8-2021

Internet of Things Security Case Studies and Internet of Things Core Service Comparions

Jaseong Koo

INTERNET OF THINGS SECURITY CASE STUDIES AND INTERNET OF THINGS CORE SERVICE COMPARISONS

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Information Systems and Technology:

Cyber Security

---

by

Jaseong Koo

August 2021

INTERNET OF THINGS SECURITY CASE STUDIES AND INTERNET OF THINGS CORE SERVICE COMPARISONS

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Jaseong Koo

August 2021

Approved by:

Benjamin Joseph Becerra, PhD, Committee Member, Chair

Conrad Shayo, PhD, Committee Member, Reader

Jay Varzandeh, PhD, Chair, Information & Decision Sciences Department

ABSTRACT

This culminating project conducted an analysis of IoT security breach case studies. The analysis identified numerous vulnerable points: software failure, node tampering attack, eavesdropping, code injection, unauthorized access, social engineering attack, hardware exploitation, and node insertion. It therefor seems that even with the proper tests conducted on vulnerabilities to discover solutions, regular end users are unable to apply patches or other technical solutions to protect themselves. This project solely focuses on analyzing of comprehensive IoT security services that come with devices connected to home network. The devices are those provided by the big three: Amazon, Google, and Microsoft, on the communication between platform and devices, how they are protected, and how costs vary depending on different situations. Also, performance differences were analyzed among different solutions based on three different scenarios with different number of settings to give a deeper insight to users. There are comparisons throughout the paper, but it is to help normal users make better choices depending on their different situations and purpose of usage.

Dedication

First, I have to give the deepest gratitude to my beloved family. This project is dedicated to my father, mother, and brother. My father who i did not get along with for many years but as I grew and started to understand how much virtue you have taught me with bittersweet disciplines. I could have become who I am now and survived here in foreign country this far because you have been there for me my entire life. Dad, you are my one and only role model for my life and thank you for the most supports that I could imagine.

To mom, who carried me around for 10 months in one of the toughest times of her life, I have to say thanks to you from my heart. She is the only person that showed me unconditional love of mother and always supported me. Without your support, I could have not made it this far. I love you with all my heart. Thank you for always believing in me and giving me a discipline to become a better person. With your nurture, I am the who I am now.

To my brother, I cannot believe he got so big and strong. You are one of my best friends who I can rely on and talk about anything. I wish you my best luck that you can stay bold and always stay healthy in New York. You are the best brother in the world.

Now, I stand at another starting line which will have a huge impact on my life. I will never forget the love you gave and how hard it was to provide me with spiritual, moral, emotional, and financial support. Lastly, I would like to thank my family once more for guidance, strength, love, and so many other things.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE:

INTRODUCTION

With the rapid advancements on human technology, it is almost impossible to separate human beings from information technologies. It is already prevalent in the fields of industries where IoT devices replace human work forces and pairs up with the cloud computing for its management and control. Not only the growth in the industrial fields, but also dramatic increase on the personal usage of IoT technology for the easier and more comfortable lifestyle it brings. However, IoT devices are normally equipped with the limited computational power and other limited functional capacities. Unlike the industries, where technical experts are ready to supplement the integration of IoT and security, normal users who utilize IoT at home network usually do not have enough knowledge to implement technical controls or understand the vulnerabilities embedded in the system. There are numerous real-world security threats awaiting.

With the rapid advancement on the field of information technology, there are various changes that require on demand adaptations by end-users (M. Chapple et al. 2021). Without the proper understanding of security threats, end users may become victims of cyber-attacks. Especially, users of IoT devices connected to regular home networks are vulnerable to the threats due to the lack of knowledge on how to manage their home network security.

There have been numerous security breaches on IoT devices and applications which caused severe damages to personal information. The

information theft happened through variety of IoT devices: (A. Tejasvi et al. 1-5), such as, IP surveillance camera system, IoT coffee port, and even from kids' IoT doll. There were huge number of incidents related to web cameras which are used video surveillance to observe their houses whether they are home or not. Many of the cases are related to the breaches on video surveillance and people were being spied on and recorded without noticing. As a result, ironically, the products that are supposed to give people relief are threatening their security.

Since the fourth industrial revolution, integration of business and technology has been booming and most of current businesses cannot separate technology from operations. Even small or local businesses transformed their payment and delivery system with ever-growing information technology because without proper IT integration, younger generation customer tend to leave for comfortable alternatives. After several years of development on the field of business, it has been spreading throughout home appliances and networks; a phenomenon now called IoT (Internet of Things). Since the outbreak of covid-19, the tide of IoT has been accelerated dramatically: (J. Steward). A lot of IoT devices are now within everyday lives of people that with a simple touch on anyone's smartphone can change the temperature of one's entire house or even huge facility. However, most of IoT users who are depending on home network tend to have a lack of knowledge on how to protect their networks. Most users heavily rely on the basic security features that are provided with small router or

modem they purchase or get serviced, even though the network connected to it controls everything inside their houses.

Problem Statement

There are many IoT security solutions out in the market for enterprise-level protection but not enough of resources are available for average end-users to learn how to securely manage and protect their information and privacy. Despite the comfort from IoT devices, it would be extremely hard for people with non-technical background to understand complex technology paper to build a sound and secure network themselves.

Therefore, many of the tech-giants are focusing on providing the comprehensive platform service that people need. This project focuses on the services of three world-famous tech-companies: Microsoft, Google, and Amazon. The project will focus on how the security is applied, where it works the best, and what would be the best practice for different spheres based on the study conducted: (P. Pierleoni et al).

The major aspects of this project focus on comprehensive services provided by three companies: AWS IoT Core, Microsoft Azure IoT Hub, and Google IoT Core. Three different services will be analyzed on the aspect of:

1. What technologies are behind the service?
2. How are the technologies integrated with other services?
3. How well do the three platforms perform in different workloads?

4. Which field of business or personal need will effectively utilize the service?
5. What are proper options that users can choose based on cost variation?

CHAPTER TWO:

CASE STUDIES

There are numerous IoT devices and applications which support comfortable usage of customers in every area of the life. For example, electric vehicle charger that support Android application and Bluetooth connection: (“Kaspersky Lab Security Services”), smart meter for home electricity usage, Fitbit area tracking personal health information, Google Nest thermostat: (G. Hernandez et al. 1-8), Tesla electric vehicle, chamberlain myQ for home garage door access, drones for work and fun, IP camera system for home surveillance, and millions of other devices are out in the market to attract customers with their features that will let people have more comfort. However, these devices and systems listed have been susceptible to cyber-attacks. Information theft on any of the devices connected to home or personal network can lead to a life destroying results.

In the first case of Chargepoint Inc. Describes vulnerable software and firmware where attackers can easily compromise connectivity. EV home charger from Chargepoint Inc. was vulnerable on password authentication phase by letting attackers bypass the process by simply changing “branch if equal” (b e q) to “branch if not equal” (b n e) in debug mode. After successful change, attackers could exploit a buffer overflow into the communication of android application and BTclassic: Bluetooth executable process. It carried out the denial of service (DOS) attack. It was tested that after gaining full access to the EV charger at

home, attackers could disable the user's entire electrical system, which will lead to a physical damage.

The second case is about one of the well-known attacks, eavesdropping/ Man-In-The-Middle (MITM) attack which enables attackers to extract network information they want. The attack was done on fit-bit aria, a smart scale, that helps people log their personal health information. Fitbit aria sends users' health information to their server for users to keep track of their health. Not only the health information of users, but attackers were also able to gain access to the network by finding service set identifier (SSID) and pre-shared key (PSK) from the log files of WireShark. The attack was done in simple steps:

a. Set up DHCP server to assign a proper IP address
b. Set up VM to forward IP packets to wlan0 interface
c. Set up "hostapd" as a wireless access point (WAP)
d. Use WireShark to sniff network traffic
e. Attackers gain full access to the network

Next case is about the device that controls and manages the thermostat from tech-giant, google. Google nest was highly susceptible when it was on device firmware update (DFU) status. When user press the hard-reset button for firmware update, it allows data input with bootable USB stick. Attackers utilized this feature and inserted customized image into the device rom. With x-loader and u-boot included in the customized image, attackers loaded the Linux kernel which has complete control over everything in the system. By executing kernel

with Linux inside the attacked nest, attackers gained root access and enabled secure shell (SSH) server installation and Odysseus malware to bypass network address translation (NAT). Nest, the thermostat, now worked as a botnet of home network. It had ability to access every part of the information at home: profiling, illegal surveillance, recording pictures, videos, and voices via connected IoT devices.

The fourth case is about a famous product of another major company, tesla model S. For tesla owners' convenience, tesla service centers and charging stations have TeslaService Wi-Fi SSID. Users' credentials are stored in tesla's web browser for auto-connect feature which is extremely comfortable for users. However, with fake SSID, attackers were able to redirect the traffic to their domain. Tesla's browser contained software bugs that granted attackers ability to read/write memory and execute customized code access shell. After gaining root access, they disabled security module, AppArmor. For the last step of attack, they used insecure token to bypass gateway integrity verification to access Engine Control Unit (ECU), which commands control of vehicles. Therefore, attackers obtained full control on both standby and driving modes. With this security flaw, not only the intellectual property could be stolen, but terrifying results could also be made to anyone in the car.

The fifth case indicates non-technical but effective method for attackers, social engineering attack. The case study on chamberlain MyQ: (J. Margulies 80-83), which is a garage door opener, getting affected on confidentiality and

integrity of data. The study shows that this smart home appliance is susceptible on being exploited by attackers accessing personal data and control of door locks and sensors taken over. As chamberlain MyQ not requiring password strength guidelines, it enabled attackers to use brute force attacks, such as, dictionary attack, to crack the password and doors to lock and unlock. Furthermore, this appliance used unencrypted user datagram protocol (UDP) to communicate between server and the device. It helped attackers to easily spoof the information during communication and steal the credentials being revealed. It shows that with simple dictionary attack and spoofing tool, anyone's home could be on the line of being physically breached.

The sixth case explains how someone's toys could be hacked and used as criminal weapons. According to study conducted: (I. Astaburuaga et al.), Parrot AR 2.0 quadcopter is a drone that was susceptible to open port attacks. The case study used Linux network mapper utility (Nmap) to reveal open ports, port 21-ftp and port 23-telnet, that are used for remote access. First, ftp was used to upload a harmful firmware to the drone and made it inoperable. Next, with anonymous ftp login, attacker downloaded password shadow file and removed hash for new root password. Therefore, telnet access is granted with no password requirements, which means that attackers have gained full access to the system. Now the drone can be utilized by attackers on any of their illegal activities, such as, smuggling weapons, drugs, terrorist attacks, and other information thefts.

The last case is about the surveillance feature that is supposed to help prevent overall system of the home network. However, from the case of Edimax IP camera system is susceptible from how the basic system works among IP camera, controller, and registration on command relay server. Attackers started with the public IoT device infected with malware, which acts as a bot and sends TCP syn (synchronization messages). Then it explores stateless and guesses the mac address which gets the confirmation with acknowledgement of one of them. This software bot now registers to the server and gets packet with authentication information. Now the IP camera system is in the hand of attacker.

Above cases indicate how IoT devices that are currently sold in the market are not thoroughly designed to protect consumers from security breaches. Of course, there are ways to implement the security with additional technical updates. However, installing technical add-ons are not an easy task for average consumers of IoT devices. Therefore, this project focuses on IoT security services that are provided from three tech-giants: Amazon, Microsoft and Google.

CHAPTER THREE:

IOT SOLUTIONS

In this chapter, the paper will analyze different functionality and features of three different IoT services that are provided from amazon, google, and Microsoft: respectively AWS(amazon web service) IoT core, google IoT core, and Microsoft Azure for IoT. The chapter will follow the order of:

1) General IoT architecture and technical terms
2) Review of performance tests on each service
3) Cost evaluation of each service based on the controlled test environment and official documentations from service providers
4) Recommendations based on the performance review and the cost evaluation

Architecture and Terms

IoT Architecture

According to the study: (L. Hou et al. 32-39), The basic architecture of IoT can be explained as a 5-layer architecture: perception layer, network layer, middleware layer, application layer, and business layer (refer to figure1). The perception layer works as sensors and actuators for different features to function. The data produced from this layer is sent to network layer, RFID Wi-Fi, Bluetooth, infrared, etc., and moves the data to middleware layer. In this layer, the data is processed and makes decisions whether to deliver or require services

to application layer. Based on data sent, business layer manages and controls overall IoT system.

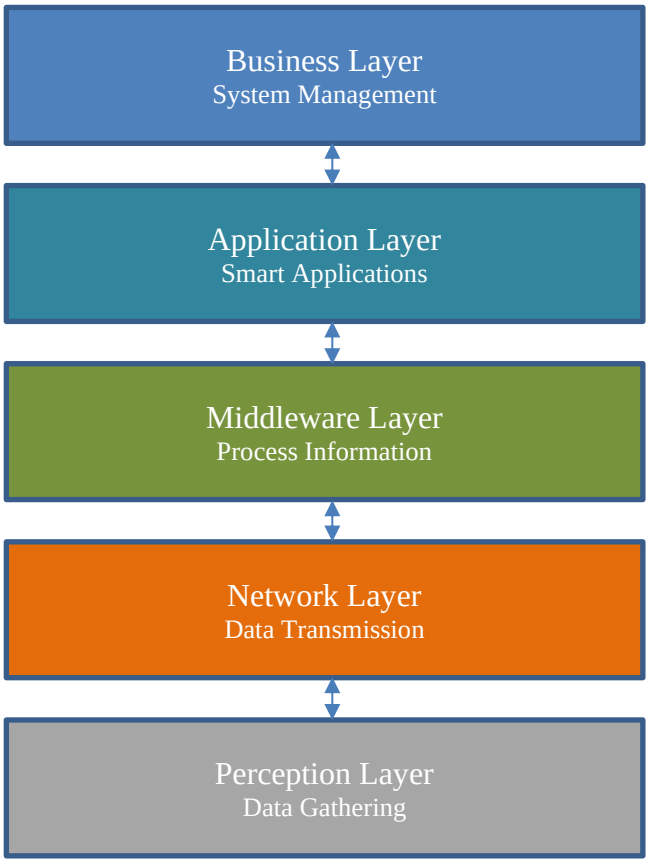

Figure1. Visual Explanation of 5-layer Cloud-IoT Architecture

MQTT

MQTT (message queuing telemetry transport) is a lightweight and simple messaging protocol: (D. Happ et al. 41-52). It supports multiple device connections which are constrained with low bandwidth. It is one of the best protocols that utilizes the communication among IoT sensor device (edge), MQTT broker, and monitor device. Two of the main functions include:

1) Send command to control output
2) Read and publish data

The basic concept of MQTT consists of three parts: publish/subscribe, topics, broker:

1) Publish / subscribe = a device can publish message on a topic and other devices can receive the message from the topic they subscribed. Topic
2) Topic = it is an interest on messages that specifies where the device want to publish. Topics have levels that are indicated with slashes:/. For example, it is indicated as home/kitchen/lamp for specific publication.
3) Broker = MQTT broker receives every message with filter from devices and published to all subscribed clients.

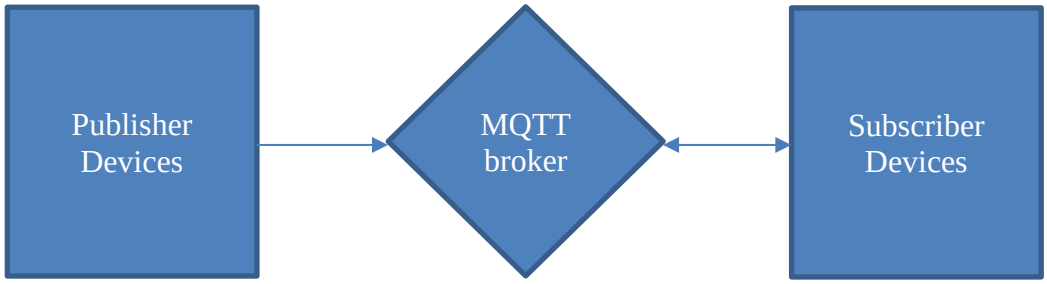

Figure 2. MQTT Communication Concept

QoS

MQTT supports three levels of quality of service. QoS level 0 is for delivery of one message without the confirmation of reception. QoS level 1 ensures every message to be delivered for once at least and reception acknowledgement message is required. QoS level 2 supports four-way handshake communication which ensures that one message is sent to the subscriber exactly once. QoS level 1 is used on every service provider in the paper, therefore, performance measurements will be based on the round-trip time of the messages from publisher. Microsoft Azure IoT Hub offers QoS level 2 service but not recommended due to increased latency and fluency of service.

API

API is an Application Programming Interface which allows multiple software applications or hardware-software mixed intermediaries to communicate. This interface aggregates requested information from different sources of databases, even from third parties, to have an extended features and functionalities that users can utilize. API also adds security on personal data because applications or software using API to communicate asks for permission to access the data. One type of API is REST API, Representational State Transfer API. It is a powerful tool that is simple and standardized for industry use. It also allows the interactions with restful web services. REST is a patterned

architecture between systems using http to operate and gain data from any possible formats, such as XML and Json.

SDK

SDK is a Software Development Kit that has one installable package with a collection of software development tools. It contains software framework, complier, and debugger which are to be facilitated. SDKs are usually customized for specifically on different hardware platforms or operating systems. It allows developers to have easier creation of applications or software with an ability of calling pre-made codes or frameworks from the library of programming languages.

Amazon Web Service IoT Core

Amazon offers comprehensive IoT management service, AWS IoT core, that allows users to audit configurations of connected devices and monitor map of connected devices for abnormal activities. Whenever IoT core detects abnormal activities, it pushes an alarm for users to take any actions it requires. The overall process of communication with AWS IoT core starts from connected devices reporting their states with MQTT publishing messages on certain topics. It has a hierarchical name order system to obtain identities of devices. Then the message is sent to MQTT broker which sends message to all subscribing clients. Each connected device stores and retrieves their state information in Json file with a current state and a desired state. At the last step, rules engine processes message and integrates other AWS services.

AWS IoT core comes with AWS IoT device management service that allows IoT platform to organize, monitor and manage IoT devices. AWS IoT device management has features to register devices in bulk and organize devices in groups with access policies attached. Also, it is possible to work with registry via AWS IoT console or AWS command line interface. Compatibility of AWS IoT core shines with device SDKs for Android, iOS, Java, JavaScript, C++, Python, and embedded C along with open-source libraries. Along with SDK usage, AWS IoT cli and AWS IoT API to create applications with http/https requests and device SDKs. Other services are provided which utilizes to collect and process data. Amazon kinesis data stream for real-time data stream, AWS

lambda to perform serverless code, amazon simple notification service for notifications and alerts, and amazon simple queue for storing data in a queue are supported.

As mentioned, AWS IoT core communicates in MQTT v.3.1.1 which does not support QoS level2. AWS message broker uses MQTT QoS level 1 to publish or subscribe, and https to publish. However, it does not allow two or more clients to connect at the same time when they have same client id. For the use of rest API, message broker supports http protocol. To ensure the security of communication and process of data, AWS IoT core is integrated with transport layer security (TLS) which ensures all traffics between devices with credentials and message broker to be encrypted. For authentication of devices, the platform requires x.509 certificates to reach higher level of security.

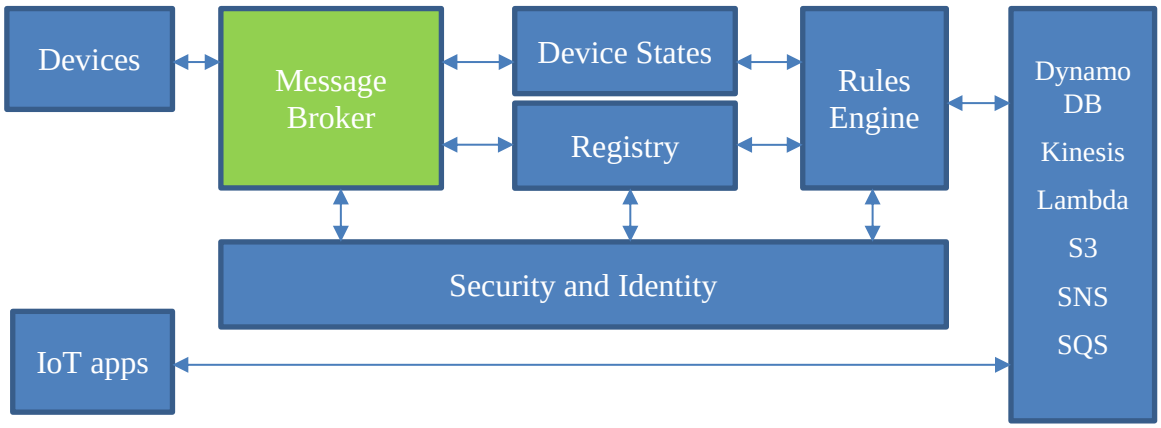

Figure 3. AWS IoT Core Architecture with Integration

Microsoft Azure IoT Hub

Azure IoT hub is a fully integrated service with PaaS solution and SaaS solution, respectively, platform as a service and software as a service. PaaS solution is provided as Azure IoT solutions accelerator and SaaS solution comes as Azure IoT central. Azure IoT hub is utilized as cloud gateway which in AWS uses message broker. It accepts data securely and works as a device manager. Thus, IoT hub integrates with other Azure cloud services natively, which in turn, offers bi-directional communication in the relationship of devices and applications. Azure for IoT has a 3-layer cloud-IoT architecture to operate. When message arrives at the hub, it is sent to one or more endpoints by its built-in message routing function. Similar to AWS IoT core, devices have a virtual representation but, in the cloud, twin device. Device identities are stored in the twin device in Json document with reported properties presenting current state and desired properties.

Microsoft Azure offers Microsoft Azure IoT hub device provisioning service that enables real-time provisioning of devices connected to hub with no human effort required. When devices are registered with IoT hub, the desired twin device states are populated. Also, device SDKs are provided with availability on .net, c, java, node.js, python, and iOS for simplified connectivity. As mentioned above, IoT hub communicates in bi-directional way between devices and applications, it also communicates for device-level identity to and from cloud. Azure IoT hub supports variety of communication protocols, such as, MQTT 3.1.1, native http

over TLS, and AMQP 1.0 with optional WebSocket support. Optional WebSocket feature enables the persistent and bidirectional connection between a client and server. Different from other service providers, Azure IoT hub offers QoS level 2 message delivery assurance, but it is not recommended due to the increased latency and the impact on the availability of the system.

In the security of Azure IoT hub, it is segmented in three areas:

1) Device: Azure Hub Identity Registry has secure storage for each device identity and security key.
2) Connection: To initiate connection, devices should connect to the Hub not connected from the Hub, along with TLS authentication with X.509 certificate.
3) Cloud: For user authentication and authorization, Azure Active Directory is used for cloud access.

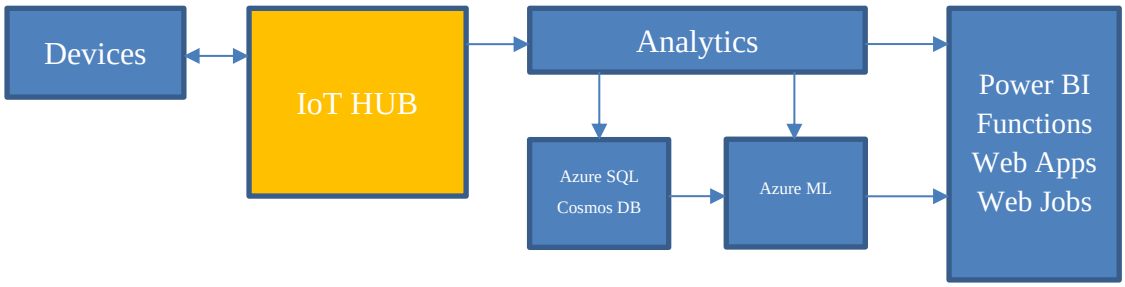

Figure 4. Microsoft Azure IoT Hub Architecture with Integration

Google IoT Core

Google's integrated solution for IoT is Google IoT Core which comes with comprehensive features. The architecture of Google IoT Core has two main parts: device manager and protocol bridge. The main function of device manager is to register devices with the service. On the other hand, protocol bridge utilizes two protocols, HTTP and MQTT, to connect and send data from devices to the cloud or vice versa. The whole process of data flow comes in this order:

1) Google IoT Core gets the data sent from devices and directs the data received to Google Cloud Pub/Sub: Enterprise message-oriented middleware that has message ingestion service.
2) Messages go into Google Cloud Data Flow, a pipe-line service, which process and sort data for different cloud services.

Each device registered to the IoT Core is represented with ID and full resource name is used to identify devices. Google IoT Core has a special feature that differs from other platforms previously discussed. It allows users to define custom metadata, a state from cloud, and a configuration.

Like the other two IoT solution platforms, Google IoT Core supports HTTP and MQTT for data communications and management of devices. By utilizing MQTT, devices cannot maintain connection to the IoT Core, but they can send requests and receive responses. With MQTT, devices can send publish requests to specific topics and offers QoS level 0 and 1 from MQTT bridge. Like other Cloud-IoT platforms, Google IoT Core comes with SDK, Google Cloud SDK, with

its own command line tool: gcloud. With the use of console or APIs client library, operations are possible on C#, Java, NodeJS, GO, PHP, Python, and Ruby. The already versatile IoT Core also natively integrates with Cloud ML, Data Studio and DataLab, which are big data and machine learning analysis services from Google.

Different from other service providers, Google IoT Core uses Json Web Tokens for authentication of each device with public or private key. To increase the level of security, IoT Core integrated RSA for secure data transmission and Elliptic Curve algorithms to verify signatures. For the security of communication, TLS 1.2 protocol is required for MQTT connections for the use of root authorities. To manage access, authentication, and authorization on IoT Core API, Google Cloud Identity and Access Management (IAM) is provided.

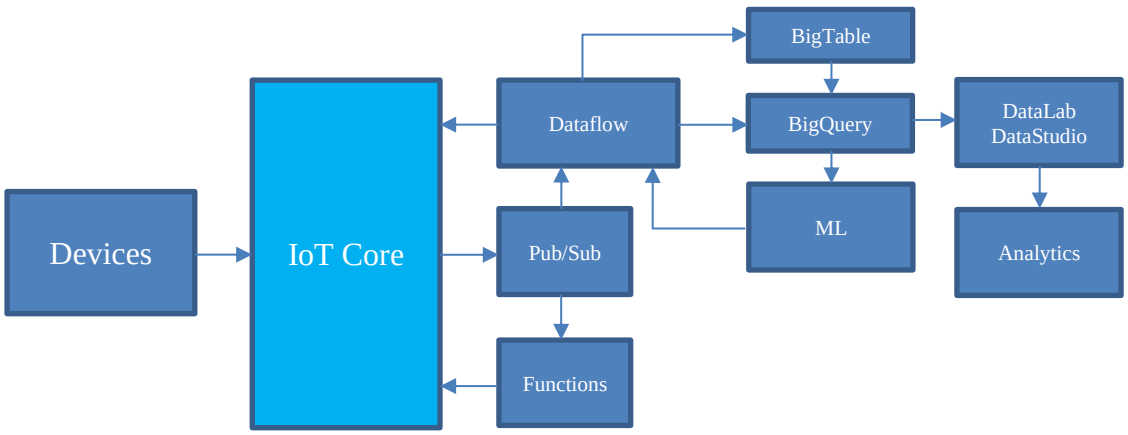

Figure 5. Google IoT Core Architecture with Integration

CHAPTER FOUR:

METHODS

Performance Analysis

The analysis is referred from previously conducted test: (P. Pierleoni et al). The test was simulated on the setting of one computation machine with following features: Intel Xeon X5650 (x2) CPU, 12 MB cache, 2.66 GHz, 16 GB RAM with Ubuntu 18.04.1 LTS. To obtain the concurrency of the tests, clients are implemented in GoLang developed by Google. Test environment was controlled with different parameters implemented: MQTT broker endpoint, scenarios based on different number of clients, number of messages, interval between messages in ms, size of messages, and Pub/Sub QoS. However, even in the strictly controlled testing environment, the performance of cloud service, which is one of the fundamental parts in IoT services, may vary in many situations. Thus, 42 different measurements for each simulation are made. For example, 2 tests per day in different times over 3 weeks. Each simulated test computed mean value of the cloud service time for each simulation and its standard deviation writing results to their database. However, the limitations are applied due to the utilization on free edition of IoT services.

One-to-One

The test was conducted with one client device connected up-to the value of 100 mps and increased the number of clients from 1 to 600. Each client had the fixed sending frequency of 10 mps. Azure IoT Hub was the only exception due to the free-tier service option that has the limitation of accepted connections per second.

Many-to-One

On this next scenario, the test conducted was based on a single subscriber that subscribes to all topics and more clients publishing message on its own topic. However, Google IoT Core and Microsoft Azure IoT Hub do not allow direct wildcard subscription, however they allowed forwarding messages to other additional services. On Google platform, all virtual devices are registered in a registry which has related topic in Pub/Sub service. Each device sends message to its MQTT topic and IoT Core forwards the message to Pub/Sub. On the other hand, Azure IoT Hub allowed native integration with one or more endpoints, also client was directly connected to MQTT broker subscribing all topics.

Broadcasting

In this last scenario, single producer generated 10 mps in a single topic and increasing number of clients were subscribed to topic. It is a broadcasting

scheme that one message is published on a topic that multiple subscribers listen to.

Cost Analysis

Cost analysis will be conducted based on the official document from IoT service providers.

Billing system of AWS IoT Core charges separately:

1) Connectivity usage
    a. Metered in 1-minute increments based on the total connected time of devices: $0.096 per million minutes
2) Messaging
    a. Metered on the number of messages transmitted: $1.20 below 1 billion messages, $0.96 for next 4 billion messages, $0.84 over 5 billion messages
3) Device state storage usage (Device Shadow)
4) Device meta data storage usage (Registry)
5) Message transformation and routing usage (Rules Engine)

Rates differ based on selected regions.

In the case of Microsoft Azure IoT Hub, costs are managed in two levels of service: Basic edition, Standard edition. Each level has three different tiers of service. Each tier has limits on daily message, throttling will be applied after exceeding the daily limit. Every consumption made are measured daily and

charged monthly. To sum up, customers of Azure IoT Hub will be charged based on the number of Hub units and the amount consumed in month.

Costs on the usage of Google's service is calculated on how much data is used in a month. Google IoT Core has four tiers of costs calculated differently. In case of creating, reading, updating, and deleting device connections will not be charged. However, Google's solution applies the minimum message volume as 1024 bytes, which means messages below 1024 bytes will be counted as 1024 bytes. The pricing is listed on the table below.

CHAPTER FIVE:

RESULTS

Performance Analysis Result

One-to-one

On this scenario, the basis is to conduct the cloud service times in relation to the number of messages published per second. Basic concept of this scenario is based on setting the number of publishers is equal to the number of subscribers and each is assigned on a single topic. Result of the tests showed that Google IoT Core responded faster than other IoT service platforms between 150 mps and 750 mps. AWS performed better on the range, which was out of 150 mps – 750 mps, but overall performance for daily usage is better with Google IoT Core. Even with the less load conditions on Azure IoT Hub, average service time took much higher than competitors. Surprisingly, all platforms provided stable performance even with the increase in load.

Different result came out when the number of clients was fixed to 100 and the load on message broker was increased. The test result showed even more stable performance for all platforms, however AWS performed slightly better on every mps difference. The most surprising part of the test results is on Azure IoT Core which showed the most symmetrical distribution overall. However, all of three platforms showed stable service time results.

Many-to-one

In the case of Amazon, it was worth noting the sharp increase on message loss when the mps was exceeding 400 mps, 40 clients were connected, and each client sent 10 mps. Significant message loss was depicted on exceeding 70 clients with 5% message loss and tremendous increase at 800 mps of 20% loss. After 810 mps, AWS stabilized at 42% message loss rate.

The performance result showed similar result to the result of previous scenario. In the environment of increasing the number of clients from 1 to 600, Google IoT Core showed significant increase in cloud service time after reaching 4000 mps sent by clients with 10 mps/client. Compared to Google, Amazon IoT platform showed less increase in cloud service time at the same point. However, this result does not impose the meaning that the tested services are not functioning normally because it was due to the limited ability of QoS 1, which delayed the forwarding intentionally. Even in the different scenario, same result was brought out from Azure IoT Hub. It seemed different scenario did not affect the performance of Azure.

Many-to-many

Google's cloud service time was lower than both Amazon and Microsoft beyond 15 connected subscribers. For the section of below 15 subscribers, AWS had the lowest cloud service time. Shockingly, Azure's IoT Hub forwarded messages 15 times slower than other two IoT service platforms. However, Azure

IoT Hub followed the previous results on having the lowest gap between outliers. When the test started with one subscriber, Google, Amazon, and Microsoft respectively showed the cloud service time of 26.479 ms, 24.991 ms, 160.567 ms. However, when the number of subscribers reached 300, the difference was 26.7%, 68.1%, and 7.1%, respectively in the same order.

Table 1. Different Scenario-Based Tests Comparison

	One to One	Many to One	Many to Many
Google	Stable at 26ms throughout 1000mps to 3500mps / Stable at 27ms throughout 4000mps to 6000mps	Stable at 26ms throughout 1000mps to 3000mps / Between 4000mps and 5000mps, dramatic increase from 31ms to 44ms and stabilizes after 5000ms at 45ms	Stable at 20ms from 0 to 170 connected subscribers / Stable at 25ms from 200 to 250 connected subscribers
Amazon	Stable at 29ms throughout 1000mps to 6000mps	Stable at 26ms until 3500mps / Stable at 33md between 4000mps and 6000mps	Stable at 25ms from 1 to 100 connected subscribers / Increase from 25ms to 37ms at 150 to 220 connected subscribers / Stable at 40ms from 220 to 300 connected subscribers
Azure	Stable at 160ms throughout 1mps to 100mps	No difference	Stable 160ms to 170ms throughout 0 to 300 number of connected subscribers

Cost Analysis Result

Table below shows a different costs variation of tiers:

Table 2. Azure IoT Hub Costs on the Number of Daily Messages Per Unit

Tiers	Monthly Cost / Unit	Message/Day/Unit
Free	Free	8000
Standard 1	$25	400,000
Standard 2	$250	6,000,000
Standard 3	$2,500	300,000,000
Basic 1	$10	400,000
Basic 2	$50	6,000,000
Basic 3	$500	300,000,000

Table 3. Google IoT Core Pricing Model Based on Data Volume

Price per MB	Monthly Data Volume
$0	Less than 250 MB
$0.0045	From 250 MB to 250 GB
$0.0020	From 250 GB to 5 TB
$0.00045	Over 5 TB

Table 4. Cost Comparison Based on the Number of Devices Connected

Number of devices	Azure basic	Azure standard	Aws	Google
1 ~ 6	$10	Free	Below $15	Free
7 ~ 70	$10	$25	Below $3	Below $10
70 ~ 250	$10	$25	$3 - $15	$10 – $45
250 ~ 1000	$50	$250	$15 – $56	$45 – $185
1000 ~ 4100	$50	$250	$56 – $230	$185 – $810
4100 ~ 10000	$500	$2500	$230 – $560	$810 – $1440
10000 ~ 50000	$500	$2500	$560 – $2500	$1440 – $4640
50000 ~ 100000	$500	$2500	$2500 – $4800	$4640 – $8640
100000 ~ 420000	$500/$1000	$2500/$5000	$4800 – $17700	$8640 – $16300
420000 ~ 500000	$1500	$7500	$17700 – $21058	$16300 – $17815

The price analysis is based on each device connected continuously and sends one message per minute of 1kB. Monthly traffic volume is calculated in:

[Number of connected devices * 1440 messages/day * 30 days]

Some sections of costs are underlined to highlight with platform offers the lowest costs. The table will help potential users who are considering to utilize one of the Cloud-IoT solution for their own IoT devices and management.

CHAPTER SIX:

DISCUSSION AND AREAS FOR FUTURE PROJECTS

After conducting thorough review on different real world case studies of current IoT device security vulnerabilities, there are numerous active threats prevalent. Mostly, devices were susceptible on its own software or firmware that the communication between devices and server could be intercepted by attackers for malicious uses. Possible attack vectors varied from the software to node itself. Also, the possibility of damage from the impact varied tremendously due to the nature of different devices. However, the most critical point of the studies indicates that the damages from manufacturers' overlooked security vulnerability should not be the burden of rightful users. Therefore, normal users should consider utilizing Cloud IoT platform as a solution for their promising security on personal information. Since the theft of personal information would result in reputational, financial, physical, and many other disastrous results.

To implement the optimal solution, the paper analyzed the tests done by Pierleoni et al. which conducted three different scenario-based tests on Cloud based solutions, respectively: Amazon IoT Core, Google IoT Core, Microsoft Azure IoT Hub. Even though all three platforms used the same communication protocol, MQTT, they had different architectures using unalike processes. Tests were conducted to compare service times with fixed message size and incrementing number of messages and connected devices in free tiers of each

platform. Performance analysis showed similar result for AWS IoT Core and Google IoT Core, but the performance of Microsoft Azure IoT Hub was significantly lagging behind compared to the other two platforms in every aspect. Not only the performance of different solutions was analyzed and compared, but also the pricing model is organized in the paper for easier comparison. However, test itself imposes the limitation of study due to the limited number of connected devices and fixed packet size since the tests are intended to help normal users' choice on which platform to utilize for their own best use.

As stated above, imposed limitations of scenario-based tests included free tier limitations and only tested on the increasing number of devices and messages, not on the decreasing number. Free tier was restricting the number of connected devices and messages which could be a possible obstacle for users who are facing different situations or surroundings. Future studies will be conducted on different paid levels to conduct how each three platform behave differently. Also, there will be a study on different behaviors based on different packet sizes and communication protocols, such as, HTTP and AMQP. It is important to conduct performance evaluations on different load levels but there should be a continuous study on current vulnerabilities and threats since the technology used in the world is ever evolving.

APPENDIX A: ABBREVIATIONS

IOT = INTERNET OF THINGS

DOS = DENIAL OF SERVICE

MITM = MAN-IN-THE-MIDDLE

VM = VIRTUAL MACHINE

SSID = SERVICE SET IDENTIFIER

PSK = PRE-SHARED KEY

DHCP = DYNAMIC HOST CONFIGURATION PROTOCOL

DFU = DEVICE FIRMWARE UPDATE

WAP = WIRELESS ACCESS POINT

SSH = SECURE SHELL PROTOCOL

NAT = NETWORK ADDRESS TRANSLATION

UDP = USER DATAGRAM PROTOCOL

NMAP = NETWORK MAPPER UTILITY

TCP = TRANSMISSION CONTROL PROTOCOL

TCP-SYN = SYNCHRONIZATION MESSAGE

TCP-ACK = ACKNOWLEDGEMENT MESSAGE

IP = INTERNET PROTOCOL

HTTP = HYPERTEXT TRANSFER PROTOCOL

AWS = AMAZON WEB SERVICE

RFID = RADIO-FREQUENCY IDENTIFICATION

MQTT = MESSAGE QUEUING TELEMETRY TRANSPORT

QOS = QUALITY OF SERVICE

API = APPLICATION PROGRAMMING INTERFACE

REST = REPRESENTATIONAL STATE TRANSFER

XML = EXTENSIBLE MARKUP LANGUAGE

JSON = JAVASCRIPT OBJECT NOTATION

SDK = SOFTWARE DEVELOPMENT KIT

TLS = TRANSPORT LAYER SECURITY

AMQP = ADVANCED MESSAGE QUEUING PROTOCOL

MPS = MESSAGE PER SECOND

MS = MILLISECOND

KB = KILOBYTE (1024 BYTE)

MB = MEGABYTE (1024 KB)

GB = GIGABYTE (1024 MB)

TB = TERABYTE (1024 GB)

REFERENCES

Asensio, Á., Marco, Á., Blasco, R., & Casas, R. (2014). Protocol and architecture to bring things into internet of things. *International Journal of Distributed Sensor Networks*, *10*(4), 158252. https://doi.org/10.1155/2014/158252

Astaburuaga, I., Lombardi, A., La Torre, B., Hughes, C., & Sengupta, S. (2019). Vulnerability analysis of ar.drone 2.0, an embedded linux system. *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. https://doi.org/10.1109/ccwc.2019.8666464

Azure service Bus-Cloud messaging Service: Microsoft Azure. Azure Service Bus-Cloud Messaging Service | Microsoft Azure. (n.d.). https://azure.microsoft.com/services/service-bus/.

Barik, R. K., Dubey, H., Misra, C., Borthakur, D., Constant, N., Sasane, S. A., Lenka, R. K., Mishra, B. S., Das, H., & Mankodiya, K. (2018). Fog assisted cloud computing in era of big data and internet-of-things: Systems, architectures, and applications. *Studies in Big Data*, 367–394. https://doi.org/10.1007/978-3-319-73676-1_14

Botta, A., de Donato, W., Persico, V., & Pescape, A. (2014). On the integration of cloud computing and Internet of things. *2014 International Conference on Future Internet of Things and Cloud*. https://doi.org/10.1109/ficloud.2014.14

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: VISION, hype, and reality for

delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), 599–616. https://doi.org/10.1016/j.future.2008.12.001

Chapple, M., & Seidl, D. (2021). *Comptia security+ study guide: Exam sy0-601*. Sybex.

Free-fall: Hacking tesla from wireless to can bus. (n.d.). https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf.

Frustaci, M., Pace, P., Aloi, G., & Fortino, G. (2018). Evaluating critical security issues of the iot world: Present and future challenges. *IEEE Internet of Things Journal*, *5*(4), 2483–2495. https://doi.org/10.1109/jiot.2017.2767291

Happ, D., Karowski, N., Menzel, T., Handziski, V., & Wolisz, A. (2016). Meeting iot platform requirements with open pub/sub solutions. *Annals of Telecommunications*, *72*(1-2), 41–52. https://doi.org/10.1007/s12243-016-0537-4

Hedi, I., Speh, I., & Sarabok, A. (2017). Iot network protocols comparison for the purpose of iot constrained networks. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. https://doi.org/10.23919/mipro.2017.7973477

Hou, L., Zhao, S., Xiong, X., Zheng, K., Chatzimisios, P., Hossain, M. S., & Xiang, W. (2016). Internet of things cloud: Architecture and implementation. *IEEE Communications Magazine*, *54*(12), 32–39. https://doi.org/10.1109/mcom.2016.1600398cm

Incipini, L., Belli, A., Palma, L., Concetti, R., & Pierleoni, P. (2019). Databases performance evaluation for IoT systems: The Scrovegni Chapel use case. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. https://doi.org/10.23919/mipro.2019.8756813

Jenkins, G. (2000). *Clouds project cloudwatch*. Amazon. http://aws.amazon.com/cloudwatch.

Kumar, S. K., Satheesh, N., Mahapatra, A., Sahoo, S., & Mahapatra, K. K. (2019). Physical unclonable functions for ON-CHIP Instrumentation: Enhancing the security of the Internal joint TEST action Group Network. *IEEE Consumer Electronics Magazine*, *8*(4), 62–66. https://doi.org/10.1109/mce.2019.2905539

Liu, Y., Akram Hassan, K., Karlsson, M., Pang, Z., & Gong, S. (2019). A data-centric Internet of Things framework based on Azure cloud. *IEEE Access*, *7*, 53839–53858. https://doi.org/10.1109/access.2019.2913224

Lodge, D. (2015, January 6). *Are your smart weighing SCALES lying to you? Quite possibly (part 1)*. Pen Test Partners RSS. https://www.pentestpartners.com/security-blog/are-your-smart-weighing-scales-lying-to-you-quite-possibly-part-1/.

Margulies, J. (2015). Garage door openers: An internet of things case study. *IEEE Security & Privacy*, *13*(4), 80–83. https://doi.org/10.1109/msp.2015.80

Mazhelis, O., & Tyrvainen, P. (2014). A framework for evaluating internet-of-things platforms: Application provider viewpoint. *2014 IEEE World Forum on Internet of Things (WF-IoT)*. https://doi.org/10.1109/wf-iot.2014.6803137

Misic, J., Ali, M. Z., & Misic, V. B. (2018). Protocol architectures for iot domains. *IEEE Network*, *32*(4), 81–87. https://doi.org/10.1109/mnet.2018.1700395

Pierleoni, P., Concetti, R., Belli, A., & Palma, L. (2020). Amazon, Google and Microsoft solutions FOR Iot: Architectures and a performance comparison. *IEEE Access*, *8*, 5455–5470. https://doi.org/10.1109/access.2019.2961511

Sklyar, D. (n.d.). *ChargePoint Home security research*. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/12/13084354/ChargePoint-Home-security-research_final.pdf.

Smart nest thermostat a smart spy in your home. (n.d.). https://www.blackhat.com/docs/us-14/materials/us-14-Jin-Smart-Nest-Thermostat-A-Smart-Spy-In-Your-Home.pdf.

Steward, J. (2021, July 18). *21+ internet of Things STATISTICS, facts & trends for 2021*. Findstack. https://findstack.com/internet-of-things-statistics/.

Thangavel, D., Ma, X., Valera, A., Tan, H.-X., & Tan, C. K.-Y. (2014). Performance evaluation of MQTT and Coap via a common middleware. *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. https://doi.org/10.1109/issnip.2014.6827678

Treichler, R., & Hardmeier, C. (2005). *Schlagwortnormdatei Schweiz für allgemeine ÖFFENTLICHE Bibliotheken: Sns*. Amazon. https://aws.amazon.com/sns.

wesmc7777. (n.d.). *Azure iot hub scaling*. Azure IoT Hub scaling | Microsoft Docs. https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-scaling.