View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by International Institute for Science, Technology and Education (IISTE): E-Journals

Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.11, No.2, 2020
www.iiste.org

A Cloud Based Approach for Data Security in IoT

Olumide Kayode
Department of Computer Science, University of Texas at San Antonio
One UTSA Circle, San Antonio, TX 78249, United States

Abstract
The number of connected devices grows rapidly each year as more and more enterprises realize its potentials. Despite the benefits, data privacy and security in Internet of Things (IoT) remain a major issue. IoT devices generates massive data but are limited in resources. The proliferation of IoT devices and the increasing network traffic have heightened the attack surface. Moreover, the susceptibility of transmitted data to eavesdropping and man-in-the-middle attacks is a great concern to users and manufacturers. It is not all IoT devices that utilize encryption. IoT traffic from devices that transmit data in plain text will expose sensitive information if intercepted by an adversary. Such a vulnerability can be exploited to facilitate identity theft and implement other devasting cyber-attacks. This paper presents an implementation of a cloud-based security approach for transmitted data in IoT. The design is based on Lambda architecture using Amazon Web Services. The proposed approach effectively processes and analyzes real-time sensor data as well as historical data from a master database on the cloud. It also ensures the security of user's information.
Keywords: Internet of Things, Cloud computing, Data security, Lambda architecture
DOI: 10.7176/CEIS/11-2-03
Publication date: February 29th 2020

1. Introduction

The Internet of Things (IoT) represents a new class of connected devices. IoT describes a group of interconnected, common devices, utilizing intra device and cloud services for communication and storage. Manufacturers who wished to leverage the "always on", "always connected" behavior of the Internet, added network functionality to their products and advertised this network connectivity as the next wave of innovation. As an example, IoT refrigerators contain internal cameras. These cameras are used as part of a feature, to track the internal contents of the refrigerator and share this view via a smart phone connection. The feature enables the user to view the current inventory, build a grocery list, and set alarms when a tracked item's quantity is below a chosen threshold. Another IoT device use at home is the smart plug. It can be used to create on and off schedules, set timer and to automate plugged-in electronics. The Wi-Fi enabled wall plug can be control from anywhere using the corresponding mobile app. The convenience and benefits derived from the adoption of IoT cannot be overemphasized.

Prominent features and technologies associated with IoT like communication, connectivity, intelligence, fog and edge computing are related to cloud computing. Actually, cloud computing provides the necessary tools and services to create IoT applications. It also provides on-demand network access to a set of configurable computational resources such as services, servers, networks and storage which can be dynamically scaled to meet specific requirement. Reliability, scalability and convenience are some of the reasons why IoT developers and researchers use cloud computing. Since IoT and its adoption in cloud computing requires the use of the Internet, security and privacy of transmitted data are of great importance. According to a study (Modi *et al*, 2013) security is an essential challenge in cloud computing environment. Data privacy and security of IoT remains a major issue and will be the main focus in the foreseeable future. The vast number of IoT devices coupled with the enormous data generated necessitate ensuring the security of sensor data and protection of user's information to enhance privacy and trust.

This research work presents an approach based on Amazon Web Services (AWS) for secure data transmission in IoT system. This study first investigates the traffic generated by a representative IoT device used at homes. The data being transmitted by smart plug are examined to determine if sensitive information is being exposed without any protection. This paper further presents a functional, scalable and versatile architecture using AWS tools and services. The goal is to create a lambda architecture that can be used to process real-time IoT data while ensuring the security of user's information.

2. Security in IoT

Many data privacy advocates cringe on the notion of connecting everything around us or at home to the Internet. IoT provides valuable features and benefits, but clearly at the cost of privacy and to some degree, a sufficiently large attack surface. IoT devices are essentially small computers or electronic devices. They execute embedded software, which is typically open source solutions coalesced into a single Software Development Kit (SDK). When a security bug is found in one of these solutions, a patch needs to be created and applied. Often the open

source project creates a fix for the exploit, but the IoT manufacturer sometime fails to package and update this new version. One common observed issue is with the embedded web server in residential network routers. A commonly observed configuration is the execution of an embedded form of Linux, containing an older version of the Tomcat web server. Utilizing the Cross-Site Scripting (XSS) test, a few XSS vulnerabilities which were attributed to either the server or the manufacturer's code could be discovered.

Several works have studied the security issues in IoT and discussed possible solutions. One of such works (Neshenko *et al,* 2019), provides a detailed study of IoT vulnerabilities, attack vectors and exploitations. It discussed the severity of IoT problems, current challenges and identified some possible future initiatives. The authors in (Yi *et al*, 2019) proposed an intelligent early warning vulnerability detection algorithm. They utilized an attack graph to model vulnerability and discussed ways to assess network security in IoT environment. An experimental investigation of IoT traffic generated by six representative IoT devices used at home was presented in (Olumide & Tosun 2019). It identified vulnerabilities in four IoT devices and proposed IoT traffic monitoring strategies. (Blythe and Johnson, 2018) propose an IoT consumer index score, to assist consumers in making informed choices. They discussed the lack of fundamental defense and suggested a clarifying security score, as a consumer protection. Much like a processed food label, their index score would serve to clarify the precise security posture of a given IoT device. The authors further suggested that consumers deserve to understand a devices' underlying behavior and choose products which best serve their interests. The authors in (Brass *et al*, 2018) chose to focus on the evolution of IoT security standards to understand the current security trends and potentially the future direction of binding standards. Their conclusion contains a proposed set of standards and they urge federal governing bodies to enforce strict adoption. Safety necessities like secrecy, unity and substantiation for IoT was presented in another research work (Gurunath *et al* 2018). IoT threat analysis involving twelve kind of different attacks, use of botnet, security issues in IoT networks and power consumption issues were also discussed.

Ways to mitigate IoT security challenges pertaining to device cloning and sensitive data exposure was addressed in (Naik & Maral 2017). Security issues like data tampering, denial of service, unauthorized device access and control were also discussed. Oliot-Discovery Service (Kwon *et al*, 2016) deals with performance as well as security issues by focusing on intra discovery service aspect. The authors discussed that data storage is a critical problem to solve at scale and they utilized the proposed service to securely identify data storage. Researchers have also highlighted the urgent need to constrain system component to a reasonably secure and private behavior (Han *et al*, 2015). The authors recommended following security best practices in home automation. Moreover, a way of providing privacy in a multi-trust-domain environment was presented in (Banerjee *et al*, 2014). The authors introduced a methodology for privacy-aware communication by utilizing a novel zero-exposure slot allocation scheme and anonymous communication between devices. (Caiming *et al*, 2013) introduces the novel idea of mitigating IoT exploits by using an analogue to the human immune system. The author's rightfully focus on the increasing way IoT devices are exploited and consider treating the IoT environment as a pathogen ridden system. Their idea uses sound defense in depth practices and combines security mechanisms into an adaptive approach. It focuses on shifting mitigating features as attacks change over time. Another work (De Rubertis *et al*, 2013) focuses on two end-to-end encryption protocols and proposes a comparison between the two, as well as a means to assist network designers in choosing the appropriate protocol, for a given set of hardware. This paper identifies a singular point within the IOT security landscape and proposes the comparison of two well-known solutions.

Moreover, most of the IoT devices lack a firewall and less expensive devices fail to sanitize communication. Security features like anti-virus and anti-malware software are also not supported by most IoT devices. Arguably, most inexpensive devices contain minimal engineering effort. Clearly, just enough work has been done to support a device's feature set, with no observed effort in the security space. More efforts need to be made to ensure that transmitted data which are generated by IoT devices are secure. Platforms, middleware and communication channels used by IoT devices should also be equipped with up to date security mechanisms. Users often serves as the weakest link in cyber security. Precautionary measures and security best practices should be applied possibly at all times by users during installation, configuration and interaction with the IoT devices.

3. Experiment: Smart plug vulnerability

The data being transmitted by a representative IoT device during user interaction with a smart phone can be intercepted using an appropriate tool to verify if user's information is being exposed without any privacy protection. In this work, I used Fiddler which is a free tool that can be used to capture Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) traffic. In this section, I explore the inherent vulnerability that may be found in a smart plug, which is one of the common IoT devices used at home. The software tool used to intercept traffic, approach and observed security flaw are discuss in this section.

3.1. Fiddler

It is a free tool that can be used for capturing HTTP and HTTPS traffic. It works as a proxy server application and can be used to decrypt HTTPS traffic via man-in-the-middle interception using self-signed certificates. It uses HTTP connect tunneling. Prominently used on Windows, it registers itself as the system proxy for Microsoft Windows Internet Services (WinInet), the HTTP layer used in windows. As the system proxy, all HTTP requests from WinInet flow through Fiddler before reaching the target web servers and all HTTP response flow through it before being returned to the client application. It unregisters itself as the system proxy once user closes the application before shutting down. Other major resources include FiddlerCap and FiddlerCore. The latter can be used for traffic viewing and modification for .NET applications. It is a .NET class library that can be integrated into .NET framework application. It allows users to capture and modify traffic without any of the Fiddler user interface features.

FiddlerScript is one of Fiddler's major components that can be used to write customized rules for filtering and capturing request/response of any web session. It is based on Jscript.NET but there is an option for C# which is what I used in this work. Predefined Methods in Fiddlerscript that can be used includes OnBoot(), OnShutdown(), OnAttach(), OnDetach() and Main(). For each web session, the session event methods are OnPeekAtRequestHeaders(), OnBeforeRequest(), OnPeekAtResponseHeaders() and OnBeforeResponse(). Most of our work includes creating additional methods and calling them in the predefined methods specified in FiddlerScript. I also edit some of the existing codes in the predefined methods to suit our needs.

3.2. Experiment and Vulnerability

I consider several factors like Wi-Fi enabled, cost, easy to setup and use in the choice of the IoT device used in our work. The *Insignia Wi-Fi Smart Plug* is a smart device that can be used to create on and off schedules, set timer and to automate plugged-in electronics at home. This Wi-Fi enabled wall plug can be control from anywhere using the Insignia Connect App and it requires no additional hardware during setup. As shown in figure 1, a man-in-the-middle attack implemented via a proxy server intercepts traffic generated

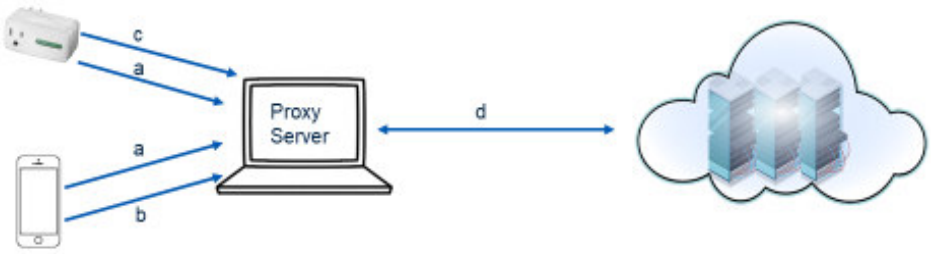

Fig. 1: Man-in-the-middle attack via proxy server

by a smart plug. I run the Fiddler on the workstation and configure the smart phone to use proxy, setting the proxy hostname to the workstation IP address and the proxy port to 8888 used by Fiddler. The communication session between the smart phone and smart plug denoted by *a* flow through the proxy server for every request and response. All requests from the smart phone to the server denoted by *b* and those from the smart plug denoted by *c* flow through the proxy server before reaching the target destination on the cloud. The proxy server forwards the intercepted request and also receives the response from the server before directing it to the client. This communication path denoted by *d* as shown in figure 1 makes the server thinks that it is communicating directly with the client. Using this approach, all transmitted data can be accessed and modified by an adversary. As shown in figure 2, the smart bulb is constantly sending the IP address and

{"local_reg":{"ip":"192.168.1.48","notify":0,"port":10275,"uri":"/local_lan"}

Fig. 2: Vulnerability in smart plug data transmission

port number repeatedly. First, this is not an efficient way to utilize bandwidth. Secondly, the IoT device leaks sensitive information in plain text. These are vulnerabilities that can be exploited by hackers to compromise the security of the smart plug and violate data privacy. The constant rate of data transmission can be accelerated to simulate denial of service attacks. Also, the leaked IP address exposes the identity of the IoT device. I suggest that transmitted data should be encrypted, and frequent transmission of device's identity should be discouraged.

4. Lambda Architecture

Towards a cloud-based approach for data security in IoT, I implemented a lambda architecture on Amazon Web Services (AWS). Concerning proxy and man-in-the-middle attacks, the first line of defense is authentication. A robust access control and authentication procedure will prevent unauthorized access to user's information or device ID. Lambda architecture is a big data solution that allows for massive queries as well as real-time data views. It is suitable for systems that need close to real-time data while performing timely queries against large databases. IoT aptly fits nicely into this description, as most IoT devices transmit data at regular interval and provide some kind of real time information. In addition to the real-time data, in lambda architecture it is possible to process old data sets to find patterns or provide more context to the end user. Any abnormality in data transmission trend can easily be detected using this technique.

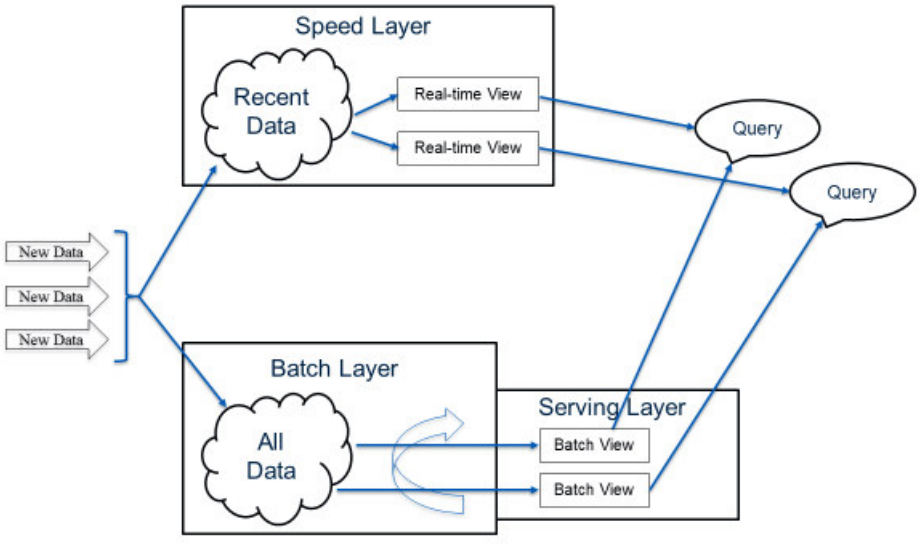

Fig. 3: Lambda architecture design

As shown in figure 3, lambda architecture comprises of three unique layers. The first layer is the batch layer. It stores a copy of the master database and precomputes batch views from this master database. It stores immutable values and the dataset are constantly growing. Any arbitrary function can be computed on the dataset to construct a batch view. Here, the large dataset is queried, and data views are pre-computed for the end user based on the specification. The second is the speed layer. This layer is a specialized, distributed database that takes in a batch view and allows random reads to be done. When a new batch view is available, it simply swaps out the old for the new one. In comparison to batch layer, it only considers recent data while batch layer looks at

all the data at once. Additionally, here in speed layer, data is taken in real time, processed and displayed in real-time view. Lastly, there is the serving layer which combines the data views of the batch and speed layers. The serving layer indexes the batch views so that they can be queried in low-latency, ad-hoc way. Lambda architecture has been widely accepted as powerful architecture for systems that need real-time views such as many IoT systems. Moreover, there are a number of ways to process and store data in the cloud. Virtually all IoT devices transmit data to their respective cloud service or storage. At any point in time, IoT devices will require firmware or software update. All these interactions with the cloud must not predispose IoT device to cyber-attacks. The security features provided by most cloud service provider can be utilized for robust authentication and ensure secure data processing. The distributed nature of IoT necessitate the need for strong security mechanism at the cloud layer. Data from various sources can be verified to be accurate and non-malicious before any processing or storage. The cloud is one ideal location where heterogeneous IoT devices and their diverse generated data can be validated and protected.

5. Proposed Approach using AWS

Figure 4 illustrates the design of the proposed architecture using multiple AWS services. In this scenario, I assume an end user control the initiation and termination of the IoT device. This takes the form of a mobile application with simple "Start" and "End" commands. Basic level of security mechanism like authentication in the form of utilization of username and password are also implemented.

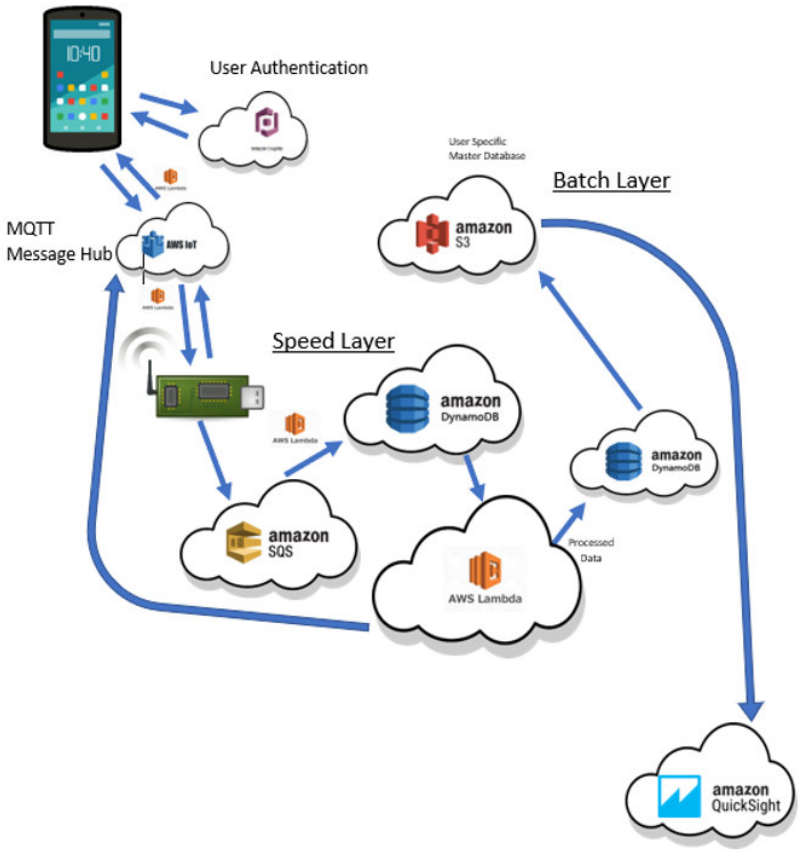

Fig. 4: Proposed architecture

The AWS has user authentication in the form of AWS Cognito. The two main components of AWS Cognito are user pools and identity pools. A user pool is a directory of users who have login access to your mobile application or web application. The identity pool limits what access each user has to other AWS services. As shown in the proposed architecture, after authentication occurs on the mobile app, users will have the ability to start and stop the IoT device. Clicking the start button sends a single MQTT message to AWS IoT core. The IoT core has a lot of capabilities and functions, but for the purpose of this architecture it is essentially a MQTT message broker. Devices have the ability to publish and subscribe to MQTT nodes managed by AWS IoT core. Hence, when the "Start" button is hit, a MQTT message is sent to the designated topic. The AWS IoT core receives it and sends the MQTT message to all subscribing devices.

Once the IoT device receives the specific MQTT message, the device will begin generating data. This data is sent to AWS Simple Queue Service (SQS). AWS SQS can receive and transmit data at any volume without any loss in data or message. By implementing AWS SQS, the proposed architecture can scale with any number

of devices without fear of dropping packets and lost of essential data. Moreover, AWS SQS is easy to set up, implement, maintain and the cost are based on usage. This service ensures the reliability and scalability of the overall design of the system.

Additionally, after the data is queued by AWS SQS, it is sent to the AWS DynamoDB. The AWS DynamoDB is a NoSQL database that is completely server-less to the end user and scales as needed. It is capable of millions read/write requests per second. Also, it can handle real-time data processing by enabling DynamoDB streams. Arguably, this is a great tool to utilize in a speed layer because of its ow latency, and ability to scale as needed. In the proposed architecture, all data from SQS is sent to a master DynamoDB database. This master database has DynamoDB streams enabled and thus anytime new data is written, it triggers analysis from AWS Lambda.

The next phase involves the use of AWS Lambda. The AWS Lambda enables the ability to run code in between back-end services. It is easily configured and can trigger specific functions on event within the service. It is utilized within the proposed architecture but mostly as a facilitator in the movement of data between AWS services. For example, AWS Lambda is invoked to move data from AWS SQS to AWS DynamoDB. However, AWS Lambda allows for any code to be implemented and triggered from an event. It allows for more complex analysis to take place within the structure of the overall system. In the proposed architecture, analysis is confined to one AWS Lambda function so tat it can easily be swapped out for different functions. This approach gives the end user the ability to customize their analysis to their specific need. Additionally, other tools can be implemented here such as machine learning algorithms or map reduce tools. End users can perform complex analysis on real-time data streams without disrupting the rest of the structural design of the architecture. After analysis is complete, data is sent to the device as a MQTT message as well as written to a user specific AWS DynamoDB database and stored until the user hit “End” which eventually stops the data collection.

The next service is the AWS Simple Storage Service (S3). It is an object storage service that is scalable and reliable. S3 objects can also be queried for analysis or visualizations. Within the proposed architecture, each user has a master S3 bucket with all archived data. When the user hits “End” on the mobile app, new data will be written to the user’s S3 bucket as a JSON object file. The batch layer then accesses this master S3 bucket when pulling old records. Subsequently, data can be further analyzed or visualized in AWS QuickSight. The AWS QuickSight is an analysis visualization tool which can be configured to read in files stored in AWS S3 and builds interestingly displays or views of large dataset. If required, additional analysis can be performed within QuickSight and results display to the end user.

Implementation of the proposed architecture requires development on three fronts. First, an android application has to be developed to interact with AWS Cognito and AWS IoT core. Secondly, a Raspberry Pi and a LSM9DS1 Inertial Measurement Unit (IMU) will be connected to transmit data to AWS SQS. Lastly, AWS Lambda functions will be written to connect AWS services and perform the analysis of incoming data. The android application accomplishes two things. First, it must connect with AWS Cognito to verify users or register new users. When new users are added, it must also add them to an identity pool to access other AWS services. In addition to the AWS Cognito connection, the app must also connect to AWS IoT. After a successful login, user will be directed to the main screen as shown in figure 5. The “Start” and “End” button sends MQTT requests to AWS IoT core. After hitting the “Start” button, the app that has subscribed to the target topic will receive incoming data. When it receives a message, it plots a graphical representation of it. The graph shows the 30 most recent points and does not save or do any analysis on the data.

Concerning AWS Lambda functions, four instances of AWS Lambda are used in the implementation of the proposed architecture. The first function will be called after the “Start” MQTT message arrives in AWS Iot core. In this function, a message will be sent to the Raspberry Pi as well as the username of the end user and the session number. Every time the “End” button is hit, the session number is updated. This way, data set can be separated easily. The second AWS Lambda function is initiated when data arrives at AWS SQS. This function moves the data into the master database. The third AWS Lambda function is the analysis function. It is triggered every time data is written into the master database and it queries the master database for data points from the specific user. It takes a running average of the data points and then sends the data to the appropriate places. First, it sends it back to the mobile app through MQTT messages. Secondly, it stores the data in a user-specific database. The last AWS Lambda function is triggered by the “End” MQTT message. This function first stops the Raspberry Pi from transmitting data into AWS SQS. After the data stops streaming, this function writes data from the user-specific database into AWS S3 for storage. After data is written to S3, it is now available to AWS QuickSight for visualization.

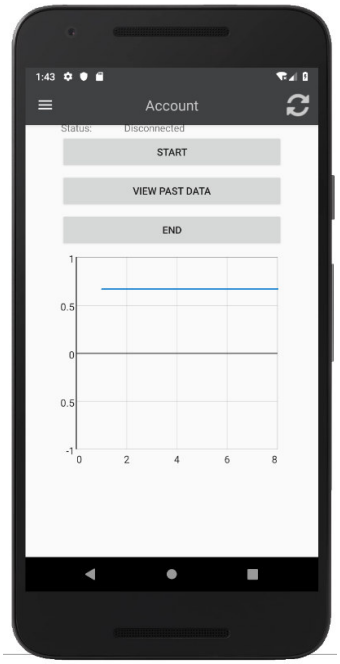

Fig. 5: Android application main page

Visualization in the speed layer occurs on the mobile device and data is transmitted through MQTT messages. The speed layer visualization is close to real-time with minimal latency. The data on the mobile application is only used for visualization and no further analysis takes place on the device. No data is stored except for the 30 most recent data points. The batch layer visualization occurs on the AWS QuickSight. Overall, the proposed architecture is scalable and flexible in its implementation. It successfully processes, analyze, and visualize real-time data in the designed speed layer while simultaneously fetching data from the batch layer and the master database. Using the provided AWS services, the security of transmitted messages can be enhanced and unauthorized access to user's information can be prevented.

6. Conclusion

Data privacy and security in IoT is a major concern for both users and manufacturers that need urgent attention as well as robust solutions. One prominent cyber-attack is proxy and man-in the-middle attacks. The susceptibility of IoT devices to exploitation of inherent vulnerabilities require more exploratory research. In this work, I investigated the traffic generated by a smart plug and discovered a major vulnerability. The IoT device transmit data in plain text without encryption and a robust security mechanism. As a solution, I proposed a cloud-based approach for secure data transmission, processing and storage. Using AWS services, I implemented a lambda architecture that handles authentication and access control. User's interaction with IoT device via smart phone was secured and MQTT messages were analyzed to meet specific user's need. Moreover, I developed a mobile application to demonstrate real-time visualization of transmitted sensor data. As a future work, I intend to include more sensors and Raspberry Pi. The hardware features can be upgraded, and multithreading can be enabled to handle multiple tasks simultaneously.

References

C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud", *Journal of Network and Computer Applications,* Vol. 36, No. 1, pp. 42-57, Jan. 2013.

N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and N. Ghani, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations", in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, thirdquarter 2019, pp. 2702-2733.

M. Yi, X. Xu, and L. Xu, "An intelligent communication warning vulnerability detection algorithm based on IoT technology," IEEE Access, vol. 7, pp. 164 803–164 814, 2019.

O. Kayode and A. S. Tosun, "Analysis of IoT Traffic using HTTP Proxy," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1-7.

J. M. Blythe and S. D. Johnson, "The consumer security index for IoT: A protocol for developing an index to improve consumer decision making and to incentivize greater security provision in IoT devices," *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, London, 2018, pp. 1-7.

I. Brass, L. Tanczer, M. Carr, M. Elsden and J. Blackstock, "Standardising a moving target: The development

and evolution of IoT security standards," *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, London, 2018, pp. 1-9.

R. Gurunath, M. Agarwal, A. Nandi and D. Samanta, "An Overview: Security Issue in IoT Network," *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on*, Palladam, India, 2018, pp. 104-107.

S. Naik and V. Maral, "Cyber security — IoT," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, 2017, pp. 764-767.

K. Kwon, D. Kim and D. Kim, "Oliot-Discovery Service: Dealing with Performance and Security Issues from Intra-DS Aspect for IoT," *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, 2016, pp. 1-6.

J.-H. Han, Y. Jeon, J. Kim, "Security considerations for secure and trustworthy smart home system in the IoT environment", *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1116-1118, 2015.

D. Banerjee, B. Dong, M. Taghizadeh and S. Biswas, "Privacy-Preserving Channel Access for Internet of Things," in *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 430-445, Oct. 2014.

C. Liu, Y. Zhang and H Zhang, "A Novel Approach to IoT Security based on Immunology", *Ninth International Conference on Computational Intelligence and Security*, 2013.

A. De Rubertis, L. Mainetti, V. Mighali, L. Patrono, I. Sergi, M. Stefanizzi, and S. Pascali, "Performance evaluation of end-to-end security protocols in an Internet of Things", *Software, Telecommunications and Computer Networks (Soft-COM)*, 21st International Conference, 2013, pp. 1-6.