Fall 12-20-2018

Management and Security of IoT systems using Microservices

Tharun Theja Kammara
San Jose State University

Management and Security of IoT systems using Microservices

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Tharun Theja Kammara

November 2018

The Designated Writing Project Committee Approves the Writing Project Titled

Management and Security of IoT systems using Microservices

by

Tharun Theja Kammara

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

November 2018

Dr. Melody Moh	Department of Computer Science
Dr. Robert Chun	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science

ABSTRACT

Devices that assist the user with some task or help them to make an informed decision are called smart devices. A network of such devices connected to internet are collectively called as Internet of Things (IoT). The applications of IoT are expanding exponentially and are becoming a part of our day to day lives. The rise of IoT led to new security and management issues. In this project, we propose a solution for some major problems faced by the IoT devices, including the problem of complexity due to heterogeneous platforms and the lack of IoT device monitoring for security and fault tolerance. We aim to solve the above issues in a microservice architecture. We build a data pipeline for IoT devices to send data through a messaging platform Kafka and monitor the devices using the collected data by making real time dashboards and a machine learning model to give better insights of the data. For proof of concept, we test the proposed solution on a heterogeneous cluster, including Raspberry Pi's and IoT devices from different vendors. We validate our design by presenting some simple experimental results.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my project advisor, Dr. Melody Moh, for her excellent guidance, and encouragement throughout the course of this project.

My sincere thanks to my committee members, Dr. Robert Chun, and Dr. Thomas Austin for their useful comments, and their valuable time for reviewing my work.

My special thanks to my friends and family members who believed in me, who supported me in every step of this endeavor.

Thanks to the wonderful open source community out there for making source code available that was used for some parts in this project. Thanks to hacker community and security enthusiasts who provided some pointers to complete the malware testing part of this project.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

BOBE – break once break everywhere

CIO – chief Information Officers

DOS – denial of service

DDOS – distributed denial of service

GHDB – google hacking data base

HMM – hidden markov model

HVAC – heating, ventilation and air conditioning

IoT – Internet of Things

MITM – man in the middle

SCADA – Supervisory control and data acquisition

SVM – support vector machine

1. Introduction

There are a vast number of IoT devices currently in use in different forms. People are employing the services provided by IoT devices to assist in various tasks like home maintenance, health care, personal care, vehicular networks and industrial management. The amount of money spent on research and development of IoT is increasing with tech giants venturing in to the field of IoT either by directly buying IoT companies or by funding them. New innovations that can make life of average person easy have a huge commercial value and companies are competing with one innovation better than the other. Today IoT devices which were linked with wealthy lifestyle like Internet controlled microwave and internet-controlled switches have become affordable and are continuing to become even more available to all. All of these will contribute to the Gartner's prediction that the number of humans to online devices ratio would be 1:4 by 2020.

The current explosive growth of IoT also brought some problems with it like security and privacy concerns being the primary concerns. While normal people are using IoT devices to make their lives easier, attackers and cyber criminals are using them for malicious purposes. Attackers are modifying their attacks to take advantage of the massive number of IoT devices and security loopholes in them. Loosely defined security standards for IoT devices and not fully enforcing those security standards gives the attackers an advantage when compromising IoT devices. The modern-day attacks have become complex thanks to the inclusion of IoT devices and platforms in them. Massive DDOS attacks ranging over 600 gigabits per second have become common with many botnets available for sale for the non-tech savvy. There is a need to increase the security standards for IoT devices and to ensure that those standards are properly enforced. Creating an awareness of changing default device passwords and closed unused ports (like telnet) among regular users of IoT devices can also prevent part of the problem.

This rest of this report is organized as follows. Section 1 talks about the advances of IoT systems and types of IoT devices used. Section 2 explains the reasons about why IoT devices are vulnerable to attacks and different types of IoT malware available. Section 3 talks about the different botnet topologies and some techniques that are used to detect botnets. Section 4 talks about the proposed architecture. In section 5, the details of the experiment conducted are given. The Section 6 describes the data collected and results obtained from the experiment. Finally, section 7 concludes this project and points out any possible future research outcomes.

1.1 Advances of IoT systems

In this section we will learn about the advances in the field of Internet of Things (IoT). Internet of Things (IoT) is currently one of the most rapidly growing technologies. IoT can be defined as a group of smart devices collectively working to accomplish a task using internet. Smart devices

were initially small individual devices that had limited computing capabilities and an interface to connect or transfer data to the internet. Some of the earliest types of smart devices that were domestically used had sensors connected to them and can send the sensor data to a destination via internet or other medium like Bluetooth, ZigBee etc. The popularity of IoT and reduced production costs for the electronic devices bought in even more investments for IoT. This lead to the invention and applications of a variety of IoT devices.

1.1.1 Different types of IoT devices and their scope

There are a wide variety of IoT devices. We will classify them based on their application and further classify on the differences between their sizes and shapes.

1.1.1.1 General Purpose IoT devices

These IoT devices are ubiquitous in our daily lives. They are used by everyone for simple household tasks and functions. They range from small sensors to large Heating, ventilation and Air Conditioning(HVAC) systems.

Smart Lights: These were one of the first IoT devices to come become an integral part of domestic lives. Philips Hue lights have been prominent for a few years already and there have been some cases where they were exploited [1].

Smart Thermostats: According to survey, 33 percent of thermostats sold in 2014 were Wifi enabled and the there is a prediction that their sale will only increase. Google Nest is of the top manufacturer of smart thermostats with a major market share. Google Nest thermostats are connected to internet and can be controlled remotely using an app.
Smart Cameras / Security cameras: Security cameras have been around since few years, but the recent explosion of IoT boom made it easy for anyone to install a security camera with internet storage. Most of the cameras come with ready out of the box settings which makes it easy for just anyone to plug and use a security camera. Google Nest after the success of thermostats ventured into the security domain.

Smart Switches: Currently smart switches are the most popular IoT devices. There are getting popular among users with intention to conserve energy. These smart switches are connected to the internet and can be operated using apps that come with the switches. These are IoT devices with the least capabilities

Other smart electronic devices with IoT capabilities: Smart Televisions that can tune in your Amazon prime video or Netflix subscriptions and display content are not expensive anymore and they are getting cheaper day by day. A smart fridge that takes commands from app and a smart coffee maker that makes coffee when instructed over wifi or internet are not rare anymore. Some of us may already been using them or we can expect them to be in our homes very soon.

1.1.1.2 Special Purpose IoT devices

Medical / Healthcare devices: Health care devices occupy a major share of IoT devices. They are mostly in the form of wearables. They started with Fitbit and now have evolved to smart watches. Every major cellular company like Apple, Samsung, LG, Lenovo etc., and watch manufacturer company like Fossil, Skagen etc., have smart watches on their product list. Though these devices come with health monitor capabilities, many features like easy payment gateways (Apple Pay and Android pay), GPS and cellular functionalities have been added to them over the iterations of research and development.

More specialized IoT devices: While commonly used wearables like smart watches have some health monitoring capabilities there are some even more specialized medical devices like IoT pacemakers [2]. These pacemakers are connected to the internet and can the alert the doctor in case of emergency. These devices are predicted to reduce the human intervention in medical procedures by 60%.

HVAC's: Heating, ventilation and air conditioning systems are systems that are responsible for maintaining temperature of a space (home/office/lab) by controlling the devices are heaters, thermostats, air conditioners etc. HVAC systems in home are less complicated and may have a small raspberry device controlling the other devices. Industrial HVAC systems have special hardware and more processing power for controlling the other devices.

Other popular IoT devices include home assistants like Amazon Echo and Google Home, smart locks, Raspberry Pie enabled IoT devices for motion tracking in security cams, blocking adware like Pi-Hole etc.

1.2 Increase in the Number of IoT Devices

There are several IoT devices specified in the previous section. Most of the IoT devices have simple and single functions like smart lights, smart switches etc., and some of them have complex functions like smart watches, HVAC systems etc. Companies are investing heavily in IoT research and development. IoT technology gives the company an edge over its competitors and way to increase their profit over less expenditure. Most organizations are looking forward for IoT's to solve the technical problems for them. Chief Information Officers (CIO) who can visualize solutions and leverage IoT technology are in high demand currently in the job market. This necessity will lead to invention of new applications or types of IoT devices we have.

Gartner predicted that by 2020 there would be 20 billion IoT devices connected to the internet and the ratio of number of devices connected online to human beings would be 4 to 1. These devices would vary from general purpose cellphones, tablets to specialized vending machines and Jet engines [3].

Many organizations are trying to bring IoT into our daily lives with a wide spectrum of creative products. There is research going on in the home assistance area by Google and Amazon. Apple and other mobile manufacturing companies have invested billions in IoT research and development. Apple is developed own home IoT app called home kit which can integrate

multiple home devices and control them. Ericsson, Hewlett and Packard are entering the IoT space with new products. Oracle bought Opower [4], a company that makes IoT meters to track energy usage of millions of homeowners across United States. Microsoft bought Solair [5], a company that analyses IoT device data. After observing a major financial value in IoT, all the major players are venturing into IoT with innovative IoT devices and multiple applications.

2. Iot Botnets Made Easy

Every day, few thousands of IoT devices are being added to Internet's compromised list of IoT devices and botnets. We will discuss reasons for IoT devices being an easy target for attackers across the globe.

2.1 Security Shortcomings In Iot Devices

There are many factors that make IoT devices vulnerable to attacks. Some factors arise from simple financial decisions taken by the manufacturer of IoT devices to save money while some are due to heterogeneous complexity of IoT systems. Low computing resources of IoT systems is also a reason for security shortcomings of IoT systems. Some of the most important factors are discussed below.

Lack of Quality Code: Majority of the code on most IoT devices is outdated. They use old code with deprecated protocols. The old protocols are proven to be vulnerable and yet the manufacturers don't upgrade the software used. It is also a common practice for a manufacturer to gather different pieces of software from different places on the internet and some parts written to patch all of that together. Most of it is spaghetti code and is difficult to maintain it unless a lot of time and money are spent. IoT manufacturers are only concerned about the profits and don't care about security of users [6].

Re-use of code: Almost all of the IoT manufacturers re-use some part of code like authentication protocols, communication protocols available freely on the net. A certain company A, can use the same code for all of its similar devices. They might have even got the code for free of the internet. An attacker who cracks one device of that company A can now gain access to all the devices of that company. This phenomenon is called BOBE (break once break everywhere). A recent example of BOBE is the famous Devil's Ivy vulnerability. Devil's Ivy vulnerability is a bug in gSOAP tool kit which is used extensively in physical security devices like security cameras, card readers etc. The bug was discovered by Senrio, an IoT focused security group on one model of security cam manufactured by Axis. The group was later able to exploit 249 models of cameras that are sold by Axis. On further looking into the vulnerability it was found out that the vulnerability lies in gSOAP code used by Axis. Axis is just one of the companies that is using gSOAP in their products. The vulnerability allows a remote user to send upto 2 gigabytes of payload to the affected device. There are at least 34 companies whose products have this vulnerability according to the creator of gSOAP [7]. The gSOAP vulnerability was patched quickly but it still unclear whether all the devices using vulnerable gSOAP were patched. We may still find some security camera models without a patched gSOAP tool kit.

Lack of security standards or guidelines: IoT has been popular over the last few years and there isn't a proper set of security standards that the devices should follow. This makes it easier for manufacturer to make more profit without adhering to security policies. Sometimes security standards can also hamper the security of IoT devices. For example, there was a rule earlier in United States that medical device software should be tested before released to users. According to the rule, a manufacturer of medical IoT equipment should test out all the devices again whenever he updates the software of patches it from vulnerabilities. This is a huge cost factor for the manufacturer and in most cases, they choose to not patch the vulnerabilities that were found. Now FDA wants to make sure that all the medical devices that can connect to internet come with a mandatory updatable software. California is the first state that proposed the Internet of Things cyber security law and that is going to effective starting January 1, 2020 [8]. Many industry experts say this may help the current state of security in IoT devices while security experts are skeptical about it. On the other end of spectrum there's 19 security guidelines for IoT devices for manufacturers to follow.

Light weight crypto systems: IoT devices come with limited resources for computation. A present-day cryptography algorithm with good strength needs more resources than that can be provided by IoT devices. As a result, IoT devices cannot be equipped with better crypto systems even though they are available. It is also observed that the Bluetooth protocol that is used by most of the smartwatches is vulnerable to Man in the Middle (MITM) attacks. The MITM attacks are done on Bluetooth secure simple pairing and it is also observed that Bluetooth protocol security depends on the capabilities of the device. So less powerful devices are easily vulnerable to Bluetooth attacks. There is a need for lightweight crypto systems that run without much computational overhead on less powerful devices.

Heterogeneous platforms: The Internet of Things is limited by the heterogeneous complexity in major aspects. The IoT ecosystem is complex with devices from different manufacturers with multiple software builds. It makes it difficult to manage the ecosystem. We may write a security software for one platform and chances are that they might not work on devices with other platforms. Even Orchestration and Management tools find it difficult to include all the southbound and northbound protocols used by IoT. A common solution that works for all the platforms of this diverse ecosystem is difficult and expensive to build. A user can have multiple devices from multiple vendors and it might be difficult for him to manage all of them from a common app. Research is going on in some of the fields like data storage for IoT that can be catered to all the platforms. If there is a breakthrough in security for one platform or one protocol used in IoT device, it might be difficult to say that most of the IoT vendors use that protocol or platform [6].

Default Login credentials: Most users using IoT for domestic purposes like security cameras, internet connected DVR players, smart Philips Hue lights are people without knowledge about security. Most of these devices come as plug and play devices and users don't bother to change the passwords once they are up and running. This makes it easy for attackers if they can reach the device. If the device has a public IP address, any attacker around the world can gain access of the device using the default login provided by manufacturer.

Lack of monitoring: Most users of IoT devices are non-tech savvy people. They employee IoT devices for the services the devices offer. These devices are used with the minimum or no security constraints. Even IoT devices that are deployed for industrial purposes are only monitored if they are functioning properly and not properly audited for security. Monitoring the IoT devices is not a function that comes out of the box with the devices. To monitor the IoT devices regularly, new tools, people, money and time to train the personnel are required. The extra costs associated with monitoring the IoT devices are not affordable for general users and mid-size companies. So, they only worry about the functionality of the device, rather the security. This lack of monitoring makes it even more easier for the attackers to compromise and take control of an IoT device. An IoT device may be part of a botnet and doesn't come under suspicion as long as it performs its functions properly.

2.2 Searching For Vulnerable Iot Devices

Finding vulnerable IoT devices is relatively easier now than it was few years ago. Today an attacker doesn't have to do all the hard work and can use tools available on the internet at their disposal. We will tell you about few such tools that have very large databases of vulnerable devices of IoT devices around the world.

SHODAN [9]: The go-to place for researchers, students and attackers. Shodan is a database of IoT devices on the internet. The site states that it to serve research purposes but there is some vital information in the site that can do a lot of harm when used by malicious users. The site has public IP address of IoT devices along with filters based on type of devices, password types, manufacturers etc. For example, a user can search specifically for webcams with default passwords. Any user can copy the address of webcams from the results of the search and login using the default login username and password. There are search filters to show IoT devices without passwords, IP address of refrigerators, DVR players etc. The Shodan database is updated daily with thousands of devices from around the world. It also has plugins that integrate it with pen testing frameworks like Metasploit. It is not clear on how much of the data is used for academic purposes, but it can be the first place to visit to increase the bot count in the botnet.

Google Hacks [10]: Google hacks or dorks are a good way to search for devices with vulnerabilities online. Google dorks can be found online by searching for google hacking database (GHDB). The site has a search bar to search for devices and gives a search string. The search string when used with google search engine displays all the publicly available devices that match the query. One example is to search for cameras by a certain manufacturer in GHDB. Paste the Google dork (search string) given by GHDB in google search engine and it would display IP addresses of all the devices by that manufacturer. This google dork can be used to find login pages of security cameras, routers and other similar devices

ERIPP- Every Routable IP Project [11]: this is a project similar to SHODAN, but this only collects IP addresses of routers with port forwarding enabled on port 80. With the IP addresses of routers, the attackers can do some information reconnaissance on the router and gain access to the network behind the router. If the router is powerful enough, it can be turned into a bot along

with the devices in the network. Project ERIPP has 5 gigabytes of data files with 34 million active routers around the world. ERIPP scans every IP address in the public domain range and stores it in the database when a port forward is identified on port 80. The project performs a scan from a hosted server and the database will be updated daily.

The conventional way: The conventional way to scan for vulnerable IoT devices is to scan every IP address and try to login into each of them. Researchers have tried this method to scan for vulnerable Supervisory control and data acquisition (SCADA) IoT devices. A python program was used to ping all the IP address from SHODAN database. The program would try to login to their telnet, SSH and other remote protocols. Banners were collected from those login pages into a database. Once the database had banners, the python program tried to login into the device by using default usernames and passwords corresponding to the organization the banner belonged to. If the banner showed HP in its login page, the python script would try all the default passwords from the official documentation on HP site for various devices. This method was successful, and access was gained to thousands of devices [12].

These methods allow attackers to collect data easily because data is already being collected by third party tools and scanners. Attackers can still search through old data and get rid of outdated data such as old IP addresses easily. The next way to look for bots is to code the malware to scan the whole internet directly.

2.3 Iot Malware

Malware for IoT evolved from simple worm programs that propagates, to complex malware that is resistant to device reboots. The following sections describes some malware that acted as milestones to present scenario of malware.

Linux.Aidra: This is reputed to be the first known malware capable of infecting IoT devices. This was discovered by a group of security researchers at ATMA.ES [13]. This malware was discovered after increase in telnet-based attacks from setup-boxes, DVR players and security cameras amongst other IoT devices. This malware was written for devices with ARM architecture running Linux operating system. This malware was also compiled for other architectures like MIPS, X86 etc. Once the malware infects the device via default telnet, it would try to download all the executables for different architectures. The executable suitable architecture runs correctly and will connect to the C2C server. A new variation of this malware was found in 2014 that was capable of mining bitcoin on the infected device [14].

Linux.Darlloz: Also known as Zollard malware. This was initially a worm to infect using a vulnerability in PHP web servers. In the initial step a POST request would be sent to the webserver and the vulnerable web server would download and run the worm. This worm would then create files on the local file system and starts its own web server, closing the already existing web server. This worm was also capable of running on IoT architectures like MIPS, X86, ARM, PPC etc. [15].

Mirai: This is one of the game changing malware in recent times. Mirai is popular in association with the massive DDOS attacks done on websites like krebsonsecurity.com, French hosting provider OVH and Dyn, a DNS service provider. The DDOS attack on krebsonsecurity reached 620 Gbits / sec while the attack on Dyn reached an alarming rate of 1.2 Tbits/ second. The author of Mirai is anna-senpai who released the source code for Mirai online on hackforums [16]. The Mirai virus attacked IoT devices over telnet with a preset list of 60 usernames and passwords. The Mirai malware at its peak had close to 65k infected bots in the botnet. Once a target was infected it would rigorously scan the network it is part of. The massive network traffic was one of the major indicators that the devices are infected by malware.
Mirai malware was termed a game changer as the author of malware hinted towards the competition in IoT botnet domain. The author of the Mirai malware updated it after the initial release so that it can block other IoT malware from infecting the devices which are already infected with Mirai. Mirai malware is known to stop and instance of qbot running and also blocks remote administration port [17].

Anna-senpai, the author of Mirai malware made the code for malware open source. This led to birth of different strains of variants of Mirai malware. Some of the variants are even more powerful and sophisticated in than the original Mirai malware. One the post where he released the source code, Anna-senpai also describes the setup of 2 servers, one for CNC and the other for database to run a basic version of Mirai bot. Once the bot starts running, it would scan the whole internet for devices with open telnet. It was hard coded in the Mirai bot to exclude some sites like defense sites, sites from security companies like MacAfee, Symantec etc. from scanning. The scan would infect all the devices possible and send the username and passwords to the database connected to CNC. The CNC is advanced enough to send commands like scans, Http flooding etc.

Satori, a new strain of malware originated from Mirai is also among the popular botnets for DDOS. Satori like Mirai scanned the internet for devices but it was built around vulnerabilities in 2 devices [18]. One was the code execution vulnerability in miniigd SOAP service in Realtek SDK and the other was undiscovered Zero-day vulnerability in Huawei HG532e home gateway. Exploiting a zero-day vulnerability is a new approach for botnets. There are also other variants of Mirai just as powerful as Satori.

Hide and Seek (HNS): this is relatively new botnet that is still in evolution phase. This bot was first discovered in Jan 2018 with advanced peer to peer communication capabilities. This malware would try to login with a preloaded set of default passwords and usernames. Once a device is infected, it would look for its neighboring devices in the same network. This malware can also setup a FTP server for other neighboring devices to download the malware. Recent findings show that this malware is updated with a new feature. i.e., the ability to copy itself to Linux boot process folder and thus respawn even after the device is restarted. General malware gets cleared once the infected device is restarted but this malware stores copy if itself in /etc/init.d/ and will respawn after a device reboot. The HNS malware may be equipped with even more features for greater damage as it is still in evolution phase [19].

3. Botnet Detection

There are different types of botnets based on their communication with the command and control server, mode of infection (HTTP, UDP etc.) and complexity. The botnet topologies and some existing methods for detection are discussed below.

3.1 Botnet Topologies

This section will describe some existing and upcoming trends in botnet detection. A botnet is a network of controlled servers via a Command and Control (CNC or C&C) server. A CNC server is also referred as botmaster or bothereder. A CNC server can be a single server or distributed over several servers with each server having its own functions. The CNC sends commands to the bots (or zombies) in the botnet and bots execute those commands. The actions of the bot depend on the complexity of the botnet and C&C server. Bots can be used for launching DDOS attacks, stealing user info and financial data etc. New botnet malwares are being developed that can steal the processing power of the infected bots for cryptocurrency mining.

Bots contact CNC servers using different approaches. Some approached can be classified as follows:

Centralized: This is the easiest approach to implement in a botnet. In this approach, all the bots have direct connections to the CNC server. Bots using this approach can easily be stopped by taking the CNC server. i.e., CNC serves as a single point of failure. The advantages include low latency and easy to code and implement structure.

P2P: In P2P approach, each bot has the capability to act as a CNC server. Once a bot receives commands from CNC (can be its neighbor) it will execute the commands and transmit the commands to the bots connected to it and thus acting as CNC server. P2P botnets are difficult to stop because they don't suffer from single point failure of CNC [20]. We need to take down most of the infected bots to prevent the infection from spreading. The disadvantages of this include low latency and complexity. Once there are significant number of botnets in the network, it is easy to discover because of the increased traffic between infected bots. The speed at which the commands spread through botnet is inversely proportional to the size of the network.

3.2 Botnet Detection Methods

To identify whether a network is infected with botnets or to know if a device is part of a botnet, most of the techniques look at network traffic coming out from the devices. Bots from conventional botnets use protocols like HTTP, IRC, SMB and P2P protocols for communication. Advanced botnets that are coded for targeted malicious purposes like targeting other competitors, nations are capable of communicating using protocols like ICMP, FTP (P2P approach) and even UDP. So, to detect botnets it is important to analyze network traffic to identify if there is a botnet infection in our network. Some Botnets detection methods are classified below.

Honeypot detection method: In this method, a honeypot is placed in the network with architectures and operating systems like that of an IoT device. The honeypot may be placed in the network with actual devices. When the malware infects the honeypot mistaking it for an actual device, data is collected from the honeypot like the incoming and outbound traffic to unidentified sites. The network then blocks connections to all those sites from the rest of the devices [21].

Signature based detection: Like viruses and worms, malware and botnets have signatures based on either network traffic or activity done by the infected bots. Botnet infection can be detected as soon as the activities of infected bots are identified by a security tool based on signature. This approach is highly effective only for infections from already known malware for which signatures are already identified. A variation from the standard signature may make this approach ineffective [22].

Anomaly based detection: In anomaly-based approaches, tools or software that can detect network traffic are employed. For example, a spike in the network traffic or increased utilization of device resources can indicate malicious activity in the network.

New malware come with advanced techniques that doesn't require a lot of communication between bots and CNC server. The bots talk to the CNC server rarely to check for updates or commands and are dormant most of the times. There is no spike in network traffic in such cases. Some malware even makes use of tunneling to open covert channels in well-known protocols. Traffic passed through covert channels looks normal when looked from a packet sniffer and difficult to identify any anomalies in it. In such cases where anomaly-based detection approaches fail, machine learning techniques are used to compliment the anomaly-based detection tools. Security researchers and scientists are trying out various efficient algorithms to find out an effective machine learning model that can identify botnets.

Use of models like support vector machine(SVM) and Hidden Markov models(HMM) are increasing to identify malware and botnets. Autocorrelation plots of network traffic was used to identify the patterns in bots [23]. The patterns when applied to actual were able to identify data from bots with very high probability. Neural network and convolutional networks are being employed to increase the accuracy of machine learning models.

4. An Architecture to Collect Data from Iot Devices

4.1 Microservices

Software organizations till now have been using either the traditional monolithic architecture or service-oriented architecture. In monolithic architecture, all the working parts of the software are coded as a single block. Service oriented architectures can be identified by their characteristic of multiple blocks of code with each block corresponding to a service. Even though there might be different parts of code in service-oriented architecture they are collective unit.

Monolithic applications are easier to code but difficult to maintain or upgrade because all the code is in a single block and developers might have to re-write the whole code to add a new feature. Applications that follow service-oriented style are a bit difficult to code compared to monolithic as each block of code can be written by a different team or individual and all of the blocks of code have to work in sync with other blocks of the code. These applications are easy to upgrade or maintain since one block of code must be modified if a specific feature of the application is to be modified. If a feature of the software doesn't work, it can be easily debugged looking at the corresponding code for the service and the blocks of code that communicate with it. Both the monolithic and service-oriented approaches have their own advantages, but they become complex to write and maintain once the size and scope of the software tool increase. Microservice architecture was introduced to tackle this is issue of increase of complexity of writing and maintain a software system proportional to its feature set.

Microservice architecture is one of the most popular buzz word used in today's software industry. It is being associated with multiple domains like cloud, security, infrastructure, software development and devops. Microservices has different meanings depending on the way it is implemented in an organization. Most of common implementations of microservices have some similar characteristics.

Microservice architecture generally refers to loosely coupled systems that work collectively with each other to produce a common output. The individual systems that are part of a microservice pipeline are self-sufficient. They can be installed and deployed individually without any dependence on other tools used in the tool chain. This style gains its popularity from the fact that multiple efficient software tools or block of code can be combined to form another efficient system that works toward a common goal. Once a part of microservice becomes old or obsolete, it can be easily replaced with better alternatives. One of the main problem of microservice is achieving the synchronized state between different software tools or pieces of code used. This might become even harder to achieve depending of choice of software and compatibility between them.

Microservices also have an advantage of ease of setup. To build the whole system, one can start from a single piece in the tool chain and start connecting it to other parts of the framework. The different tools can be on different platforms of their own. For example, a part of the framework can be installed on one operating system and other part can be installed on a different operating system. This is true for n number of pieces used in the framework as long as they can communicate with each other. Microservice architecture also provides a solution for scalability issues. If the software chosen are scalable, then the combination of those tools is scalable as well.

4.2 Architecture

From the previous section, it is evident that there are rapid innovations in malware space. Attackers are developing malware to be innovative and resilient to the traditional approaches. The growth in complexity of IoT ecosystems also have made it difficult to develop a common

solution to tackle all or most of our security concerns. This section proposes an architecture by making use of microservices to collect data from different IoT devices without much overhead. The solution is designed to be applicable to all IoT devices that is running a Linux based operating system or has enough capacity to run a container.

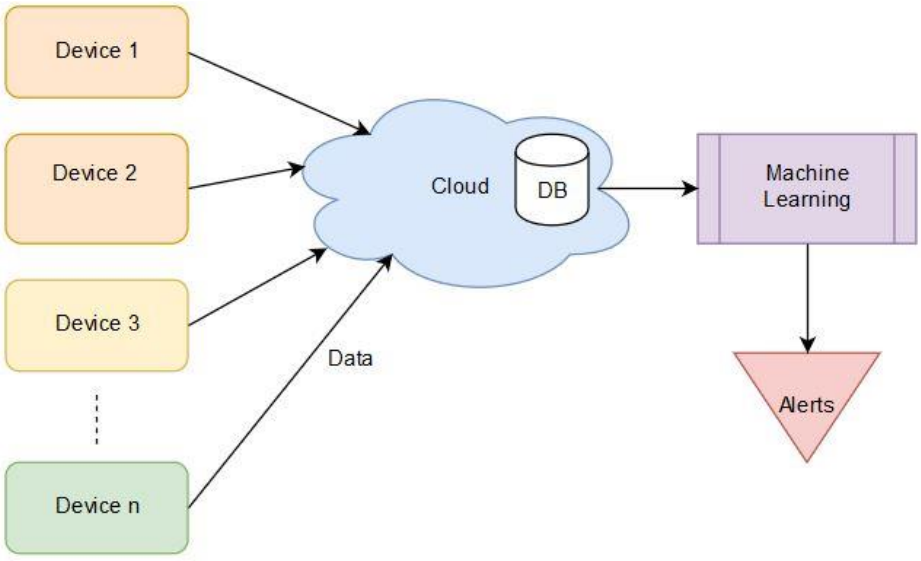


Figure 1.1: Proposed Micro service IoT framework.

The architecture diagram shows different microservices deployed on multiple components.

IoT: The IoT devices are assumed to be powerful enough to run a container on the IoT device or it should be having Linux operating system. Most of the IoT systems today meet this requirement.

Container: For IoT devices without a Linux based operating system, the architecture suggests a containerized approach. This makes is easier for a user to bundle everything in the container and deploy it on multiple devices.

End points: The architecture shows multiple endpoints. This enables the IoT device to send data to multiple devices without congestion. One idea is that IoT devices from different platforms will send data to different endpoints. This would be achieved using microservices and solves the problem of heterogeneity.

Cloud: The architecture shows that the endpoints send data to cloud. This can be a local storage too depending on the organization and infrastructure.

Machine Learning: Once data is collected at a place, it can be analyzed using machine learning to make sense of data. There can be many use cases and applications of data depending

on the data collected. CPU data can be analyzed for power consumption, device health etc. Network traffic data can be analyzed for identifying malicious activities. Alerts can be included in our machine learning framework once the model identifies suspicious data.

The framework collects data from IoT devices and sends it to data storage in cloud by using Kafka microservice and golang programs. The data is then brought in using python and meaningful observations are drawn.

4.3 Setting Up Test Environment

There are limitations to the devices used in the test setup. Kafka has been chosen to stream data from IoT to storage.

IoT devices: The IoT devices in the test environment should be powerful enough to run containers if needed. Raspberry Pi's are the least expensive single board computers closer to our requirements and easily available. So, we choose raspberry pi devices as our IoT device

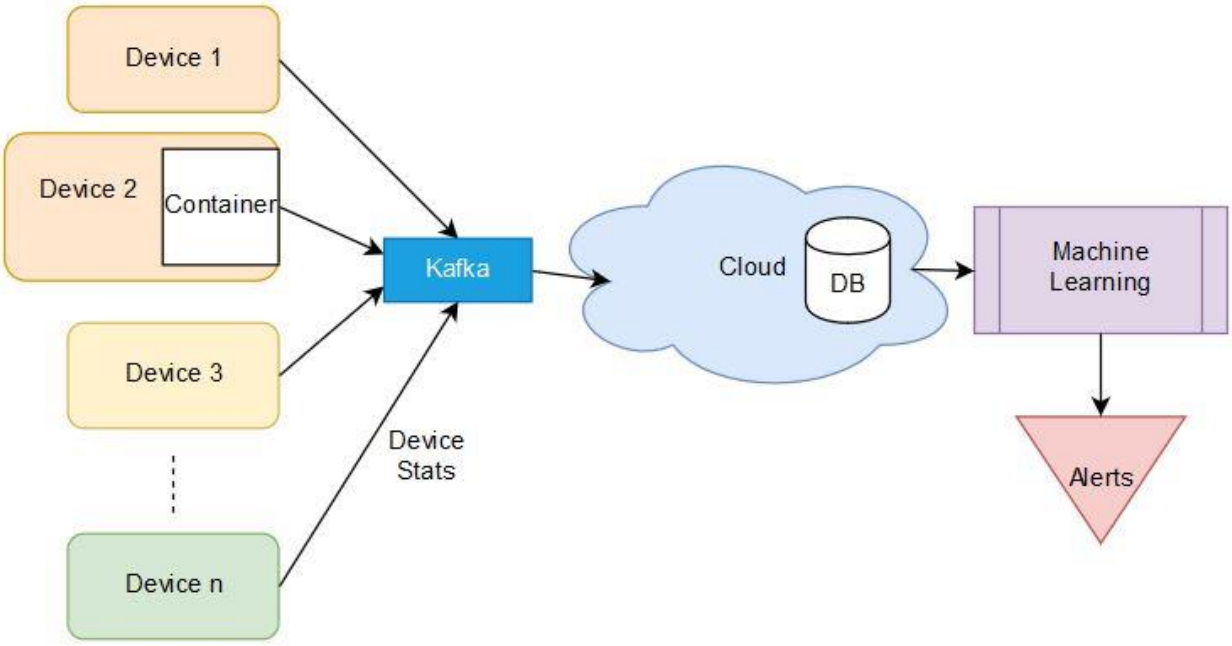


Figure 1.2: Conceptual test environment.

IoT devices considered in the setup:

To prove that the proposed solution works in a heterogeneous environment of IoT devices, we use devices of different architectures and different software is installed on them.

1 Raspberry Pi model 3
1 Raspberry Pi 3b +
1 Raspberry Pi 2
1 Jetson Nvidia TX1 development board.

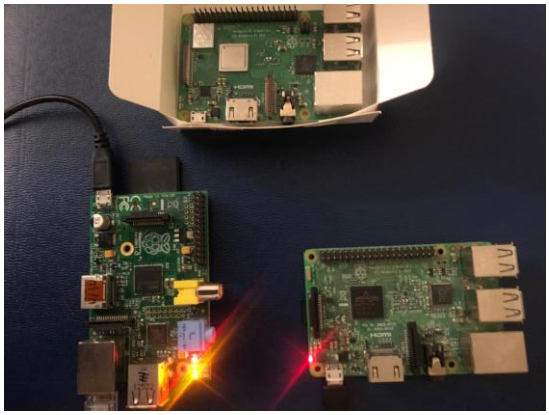


Figure 1.3: Devices in test environment.

The Raspberry Pi 2 is the oldest IoT devices used for experiment. It is installed with Arch Linux ARM, a distribution of Linux for ARM computers. Raspberry Pi model 3 and 3b+ were installed with Hypriot OS 1.8. The Nvidia TX1 board has an ARM variant of ubuntu.

All the boards have different versions of ARM architectures like Nvidia TX1 kit ARM v8 and the Raspberry pi 2 has ARM v6. The Telegraf is compiled differently for each device based on the architecture.

Data collection: Telegraf is used for metric collection. It is tool written in Golang and can be installed on all most all IoT architectures like ARM, PPC, X86 etc.

Containers: Docker is used as the containerization tool. Fedora_harm image was used as the container image

Data transfer: Kafka is being used as the data transfer tool. Kafka is used based on its ability to transfer data from multiple inputs to multiple outputs simultaneously. Kafka is industry standard for data transfer in data pipelines.

Cloud: The setup environment is done in the local network. One of the virtual machine's in the local network is considered as cloud.

Databases: InfluxDB is chosen as the database for the test environment. InfluxDB can scale well with input from thousands of servers and writing up to a million data points per second.

Machine Learning Model: Python libraries are used for machine learning. Data is read from InfluxDB and analysis is performed on it.

Advantages of the architecture:

1. The architecture makes it possible for multiple devices to send data simultaneously taking advantage of Kafka scalability.
2. It is possible to collect data from IoT devices running Linux operating system on multiple architectures as Telgraf can be compiled to work on multiple architectures
3. Data can be collected from IoT devices even without the Linux operating system by using containers
4. The same IoT device can send multiple streams to data like sensor data, device data to different Kafka streams. Data can easily be stored into the database by reading the respective stream. For example, if data from a sensor is pushed to Kafka topic 1 and device data is pushed to Kafka topic 2. Both the data can be fed into multiple data locations just by reading the respective Kafka topic.

5. The architecture can be scaled to multiple devices without any performance bottleneck. This is because of microservices. Each of the tools used like Kafka, InfluxDB are designed for thousands of systems to use simultaneously.

6. The architecture can be upgraded easily if a user decides to change one or some tools from the architecture. This is one of the main advantages of using microservices, that they can be decoupled and used with different tools without any issue. For example, we can replace Kafka with MQTT or other similar protocol.

5. Turn Test Setup Into Botnets

As described in the above sections, there are different types of botnets with various modes for infection. Metasploit and Mirai bot were used to infect the test setup and data was collected from it. Mirai botnet is an example of the latest malware and Metasploit gives a better understanding of a single infected machine.

5.1 Mirai botnet

Anna-Senpai, the author of Mirai malware has open sourced the code of malware. Researchers point that it is a diversion tactic to avoid getting caught. Even though Anna-Senpai was caught and convicted, the open sourced code was widespread among hacker forums and dark web. The code has undergone several mutations and is widely circulated on the dark web. Some of the mutated versions are more difficult to mitigate than the original Mirai malware.

Hoho Mirai, a slight mutation of the original mirai malware was used for the experiment. Mirai malware comes with a CNC and reporter domains. For this experiment, only CNC was used, and the bots were loaded using loader scripts which are part of the malware code. The malware was

setup on VPS with Centos 7 operating system. Once the malware is compiled, the CNC can be opened using a telnet session to the Centos VPS on the port which runs CNC.

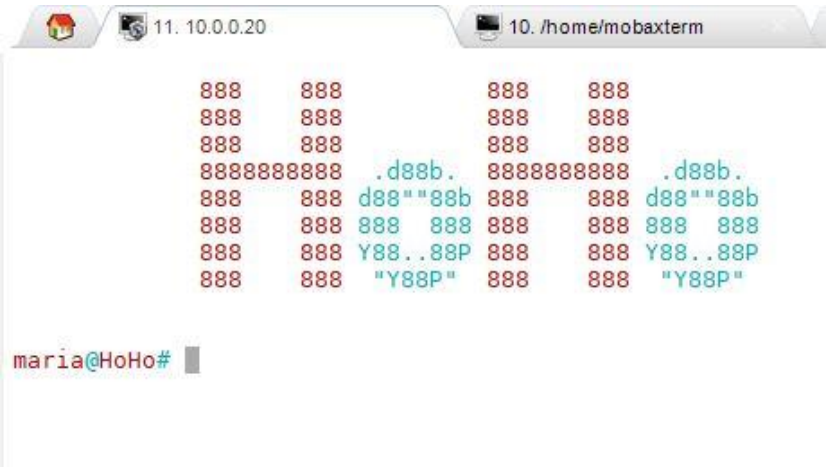


Figure 1.4: HoHo Mirai CNC.

The telnet session in Figure 1.3 shows the mirai CNC. Various attack options available from the CNC can be viewed by typing '?' key in the prompt.

Figure 1.5: HoHo Mirai attack options.

Once a attack command is issued all the bots in the botnet perform that command. There are many ways to load bots into the botnet like running a scan of IP's on the local network and trying to brute force the SSH or telnet logins with most used list of passwords or running a payload on a victim system or using of list of known vulnerable devices. Loading a list of vulnerable devices with usernames and passwords is the easiest way to increase the bot count in the botnet. IP address of the raspberry Pi's and their passwords were saved in a text file and run against loader.

The saved file with login credentials of the devices in experiment is run against the loader and those devices become part of the botnet. Commands to perform attacks can be given from the CNC and all the devices that are part of the botnet will perform the attack.

The attack commands have a general syntax of "ATTACK_NAME IP TIME_IN_SEC". In the experiment a xmas attack was initiated on one of the local servers and data was collected in influx. Xmas attack (Christmas attack) is a type of Denial of Service attack that advantage against stateless firewalls. While the bot in the mirai botnet is carrying out the attack, data was collected using the architecture mentioned in previous section.

5.2 Metasploit

Metasploit is an open source software currently maintained by Rapid7, used for pen testing and exploitation of remote or local machines. Most of the vulnerabilities disclosed online are present as exploits in Metasploit. A user can choose an exploit and a target machine and check if the target machine is vulnerable to that attack. There are over a thousand exploits to test against a machine in Metasploit.

For the experiment, Metasploit was run in a docker container exposing only required ports for executing the exploit. Since Raspberry Pi's in setup have Debian based operating system. It is easy to exploit them using exploits written for Debian systems. Web delivery exploit is used to infect the Raspberry Pi's. The raspberry has to access the URL where the exploit is being run. Metasploit makes it easy to configure and deploy many exploits just from the command line.

Figure 1.6: Open Metasploit in command line.

Once the Raspberry Pi's access the web server that is running the web exploit, a meterpreter shell is open from Raspberry device to the docker container where Metasploit is running.

The Raspberry Pi behaves like a bot because of the meterpreter shell. An attacker can run various commands, upload and download files on Pi. Even the shell on Raspberry Pi's operating system can be accessed via the meterpreter shell. DDOS can be performed via the meterpreter shell. Data is collected regarding the network statics using the input net module of telegraf in the database via Kafka.

6. Botnet Detection Experimentation Results

6.1 Data Collection

Time series data is collected for this experiment using a microservice architecture. The telegraf plugin pushes data to a given kafka topic. A golang program has been written to check the kafka topic for new messages and pushes the data to a database in InfluxDB. The data is obtained from InfluxDB using python.

General machine learning algorithms cannot be applied on time series data as the values of data considered in feature set will always be increasing. So novel techniques are used for this experiment to identify the anomalies in time series data.

The input plugin "net" of telegraf [24] is used alongside the out of the box configured modules. Data from the device on stats like CPU usage, CPU idle, RAM usage, number of processes running, number of threads, Sys uptime etc. from the out of the box configuration module are collected in database. After adding the net module data of more fields like bytes_sent, bytes_recv, packets_sent, packets_recv, err_in, err_out, drop_in, drop_out are collected. So, the initial size of features is high. It is observed that all the features except bytes_sent, bytes_recv are redundant. Hence the final features are bytes_recv and bytes_sent.

Data is pushed from Raspberry Pi device to database once every 5 seconds. This ensures a high number of data points to run the model. The 5 second interval is also useful to identify a suspicious activity within the least time possible.

The data from just the net input plugin has close to 90 columns and most the columns have empty data.

```
Out[64]: name                      object
         time                      object
         bytes_recv               float64
         bytes_sent               float64
         drop_in                  float64
         drop_out                 float64
         err_in                   float64
         err_out                  float64
         host                      object
         icmp_inaddrmaskreps      float64
         icmp_inaddrmasks         float64
         icmp_incsumerrors        float64
         icmp_indestunreachs      float64
         icmp_inechoreps          float64
         icmp_inechos             float64
         icmp_inerrors            float64
         icmp_inmsgs              float64
```

Figure 1.7: Some columns from net plugin.

A large sample of data is collected for data analysis. Data from devices before infection are collected for 15 days, one week, 3 days and few hours. Data for short intervals like 30 min was also collected under multiple scenarios. Data is performed before using it for data analysis. First, the inconsistencies in data are removed by eliminating rows with empty columns. Data is cleaned by removing all the empty and irrelevant columns. The time stamps are standardized by converting them into nanoseconds. After cleaning, we only have two columns bytes_recv and bytes_sent.

6.2 Using Collected Data

Data regarding the device was collected using the Telegraf plugin. A part of data specifically data regarding network statistics was used for machine learning to determine if the device is performing any malicious activity or not. A large of other collected data is leftout. Dashboards are made using the remaining data.

6.2.1 Data Analysis For Botnet Detection

General analysis: Once data is collected, it is prepared by removing the irrelevant columns and only 2 features bytes_recv and bytes_sent are kept. The rest of the features which are removed, provided no value to the model for analysis.

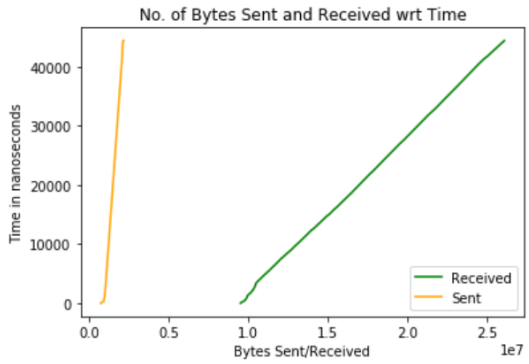
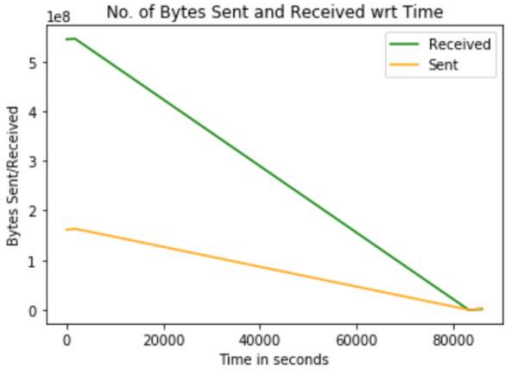


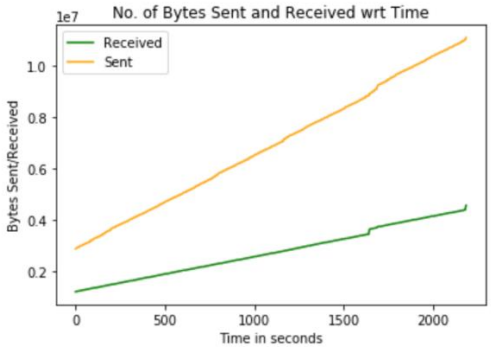
Figure 1.8: Bytes sent/ received with respect to time.

The behavior between bytes_sent and bytes_recv with respect to time is shown in figure 1.10. General analysis like this can be used to identify the anomalies in case of simple cases but when there is mixed data it may not be simple.

Data was collected from uninfected device, infected device when launching a DOS attack and an infected raspberry device carrying its normal functions along with some attacks instructed by the bot master or CNC. To make clear and distinct observations, data of 30 min was collected for the three different scenarios mentioned. 30 min data from an uninfected device, 30 min of data from the infected raspberry device when it is executing a DOS attack, 30 min of data from infected raspberry device where it's executes commands from the botmaster only for 15 minutes timespan and behaves like a normal device for the rest of the time is collected. The data was analyzed without machine learning models using simple slope functions and graphs.



(a)



(b)

Figure 1.9: Bytes sent/ received with respect to time from (a) uninfected Raspberry Pie (b) an infected Raspberry device while its participating in a DOS attack.

From the Figure 1.11, it can be seen that there is a correlation between bytes_sent and bytes_recv. The differences between the data from a malicious and good device are clearly distinguishable. In case of an infected Raspberry device which is part of a Distributed Denial of Service (DDOS) attack, the data sent is high compared to data received and that can be observed from the figure.

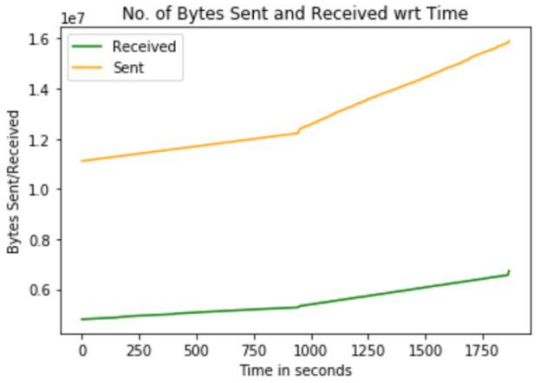


Figure 1.10: Bytes sent/ received with respect to time from an infected Raspberry device that executes some commands given by botmaster for few min while the total duration for data collected is 30 min.

There is difference in slope when the device is behaving normally and when the device is carrying out instructions given by the CNC. The CNC started sending instructions like file uploads, shell commands and some ping commands to the Raspberry device after 900 seconds and it can be clearly seen in the figure.

Comparison of data sent and received using simple math functions can be helpful to differentiate between data from an infected device and non-infected device in simple cases. This was possible in the observations because the Raspberry Pi was idle most of the time and only sending the device data to Kafka. That may not be possible in real life scenarios. In real life scenarios the Raspberry Pi may be connected to different sensors and may be sending data continuously. There might be a need to alert only when the device is infected and not when one of the sensors of Pi fail. The above general analysis may also fail when there is a short burst of data sent or received statistics because of execution of small command on the Pi by the CNC. For all the cases where general analysis may fail because of its limitations, machine learning provides the solution.

SVM models are supervised machine learning models used for classification. Instead of giving probability, it indicates if a point belongs to a particular class or not. SVM can be used with different types of kernels like rbf, polynomial etc. based on the types of data. One Class SVM is a special case of SVM that has only one class. One class SVM given by Scholkopf [25] is a good way to identify if the incoming data is normal or outlier. In one class SVM or one class classification(OOC) data is trained only on one set of data. In this experiment the model is trained with only data from a normal device. After the training is completed, model is tested with data from infected data. The model will then indicate if the new data is like previous class (with data from good device) or not. This can help in identifying to what type of device (infected or normal) the data belongs to.

The one class SVM model converts the data into a series of 1's and -1's where 1 represents the next observation point to regular or normal data and -1 represents outlier or abnormal data.

```
det.fit_predict(bytes_df_newdata_train_rec)
```

```
array([-1,  1, -1,  1,  1,  1,  1, -1, -1, -1, -1,  1,  1,  1,  1, -1, -1,
        1,  1, -1, -1, -1,  1,  1, -1, -1,  1,  1,  1, -1, -1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1,  1,  1,
       -1, -1, -1, -1, -1, -1, -1, -1,  1,  1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  1,  1, -1, -1,
       -1, -1, -1, -1, -1, -1,  1,  1, -1, -1,  1,  1,  1,  1, -1, -1, -1,
       -1,  1,  1, -1, -1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1,  1,  1,
        1,  1, -1, -1, -1, -1, -1, -1,  1,  1, -1, -1, -1, -1, -1, -1, -1,
       -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1,  1,  1, -1, -1,
        1,  1,  1,  1, -1, -1,  1,  1,  1,  1, -1, -1,  1,  1,  1,  1,  1,
```

Figure 1.11: Results of one class SVM model.

To make sense of the above data, a simple technique of longest sequence is used. The resultant array is looked upon for the longest sequence of -1's. For the sake of simplicity, the longest sequence of 1's is called 'score' in the scope of this experiment.

The model is run against data collected from Raspberry Pi under different conditions. Some of the results from the experiment are recorded.
When the model is run against data collected from infected machine which is running a meterpreter shell, it gives a high score (value of longest sequence of -1's) whereas when model runs on data from a normal device, the scores are low.

```
#bad data
arr = det.fit_predict(bytes_df_baddata_train_rec)
print(longest_substring(arr))
```

```
44
```

Figure 1.12: Score of model from data during a meterpreter session.

The meterpreter session was active for few hours and many instructions were carried out like uploading, downloading files, pinging few known websites and some basic Linux commands like ls, cat, grep, netstat etc.

Data was collected for various time periods of days and weeks. For the model to alert as soon as it detects the malicious activity, the time period on which data was trained should be shorter. After testing out the model of data for 5min, 10min and other time periods, it is found that the model works better for data of time duration 30 min and above. Model was run against data of 30 min duration from different devices in the experiment and the results are tabularized.

Table 1 Examples for illustrating attacks

Data from	Score
Uninfected Device	6
Infected Device performing a DDOS attack	15
Infected device acting normally for some time and performing malicious activity for the rest of time span	13

The score 13 is relatively higher than the score of data from an un infected device and is close to score of data when the Raspberry is launching a DOS attack. The threshold score for our use case in the experiment was set to be 10. So, any data that has a score less than 10 is from an uninfected device and data with score more than 10 is from an infected device. Hence the model that was constructed using novel detection techniques for time series data and calculating the scores is an efficient way to distinguish whether an IoT device is performing any suspicious activity. The model can be programmed to run in real time and can alert as soon as the score is greater than a particular threshold depending on how the devices are being used.

6.2.2 Making Dashboards

Using the Telegraf plugin, we collected device statistics from Raspberry Pi for every 5 seconds. We have data on multiple stats of hardware data like CPU usage, RAM usage, Number of processes, number of threads, disk usage, network statistics, system up time etc. All the data is stored in InfluxDB with their respective measurement names.

InfluxDB stores data in the time series format. i.e. timestamp can be unique key of a row in the database. The data gets written to the database every 5 seconds and that frequency can be modified to even lesser value. This can help in finding the status of the device in real time as there is live data being pumped into the database.

```
> show measurements
name: measurements
name
----
cpu
disk
diskio
kernel
mem
net
processes
swap
system
> select * from disk limit 5
name: disk
time                device    free        fstype host        inodes_free inodes_total inodes_used mode path  total       used       used_pe
rcent
----                ------    ----        ------ ----        ----------- ------------ ----------- ---- ----  -----       ----       -------
-----
1542500440000000000 mmcblk0p1 30363648    vfat   black-pearl 0           0            0           rw   /boot 66959360    36595712   54.6536
16760972625
1542500440000000000 root      27510779904 ext4   black-pearl 7079096     7120960      41864       rw   /     30004998144 1197535232 4.17138
8067627488
1542500445000000000 mmcblk0p1 30363648    vfat   black-pearl 0           0            0           rw   /boot 66959360    36595712   54.6536
16760972625
1542500445000000000 root      27510779904 ext4   black-pearl 7079096     7120960      41864       rw   /     30004998144 1197535232 4.17138
8067627488
1542500450000000000 mmcblk0p1 30363648    vfat   black-pearl 0           0            0           rw   /boot 66959360    36595712   54.6536
16760972625
>
```

Figure 1.13: Data in InfluxDB.

The data in InfluxDB is difficult to make sense of, given a large number of colums. One way to make better sense of the data is to visualize the data. Grafana tool is used to change the data in InfluxDB to graphs. Grafana is a free open source tool that can take inputs from multiple databases of different formats and convert them to graphs. In Grafana, a dashboard is a collection of graphs and other panels that help visualize the data. Since there is lot of data collected for this experiment and it is possible to push live data to the database by using the micro service architecture built for this experiment, it is possible to build live monitoring dashboards for the IoT environment.

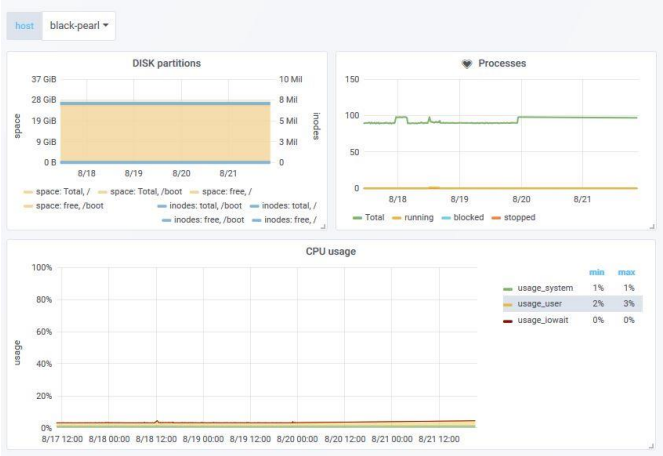


Figure 1.14: Part of Grafana dashboard.

Making live dashboards can also help us detect any anomalies in the behavior of devices in the environment in some cases. The graphs in dashboards can be configured to alert the admins. For example, a disk usage graph can be configured to send alerts when the disk space usage exceeds 90%. When the alert is sent, the person responsible for maintaining the IoT device can take necessary actions. In this way all the data generated by this experiment can be used. Some data is used for data analysis to detect malicious activity of IoT devices and the rest unused data can be used to generate dashboards and monitor the IoT environment in real time.

7. Conclusion

7.1 Project Summary

In this project, we talked about the reasons for vulnerabilities of IoT devices. A new architecture was proposed making use of the current trend of microservices architecture to solve the problem of complexity due to heterogeneity and lack of monitoring. The microservice architecture proposed can collect device data from multiple IoT devices from different vendors and storing it in a database. The experiment was conducted on a heterogeneous cluster of 3 models of Raspberry Pi's and a one NVIDIA jetson tx1 module. Telegraf was installed on all the devices and data has been collected from IoT devices and used for making dashboards and machine learning. Some of the Raspberry Pi's in the cluster were turned into bots by infecting them mirai malware and using Metasploit to gain access of the devices.

Data was being collected before and after the infection of devices. A machine model was created using Novel detection methods. The machine learning model gives a score for the data that is being fed. The score of the model was high when data from an IoT device doing malicious or suspicious activity was fed into the model. The model score was low when the data used is from an uninfected device. The proposed solution can differentiate between an uninfected device and a device doing suspicious activity with the help of the model that can differentiate between data from a normal uninfected IoT device and data from a device infected with malware. Any suspicious activity can be seen from the dashboards built using the collected data.

The Proposed solution demonstrates that it can be applied to IoT devices of different architectures as long as telegraf can be compiled and installed. Since this follows microservice architecture, the telegraf part can be opted out and kafka can be used directly for communication. The proposed solution is also scalable to at least thousands of devices because of the software tools used.

7.2 Future Research directions

The experiment described in this project is a simple proof of concept of applying microservices architecture to overcome heterogeneous complexity and to use device statistical data for determining malicious activity. The data used for the experiment uses bytes sent and received by an IoT device as a whole without any filtering between the types of data. Note, however, Telegraf net plugin used in the experiment is powerful enough to differentiate bytes based on the type like UDP, TCP and ICMP packets. Further research can be done on how these data will change based on the type of malware infected. For example, the proposed model may be used to differentiate a malware that spreads using IRC from a bot that spreads using UDP by collecting and observing UDP data patterns.

References

[1] E. Ronen, A. Shamir, A.-O. Weingarten, and C. Oflynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[2] J. R. Stachel, E. Sejdic, A. Ogirala, and M. H. Mickle, "The impact of the internet of Things on implanted medical devices including pacemakers, and ICDs," *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2013.

[3] Hung, M. (2017). *Leading the IoT*. [ebook] Gartner. Available at: https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf [Accessed 7 Nov. 2018].

[4] A. Gregg, "Oracle agrees to buy Arlington energy data firm Opower for $532 million," *The Washington Post*, 02-May-2016. [Online]. Available: https://www.washingtonpost.com/business/economy/oracle-agrees-to-buy-arlington-energy-data-firm-opower-for-532-million/2016/05/02/83739416-107f-11e6-93ae-50921721165d_story.html. [Accessed: 07-Nov-2018].

[5] F. Lardinois, “Microsoft acquires Italian IoT platform Solair,” *TechCrunch*, 03-May-2016. [Online]. Available: https://techcrunch.com/2016/05/03/microsoft-acquires-italian-iot-company-solair. [Accessed: 07-Nov-2018].

[6] Z. Zhang, M. C. Y. Cho, C. Wang, C. Hsu, C. Chen and S. Shieh, "IoT Security: Ongoing Challenges and Research Opportunities," 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Matsue, 2014, pp. 230-234. doi: 10.1109/SOCA.2014.58

[7] A. Greenberg, “‘Devil's Ivy’ Vulnerability Could Hit Millions of IoT Devices,” *Wired*, 18-Jul-2017. [Online]. Available: https://www.wired.com/story/devils-ivy-iot-vulnerability/. [Accessed: 07-Nov-2018].

[8] A. Robertson, “California just became the first state with an Internet of Things cybersecurity law,” *The Verge*, 28-Sep-2018. [Online]. Available: https://www.theverge.com/2018/9/28/17874768/california-iot-smart-device-cybersecurity-bill-sb-327-signed-law. [Accessed: 07-Nov-2018].

[9] “The search engine for the Internet of Things,” *Shodan*. [Online]. Available: https://www.shodan.io/. [Accessed: 07-Nov-2018].

[10] “Google Hacking Database (GHDB),” *Google Hacking Database, GHDB, Google Dorks*. [Online]. Available: https://www.exploit-db.com/google-hacking-database/. [Accessed: 07-Nov-2018].

[11] “Every Routable IP Project,” *ERIPP*. [Online]. Available: http://www.eripp.com/. [Accessed: 07-Nov-2018].

[12] M. Patton, E. Gross, R. Chinn, S. Forbis, L. Walker, and H. Chen, “Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things (IoT),” *2014 IEEE Joint Intelligence and Security Informatics Conference*, 2014.

[13] “ATMA.ES Fighting malware | Seguridad en la red,” *ATMA.ES Fighting malware | Seguridad en la red*. [Online]. Available: http://www.atma.es/. [Accessed: 07-Nov-2018].

[14] NJCCIC. (2018). *Aidra Botnet*. [online] Available at: https://www.cyber.nj.gov/threat-profiles/botnet-variants/aidra-botnet [Accessed 7 Nov. 2018].

[15] Hayashi, K. (2013). *Linux.Darlloz*. [online] www.symantec.com. Available at: https://www.symantec.com/security-center/writeup/2013-112710-1612-99 [Accessed 7 Nov. 2018].

[16] Jgamblin, “jgamblin/Mirai-Source-Code,” *GitHub*. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code/blob/master/ForumPost.md. [Accessed: 07-Nov-2018].

[17] “Krebs on Security,” *Brian Krebs*. [Online]. Available: https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/. [Accessed: 07-Nov-2018].

[18] C. Zheng, C. Xiao, and Y. Jia, “IoT Malware Evolves to Harvest Bots by Exploiting a Zero-day Home Router Vulnerability,” *Palo Alto Networks Blog*, 11-Jan-2018. [Online]. Available: https://researchcenter.paloaltonetworks.com/2018/01/unit42-iot-malware-evolves-harvest-bots-exploiting-zero-day-home-router-vulnerability/. [Accessed: 07-Nov-2018].

[19] C. Cimpanu, “‘Hide and Seek’ Becomes First IoT Botnet Capable of Surviving Device Reboots,” *BleepingComputer*, 08-May-2018. [Online]. Available: https://www.bleepingcomputer.com/news/security/hide-and-seek-becomes-first-iot-botnet-capable-of-surviving-device-reboots/. [Accessed: 07-Nov-2018].

[20] G. Vormayr, T. Zseby and J. Fabini, "Botnet Communication Patterns," in IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2768-2796, Fourthquarter 2017. doi: 10.1109/COMST.2017.2749442

[21] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, “IoTPOT: A Novel Honeypot for Revealing Current IoT Threats,” *Journal of Information Processing*, vol. 24, no. 3, pp. 522–533, 2016.

[22] N. S. Raghava, D. Sahgal and S. Chandna, "Classification of Botnet Detection Based on Botnet Architechture," 2012 International Conference on Communication Systems and Network Technologies, Rajkot, 2012, pp. 569-572.doi: 10.1109/CSNT.2012.128

[23] P. Nagarajan, F. D. Troia, T. H. Austin, and M. Stamp, “Autocorrelation Analysis of Financial Botnet Traffic,” *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018.

[24] Influxdata, “influxdata/telegraf,” *GitHub*. [Online]. Available: https://github.com/influxdata/telegraf/blob/master/plugins/inputs/net/NET_README.md. [Accessed: 07-Nov-2018].

[25] B. Schlkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, J. Platt, "Support Vector Method for Novelty Detection", Advances in Neural Information Processing Systems 12, pp. 526-532, 1999.