Security for oneM2M-Based Smart City Network: An OM2M Implementation

G.V. Ihita, Vybhav.K. Acharya, Likhith Kanigolla, S. Chaudhari, Thierry Monteil

Security for oneM2M-Based Smart City Network: An OM2M Implementation

G.V. Ihita[1], Vybhav.K. Acharya[1], Likhith Kanigolla[1], S. Chaudhari[1], Thierry Monteil[2]
[1]*International Institute of Information Technology - Hyderabad (IIIT-H), India*
[2]*IRIT, Université de Toulouse, INSA, Toulouse, France*
ihita.g@research.iiit.ac.in, sachin.chaudhari@iiit.ac.in, thierry.monteil@irit.fr

***Abstract*—Integrating scalability, interoperability, and security has become crucial with the widespread adoption of IoT-enabled smart city solutions. In this context, the oneM2M provides promising technical specifications for an interoperable and secure IoT/M2M system. This paper focuses on the potential threats and their impact on oneM2M standard-based smart cities. Further, configurations for baseline security of the oneM2M open-source implementation, called Eclipse OM2M, are presented. The configurations and recommendations are proposed based on the tests conducted on an existing smart city deployment of IIIT Hyderabad (IIIT-H) in India.**

***Index Terms*—Eclipse OM2M, Internet of Things (IoT) security, IoT security standardisation, oneM2M, smart city**

I. Introduction

There is widespread adoption of smart city solutions to make cities more livable, economically diverse, and environmentally sustainable. Smart city applications such as smart grids, waste management, traffic management, air pollution monitoring, and energy management leverage IoT devices to monitor, analyze and regulate various parameters for effective governance. IoT-enabled smart city requirements consist of heterogeneity, interoperability, scalability, mobility, connectivity, and security. To satisfy the need for a common platform supporting these requirements, oneM2M [1] proposes a common middleware technology in a horizontal layer covering use case-centric requirements, architecture, API specifications, security solutions, and interoperability for Machine-to-Machine and IoT technologies. These make the standard a de facto for a smart city as it reduces fragmentation, facilitates large amounts of data sharing, increases the re-usability of underlying existing technologies, and optimizes the costs. Various implementations of the standard are being developed and implemented globally. India has adopted the oneM2M standard as the national standard for IoT/M2M. There are multiple oneM2M implementations [2] such as Mobius, OASIS, CCSP, and eclipse OM2M. Each of these implementations has security provisions along with functional and operational requirements. The Eclipse OM2M [3] implements oneM2M and the smartM2M standard. It is an open-source project under the Eclipse Technology Project.

There are multiple trade-offs involved with the deployment of IoT-enabled smart cities. The trade-offs between cost, deployment scenario, processing power, and security of IoT devices usually result in security taking the backseat. Most IoT devices comprise low-cost sensors with memory, power consumption, and processing constraints. Thus, incorporating the principles of CIA (confidentiality, integrity, availability) becomes challenging.

Much work has been done on IoT/M2M systems and their security. Authors in [4] present a survey on the security requirements of mission-critical IoT applications, vulnerabilities, and sources of threats and culminate work on mitigation strategies for emerging applications. The work also presents the integration of blockchain with IoT for enhancing security. RFC 8576 [5] describes challenges with securing IoT deployments and categorizes the threats and risks to such systems. Proposing recommendations through consultations from stakeholders, the [6] focuses on challenges with IoT, threats, attack scenarios and possible mitigation strategies. Further ETSI standards on consumer IoT security [7] and [8] cover security and privacy best practices for consumers and manufacturers of IoT devices. Particularly in the case of IoT-enabled smart cities, [9] 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, [10] mentions security provisions and procedures on access control policies (ACP), dynamic authorization, application/device impersonation prevention, and privacy protection. The paper [11] 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 [12] show the implementation of secure MQTT binding as per [13] 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, [14] proposes a "Privacy_Enforcement" plugin.

This paper 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 modelled on oneM2M standard compliant implementations using STRIDE methodology [15]. Second, five security analyses are performed on this network: eavesdropping, brute force attack for OM2M credentials, authorization via access control policies, scalabil-

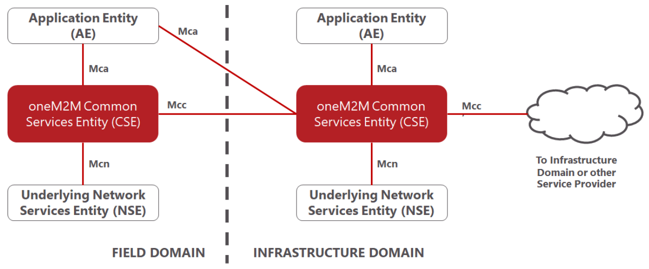

Application Entity (AE)
Mca
Mca
oneM2M Common Services Entity (CSE)
Mcc
Mcn
Underlying Network Services Entity (NSE)
FIELD DOMAIN
Application Entity (AE)
Mca
oneM2M Common Services Entity (CSE)
Mcc
Mcn
Underlying Network Services Entity (NSE)
INFRASTRUCTURE DOMAIN
To Infrastructure Domain or other Service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

Fig. 1. oneM2M architecture [16]

ity 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 our knowledge, this kind of security analysis has not been done for an actual OM2M based smart city deployment till date.

The paper is structured as follows: Section II describes the oneM2M standard, followed by Section III that focuses on modeling the major potential threats to any oneM2M standard-based system. Section IV describes the OM2M platform and its IIIT-H smart city implementation. Next, Section V presents a security analysis of OM2M followed by recommendations for baseline security in OM2M-based smart city proposed in Section VI.

II. 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.

A. 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. 1. Application entities deal with the application layer in the IoT setup, residing in the sensors and communicating with the M2M service layer using 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).

III. oneM2M FOR SMART CITY: THREATS AND VULNERABILITIES

Technical report [17] 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 I describing the potential security threats and vulnerabilities in an oneM2M based smart city-centric implementation. In addition to the threats, security provisions of [10], and technical specifications [18] 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:** Spoofing 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:** Tampering 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:** An application or system that fails to provide facilities to monitor and log user activities, allowing for malicious modification or falsifying the identification of new actions, is subject to a repudiation attack. 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,

TABLE I
STRIDE MODELLING OF THREATS TO ONEM2M

STRIDE	Potential threats to oneM2M standard based implementations	Security provisions in oneM2M	OM2M features for securing smart cities
Spoofing	• AE impersonation • Broken Authentication • Session Hijacking	• AE impersonation prevention • Authentication mechanisms: Symmetric key-based Security, Certificate-based security, Generic Bootstrapping Architecture (GBA) Framework	• IP and port blocking for replay attack mitigation • "m2m:nodeID" for node identification • Application entity identifier, Common service entity identifier
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" provide consistent updates to the end nodes to patch the security vulnerabilities • "m2m:DeviceID" uniquely identifies a device using a URN(Uniform Resource Name) • Device certificates to authenticate the AEs or CSEs	• Creation time(CT) and Last modified time(LT) present in the ContentInstance indicate illegitimate modified ContentInstance • X-M2M-OT as a HTTP header parameter indicating originating Timestamp of request and response
Repudiation	• Unavailability of access logs • Log injection-tampering-forging	• Token based authorization • Role based access control • "m2m:logStatus", "m2m:logTypeId" for checking the log status of the event management resource	• OM2M logging mechanism at server
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 • End-to-End Security of Primitives (ESPrim) • Secure environment plug-in	• Secure protocols binding as plugins have been implemented in the set up: HTTPs and secure MQTT. • Locking of account after failed attempts • Each AE or container in the resource tree has a unique id • Hashing of ContainerID • Each application vertical of the resource tree has separate credentials.
Denial of Service	• Buffer overflow • Flooding of RestAPI requests	• "m2m:accessControlRule" (with assigning access based using m2m:ipv4, m2m:ipv6 and m2m:locationRegion) • mutual authentication through secure association establishment procedure	• 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. • Transition from H2 to Mongo to address buffer overflow challenges
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	• "m2m:authorizationStatus" provides status of access control policies • "m2m:accessControlRule" to define privileges of entities

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:** A Denial-of-Service (DoS) attack 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.

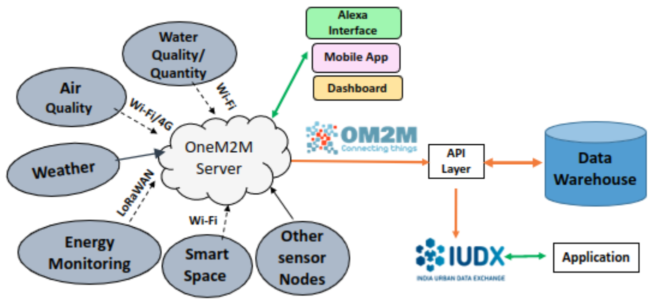


Fig. 2. OM2M based smart city deployment

- **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.

IV. OM2M IMPLEMENTATION AT IIIT-H

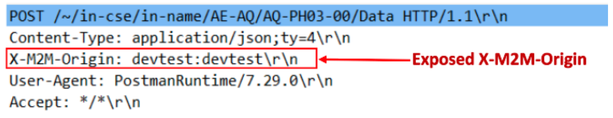
This section studies the OM2M platform and its integration with the smart city setup. Various experiments are conducted, and observations are made to understand the platform's security provisions and effectiveness in catering to the security requirements. The experiments are conducted on an actual OM2M-based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

A. About OM2M

OM2M [19] stands for open-source M2M (Machine To Machine) 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.

B. Smart city at IIIT-H

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



```
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
```

Fig. 3. Confidential X-M2M-Origin visible in HTTP packet on Wireshark

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 [20], smartphone applications, Alexa interface, and home automation access the data from the warehouse. The general user must utilize the Indian Urban Data Exchange (IUDX) [21] 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. Fig. 2 shows the current smart city deployment based on OM2M.

V. SECURITY ANALYSIS OF OM2M

A. Eavesdropping attack

Eavesdropping was performed as an initial analysis to gain visibility into the communication between the AE and CSE. Fig. 3 shows the result of performing a passive eavesdropping on the network using Wireshark. 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. X-M2M-Origins assigned by the Originator of the request which maybe a CSE or AE.

The attacker can easily create custom requests with this header value, which can severely impact the entire infrastructure. Therefore, in addition to enabling HTTPS to mitigate this scenario, separate XM2M credentials were created for all the verticals present such as air, water, and energy, thereby restricting the scope of the attack.

B. Brute force attack for OM2M credentials

In order to access the entries of the resource tree, users must login 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

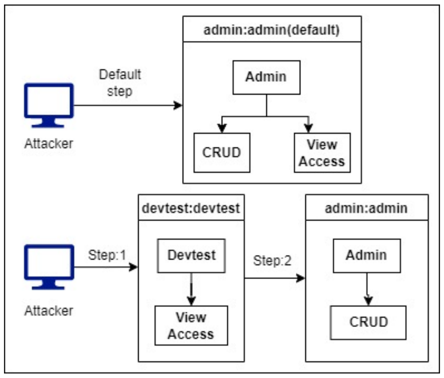


Fig. 4. Mitigation against brute-force attack

IP address. Further, there is a limit on the type of special characters allowed for the passwords making our brute force attack easier. Passwords with characters such as '(', ')', '[', ']' were not allowed.

C. Authorization via access control policies

To access the setup, the configuration file in the default OM2M setup 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 application entities (AE), internetworking proxy entities (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. Therefore, the attacker can never obtain the admin credentials by guessing them on the server login page. As shown in Fig. 4, the attacker must perform two steps to obtain all the previous privileges. This setup is a form of mitigation against the brute force on the server login page.

- 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.

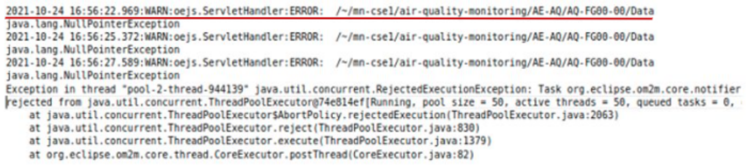

```
2021-10-24 16:56:22.969:WARN:oejs.ServletHandler:ERROR:  /~/mn-cse1/air-quality-monitoring/AE-AQ/AQ-FG00-00/Data
java.lang.NullPointerException
2021-10-24 16:56:25.372:WARN:oejs.ServletHandler:ERROR:  /~/mn-cse1/air-quality-monitoring/AE-AQ/AQ-FG00-00/Data
java.lang.NullPointerException
2021-10-24 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
rejected from java.util.concurrent.ThreadPoolExecutor@74e814ef[Running, pool size = 50, active threads = 50, queued tasks = 0,
    at java.util.concurrent.ThreadPoolExecutor$AbortPolicy.rejectedExecution(ThreadPoolExecutor.java:2063)
    at java.util.concurrent.ThreadPoolExecutor.reject(ThreadPoolExecutor.java:830)
    at java.util.concurrent.ThreadPoolExecutor.execute(ThreadPoolExecutor.java:1379)
    at org.eclipse.om2m.core.thread.CoreExecutor.postThread(CoreExecutor.java:82)
```

Fig. 5. H2 database server crash - a scalability issue

D. 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. We 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, we migrated our database from the high-speed in-memory H2 database to MongoDB, an auto-scalable production development NOSQL database. This database change helped solve the server crash issues, thus preventing a denial of service attack.

Fig. 5 shows 944222 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.

E. 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.

In our demo experiment, we use MITM Proxy, a Kali Linux-based cybersecurity tool for penetration testing and replaying web traffic. 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. The experiment proceeds as follows:

- Post requests are sent to the OM2M server that gets intercepted by the MITM proxy
- MITM proxy keeps track of the request from the client side, allowing replay of the packet at a different time interval.

We observe a new content instance created in the 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.

VI. OM2M security recommendations

This section presents solutions implemented for each STRIDE element based on the experiments conducted. The baseline security configurations for OM2M are shown in Table I. In the case of spoofing or impersonation, oneM2M technical report TR0008 mentions ways to mitigate this threat using a secure communication link or Role Based Access Control (RBAC). In OM2M, we use HTTPS binding to encrypt the headers, node identifiers, and timestamps. Technical specifications on service layer core protocol [18] specify a list of common data formats, interfaces, and message sequences to be used by developers. M2M node identifier (m2m:NodeID) data format has been implemented to mitigate impersonation and replay attacks. This does not require modifying the entire authentication mechanism of the deployment. Data tampering is detected by observing the discrepancy in the resource tree's creation and last modified time. The access control policies authorize entities for data retrieval and manipulation. Further, using "m2m:authorizationStatus", the status of a resource authorization is known. HTTP header X-M2M-OT provides the originating timestamp parameter of request and response, thus indicating possible packet injection and similar man-in-the-middle attacks. Similarly, "m2m:authorizationStatus" along with "m2m:operationMonitor" and "m2m:accessControlRule" help reduce the elevation of privilege. In the case of repudiation, provisions for tracking and logging users' actions have been implemented by OM2M server. The "m2m:logStatus" provides additional update on the logging activity. To mitigate the threat of information disclosure, secure HTTP provisions are implemented. SAEF can be established to mutually authenticate the MQTT Client and MQTT Server when using the MQTT protocol. In an attempt to mitigate the DoS attack, modifications were made both at the node end and the server end. Additionally, the nodes are MAC, and IP bound, thus a white list of permitted IP and MAC addresses is implemented to implement IP blocking during DoS.

VII. Conclusions and Future work

This work analyses the security provisions by oneM2M and its implications for smart cities. Numerous potential threats and vulnerabilities to oneM2M implementations are explored and modeled using the STRIDE threat modeling framework. Experiments are conducted to understand the impact of these threats on an actual OM2M-based smart city deployment based on the existing configurations of the OM2M platform. The appropriate features and configurations for baseline security of smart cities are then proposed based on the STRIDE framework. In the future, we aim to implement a SAEF and conduct a comparative analysis to understand the tradeoffs in a constrained environment.

Acknowledgment

This research was supported partly by National Geospatial Programme (NGP), India, under grant no. 2073 (2020), PRIF Social Incubator Program (2019) and the Ministry of Electronics and Information Technology (MEITY) under grant no. 3070665 (2020), with no conflict of interests. The author, G.V. Ihita thanks IHub-Data, IIIT Hyderabad for a research fellowship.

References

[1] *OneM2M*, [Online] Accessed: 15-Mar-2021, https://www.onem2m.org/.

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

[3] "Eclipse OM2M," Accessed 25-August-2022, https://www.eclipse.org/om2m/.

[4] Hassija et.al, "A survey on iot security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019.

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

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

[7] "ETSI- Cyber Security for Consumer Internet of Things:Baseline Requirements ," [Online] Accessed: 15-Sept-2021, https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf.

[8] "ETSI-Cyber Security for Consumer Internet of Things: Conformance Assessment of Baseline Requirements ," Accessed 11-Mar-2021, https://www.etsi.org/deliver/etsi_ts/103700_103799/103701/01.01.01_60/ts_103701v010101p.pdf.

[9] G.V. Ihita et.al, "Security analysis of large scale IoT network for pollution monitoring in urban india," in *IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021, pp. 283–288.

[10] "TS-0003-V3.10.2 Security," Accessed 11-Mar-2021, https://www.onem2m.org/images/files/deliverables/Release3/TS-0003_Security_Solutions-v3_10_2.pdf.

[11] 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.

[12] 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.

[13] "MQTTbinding," Accessed 13-August-2022, https://www.onem2m.org/images/files/deliverables/Release2A/TS-0010-MQTT_protocol_binding-v_2_7_1.pdf.

[14] Sicari et.al, "Secure om2m service platform," in *2015 IEEE International Conference on Autonomic Computing*, 2015, pp. 313–318.

[15] Microsoft Security. 2007. STRIDE chart Microsoft Security, "STRIDE threat modelling framework," [Online] Accessed: 14-Feb-2021, https://www.microsoft.com/security/blog/2007/09/11/stride-chart/.

[16] "Architecture," Accessed 21-July-2022, https://onem2m.org/using-onem2m/developers/basics.

[17] "OneM2M TR-0008," [Online] Accessed: 11-May-2022, https://member.onem2m.org/static_Pages/others/WPM-pages/TR-TS_List.htm.

[18] "TS-0004-V3.11.2 Service Layer Core Protocol," Accessed 11-10-2022, https://www.onem2m.org/images/files/deliverables/Release3/TS-0004_Service_Layer_Core_Protocol_V3_11_2.pdf.

[19] M. Ben Alaya et.al, "Om2m: Extensible etsi-compliant m2m service platform with self-configuration capability," *Procedia Computer Science*, vol. 32, pp. 1079–1086, 2014, The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014).

[20] "Dashboard," Accessed 25-August-2022, https://smartcitylivinglab.iiit.ac.in/building/.

[21] "India Urban Data Exchange," Accessed 11-Dec-2021, https://iudx.org.in/.