protocol and network security of three types of communication networks were examined. The STRIDE framework is used to model the entire analysis. To increase data availability during deauthentication assaults, a novel proof-of-concept method is put forth. It implements a workaround for such attacks. Also, the dashboard's security is assessed, and additional security precautions are advised. 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. The oneM2M standard and the OM2M platform are studied in chapter 5. This chapter analyses the security provisions of oneM2M and its implications for smart cities. Using the STRIDE threat modeling framework, a wide range of potential risks and vulnerabilities to oneM2M implementations are investigated and modeled. The impact of these vulnerabilities on a real OM2M-based smart city deployment based on the platform's current default configurations is investigated through experiments. The appropriate elements are suggested for smart city baseline security based on the STRIDE framework.

6.2 Future scope

IoT security has immense research potential. Contributions presented in this thesis can be extended to create test beds to test and manage security provisions of smart city applications. Future research could focus on developing advanced machine learning algorithms that can identify security threats to smart cities through predictive analysis and help automate mitigation procedures. Further, due to the huge amount of data involved in the IoT ecosystem, developing privacy-preserving algorithms and protocols to ensure user data confidentiality while generating insights is another area requiring work. Over the years, blockchain-based solutions for improving transparency and security have emerged. The use of blockchain and distributed ledger technology can be explored as solutions for securing critical infrastructure and data in smart cities. Yet another important area of work is developing standardized security protocols and guidelines that can be used across different IoT platforms.

Related Publications

Relevant Publications:

- **G.V. Ihita**, K.S. Viswanadh, Y. Sudhansh, S. Chaudhari and S. Gaur, “Security Analysis of Large Scale IoT Network for Pollution Monitoring in Urban India”, *IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021
- **G.V. Ihita**, Vybhav.K. Acharya, Likhith Kanigolla, S. Chaudhari and Thierry Monteil, “Security for oneM2M-Based Smart City Network: An OM2M Implementation”, *15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2023

Other Publications during the study:

- S. K. A. Kumar, **G. V. Ihita**, S. Chaudhari and P. Arumugam, “A Survey on Rural Internet Connectivity in India”, *14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2022

Bibliography

[1] David Norris. *The Internet of Things*. Mc Graw Hill Education, 2015.

[2] Arshdeep Bahgya and Vijay Madisetti. *Internet of Things: A Hands-on Approach*. Universities Press, 2015.

[3] Raj Kamal. *Internet of Things: Architecture and Design Principles*. Mc Graw Hill Education, 2017.

[4] Perry Lea. *Internet of Things for Architects*. Birmingham, UK: Packt Publishing Ltd, 2018.

[5] *oneM2M*. `https://www.onem2m.org/`. [Online] Accessed: 15-Mar-2021.

[6] *Eclipse OM2M*. `https://www.eclipse.org/om2m/`. [Online] Accessed: 25-August-2022.

[7] *IIIT Hyderabad*. `https://www.iiit.ac.in/`. [Online] Accessed: 25-August-2022.

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

[9] *Microsoft Security. 2007. STRIDE chart - Microsoft Security*. `https://www.microsoft.com/security/blog/2007/09/11/stride-chart/`. [Online] Accessed: 15-Mar-2021.

[10] G.V. Ihita et al. “Security Analysis of Large Scale IoT Network for Pollution Monitoring in Urban India”. In: *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*. 2021, pp. 283–288.

[11] G.V. Ihita et.al. “Security for oneM2M-Based Smart City Network: An OM2M Implementation”. In: *2023 15th International Conference on COMmunication Systems NETworkS (COMSNETS)*. 2023, pp. 808–813.

[12] C. C. Sobin. “A Survey on Architecture, Protocols and Challenges in IoT”. In: *Wireless Personal Communications* 112 (2020).

[13] *Internet of Things Global Standards Initiative*. URL: `https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx`.

[14] *The Internet of Things at the IETF*. URL: `https://www.ietf.org/topics/iot/`.

[15] *Internet Of Things (IoT)*. URL: `https://www.gartner.com/en/information-technology/glossary/internet-of-things`.

[16] Luigi Atzori, Antonio Iera, and Giacomo Morabito. “The Internet of Things: A Survey”. In: *Computer Networks* (Oct. 2010), pp. 2787–2805.

[17] Deepika Navani, Sanjeev Jain, and Maninder Singh Nehra. “The Internet of Things (IoT): A Study of Architectural Elements”. In: *2017 13th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*. 2017, pp. 473–478.

[18] R. Porkodi and V. Bhuvaneswari. “The Internet of Things (IoT) Applications and Communication Enabling Technology Standards: An Overview”. In: *2014 International Conference on Intelligent Computing Applications*. 2014, pp. 324–329.

[19] AbdelRahman H Hussein. “Internet of things (IoT): Research challenges and future applications”. In: *International Journal of Advanced Computer Science and Applications* 10.6 (2019).

[20] Jan Markendahl et al. “On the role and potential of IoT in different industries: Analysis of actor cooperation and challenges for introduction of new technology”. In: *Internet of Things Business Models, Users, and Networks*. 2017, pp. 1–8.

[21] Li Da Xu, Wu He, and Shancang Li. “Internet of Things in Industries: A Survey”. In: *IEEE Transactions on Industrial Informatics* 10.4 (2014), pp. 2233–2243.

[22] Ravi Kishore Kodali, Govinda Swamy, and Boppana Lakshmi. “An implementation of IoT for healthcare”. In: *IEEE Recent Advances in Intelligent Computational Systems (RAICS)*. 2015, pp. 411–416. DOI: `10.1109/RAICS.2015.7488451`.

[23] Shruthi Koratagere Anantha Kumar et al. “A Survey on Rural Internet Connectivity in India”. In: *2022 14th International Conference on COMmunication Systems NETworkS (COMSNETS)*. 2022, pp. 911–916.

[24] Xi Chen et al. “Integration of IoT with smart grid”. In: *IET International Conference on Communication Technology and Application (ICCTA 2011)*. 2011, pp. 723–726. DOI: `10.1049/cp.2011.0763`.

[25] Asma Haroon et al. "Constraints in the IoT: the world in 2020 and beyond". In: *International Journal of Advanced Computer Science and Applications* 7.11 (2016).

[26] "ENISA Baseline Security Recommendations for IoT". In: ([Online] Accessed: 11-Mar-2021). https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot/.

[27] Pintu Kumar Sadhu, Venkata P. Yanambaka, and Ahmed Abdelgawad. "Internet of Things: Security and Solutions Survey". In: *Sensors* 22.19 (2022).

[28] Fadele Ayotunde Alaba et al. "Internet of Things security: A survey". In: *Journal of Network and Computer Applications* 88 (2017), pp. 10–28. ISSN: 1084-8045.

[29] Phillip Williams et al. "A Survey on Security in Internet of Things with a Focus on the Impact of Emerging Technologies". In: *Internet of Things* 19 (July 2022), p. 100564.

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

[31] "ENISA Threat Landscape 2022". In: ([Online] Accessed: 15-Dec-2022). https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022.

[32] *Threat Modeling: A Summary of Available Methods*. https://apps.dtic.mil/sti/citations/AD1084024. [Online] Accessed: 09-Sept-2022.

[33] Vikas Hassija et al. "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures". In: *IEEE Access* 7 (2019), pp. 82721–82743.

[34] *ISO/IEC 27001*. https://www.iso.org/home.html. [Online] Accessed: 15-Dec-2022.

[35] *NIST Cybersecurity Framework*. https://www.nist.gov/cyberframework. [Online] Accessed: 15-April-2022.

[36] "ETSI EN 303 645 Cyber Security for Consumer Internet of Things: Baseline Requirements ". In: (Accessed 19-Dec-2021). https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf.

[37] *GSMA IoT Security Guidelines and Assessment*. https://www.gsma.com/iot/iot-security/iot-security-guidelines/. [Online] Accessed: 15-April-2022.

[38] *Recommendations for IoT / M2M Security*. https://www.tec.gov.in/M2M-IoT-technical-reports. [Online] Accessed: 25-August-2021.

[39] *F5 Labs. 2019. What Is The CIA Triad?* https://www.f5.com/labs/articles/education/what-is-the-cia-triad. [Online] Accessed: 15-Feb-2021.

[40] *Top-10-growing-smart-cities.* https://www.asme.org/topics-resources/content/top-10-growing-smart-cities. [Online] Accessed: 25-August-2021.

[41] *Smart Cities Mission.* https://smartcities.gov.in/. [Online] Accessed: 25-August-2021.

[42] *Smart City Living Lab.* https://smartcityresearch.iiit.ac.in/. [Online] Accessed: 25-August-2022.

[43] Reem Al Sharif and Shaligram Pokharel. "Smart City Dimensions and Associated Risks: Review of literature". In: *Sustainable Cities and Society* 77 (2022), p. 103542. ISSN: 2210-6707.

[44] Tai-Hoon Kim, Carlos Ramos, and Sabah Mohammed. "Smart City and IoT". In: *Future Generation Computer Systems* 76 (2017), pp. 159–162. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2017.03.034.

[45] Chai Toh. "Security for Smart Cities". In: *IET Smart Cities* 2 (July 2020).

[46] *ISO/IEC TS 27570:2021.* https://www.iso.org/standard/71678.html.

[47] *Cyber security for Smart Cities.* https://www.enisa.europa.eu/publications/smart-cities-architecture-model/.

[48] *ITU SG20: Internet of things (IoT) and smart cities and communities (SCC).* https://www.itu.int/en/ITU-T/studygroups/2022-2024/20/Pages/default.aspx. [Online] Accessed: 31-Oct-2022.

[49] *ITU SG17 - Security.* https://www.itu.int/en/ITU-T/studygroups/2022-2024/17/Pages/default.aspx. [Online] Accessed: 30-Oct-2022.

[50] Anwaar AlDairi and Lo'ai Tawalbeh. "Cyber Security Attacks on Smart Cities and Associated Mobile Technologies". In: *Procedia Computer Science* 109 (2017), pp. 1086–1091. ISSN: 1877-0509.

[51] Luo et.al. "On the Security and Data Integrity of Low-Cost Sensor Networks for Air Quality Monitoring Sensors". In: *Sensors 2018* ().

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

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

[54] *ThingSpeak*. `https://thingspeak.com/`.

[55] *Airodump-ng*. `https://www.aircrack-ng.org/doku.php?id=airodump-ng`.

[56] *Kismet*. `http://www.kismetwireless.net/`.

[57] *MDK3*. `https://tools.kali.org/wireless-attacks/mdk3`.

[58] Abdullah Mamun et al. “Security Analysis of AES and Enhancing its Security by Modifying S-Box with an Additional Byte”. In: *International journal of Computer Networks Communications* (2017).

[59] *Matheus-Garbelini: esp32-esp8266-attacks, GitHub*. `https://github.com/Matheus-Garbelini/esp32_esp8266_attacks`. [Online] Accessed: 27-Nov-2021.

[60] Altaf Shaik et al. “Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems”. In: Jan. 2016.

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

[62] *“Where is HTTPS for IoT?,” DEV Community, 22-Oct-2018*. `https://dev.to/danielkun/where-is-https-for-iot-49ao`. [Online] Accessed: 11-Mar-2021.

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

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

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

[66] *oneM2M platform, gateway and device components*. `https://www.onem2m.org/using-onem2m/developers/device-developers`. [Online] Accessed: 20-April-2021.

[67] Hassija et.al. “A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures”. In: *IEEE Access* 7 (2019), pp. 82721–82743.

[68] “ETSI-Cyber Security for Consumer Internet of Things: Conformance Assessment of Baseline Requirements ”. In: (Accessed 19-Dec-2021). https://www.etsi.org/deliver/etsi_ts/103700_103799/103701/01.01.01_60/ts_103701v010101p.pdf.

[69] “TS-0003-V3.10.2 Security”. In: (Accessed 06-Aug-2022). https://www.onem2m.org/images/files/deliverables/Release3/TS-0003_Security_Solutions-v3_10_2.pdf.

[70] Imran et.al. “MISA: Minimalist Implementation of oneM2M Security Architecture for Constrained IoT Devices”. In: *2019 IEEE Global Communications Conference (GLOBECOM)*. 2019, pp. 1–6.

[71] Muhammad et.al. “oneM2M Architecture Based Secure MQTT Binding in Mbed OS”. In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*. 2019, pp. 48–56.

[72] “MQTT Binding”. In: (Accessed 13-August-2022). https://www.onem2m.org/images/files/deliverables/Release2A/TS-0010-MQTT_protocol_binding-v_2_7_1.pdf.

[73] Sicari et.al. “Secure OM2M Service Platform”. In: *2015 IEEE International Conference on Autonomic Computing*. 2015, pp. 313–318.

[74] *oneM2M Architecture*. https://onem2m.org/using-onem2m/developers/basics. [Online] Accessed: 21-July-2022.

[75] “OneM2M TR-0008”. In: ([Online] Accessed: 11-May-2022). https://member.onem2m.org/static_Pages/others/WPM-pages/TR-TS_List.htm.

[76] “TS-0004-V3.11.2 Service Layer Core Protocol”. In: (Accessed 11-10-2022). https://www.onem2m.org/images/files/deliverables/Release3/TS-0004_Service_Layer_Core_Protocol_V3_11_2.pdf.

[77] *Dashboard*. https://smartcitylivinglab.iiit.ac.in/building/. [Online] Accessed: 5-August-2022.

[78] *India Urban Data Exchange*. https://iudx.org.in. [Online] Accessed: 17-Aug-2021.

[79] *OM2M Setup*. `https://wiki.eclipse.org/OM2M/one/Starting`. [Online] Accessed: 10-Aug-2022.

[80] *mitmproxy: A free and open source interactive HTTPS proxy*. `https://mitmproxy.org/`. [Online] Accessed: 19-Aug-2022.