International Journal of Hybrid Information Technology
Vol.13, No.2 (2020), pp.1-16
http://dx.doi.org/10.21742/ijhit.2020.13.2.01

Monitoring and Detection of Security Events through IoT Device Identification Using Application Layer Protocols

Ammad Khan[1], Yongle Chen[2], Waqas Ahmad[3], Kamran Javed[4], M. Bilal Zia[5,] and Arooj Khan[6]

[1,3,5]Master's Student,Dpt. of Information and Computer,Taiyuan University,China
[2]Asst. Professor, Dpt. of Information and Computer, Taiyuan University,China
[4]PhD Scholar,Dpt. of IoT Engineering, Hohai University, Nanjing, China
[6] Master's Student, Dpt. of Computer Science, Khawaja Fareed University, Pakistan
[1]ammadkhalid026@gmail.com, [2]chenyongle@tyut.edu.cn,
[3]waqaswebmaster@gmail.com, [4]kamran.javed86@hotmail.com,
[5]zia.bilal6@yahoo.com, [6]ardeel56@gmail.com

Abstract

Internet of Things network is based on the distributed infrastructure as large of number of devices connected to the network makes the network an ultra-dense network. The profound devices are becoming capable of connecting to the other devices operating on different networks nature and different architecture thus giving birth to the heterogenic nature of the networks. In such an environment where incident responders face challenges postured by the event that occurred from IoT device networks becomes difficult to gather, analyze and examine its impending traces. This study proposed a contrivance to fetch and provide the information of the IoT devices connected to a certain network using protocols of application layers and associated open ports to the investigators and incident responders. This will help detect and identify the IoT devices connected to the network that will to a significant certainty aided the work of investigators. For this purpose, a tool will be presented through series of experiments and algorithmic development. The results of the experiment show that the proposed tool effectively identified the IoT devices associated with open ports and also classification of the IoT and non-IoT devices is achieved.

Keywords: *Banner grabbing, IoT, Device discovery tools, Fingerprinting*

1. Introduction

When a cyber-attack occurs many details can be considered as the foundation of evidence because in the present cyber-security techniques using or identifying the threats just from scanning or recognizing the physical evidence is not remarkable thus the importance of digital evidence cannot be neglected. The collection of digital evidence is important because of the digitized nature of the data. There are many examples where digital evidence makes a remarkable contribution to the investigative authorities to identify the attackers. Studying these

Article history:
Received (November 27, 2019), Review Result (February 14, 2020), Accepted (March 22, 2020)

examples this fact can be established that technology makes a significant contribution in modern law enforcement and investigation [1]. This example refers to the use of social media networking to collect evidence against the alleged. This further intimates that the escalation of use of technology in social media and routine matters can elevate the incident response upon its occurrence.

The involvement of technology in the investigation has its vital importance, however, the point of pondering is that what upcoming technologies will also be capable of aiding the investigative procedures? The answer to this question is the Internet of Things technology, the devices compatible with IoT act as the bridge of communication between the factual and digital world. The bridging of the actual and digital world may be useful to retrieve information aiding investigators. The innovations of automated houses, driver-less cars, unmanned airplanes and autonomous farms and industries are examples of IoT-connected devices with sensors. These sensors can be simple and complex based on the application i.e. sensors working in an unmanned airplane or a sensor acting as an electric light switch [2]. IoT is said to work on millimeter wave and will have high bandwidth due to which it is expected that by the next year 2020 more than 20 billion devices will be able to connect each other through heterogeneous networks [3].

The purpose of this study is to find out that what type of information and how this information can be retrieved from the devices working as the elements of IoT and also how it can aid the investigator. The IoT devices have to be discovered and assessed on the network of investigators for this purpose tool will be developed to discover the IoT devices, to achieve this objective the IoT devices will be studied. The discovery of the devices on the network of the investigator is achieved to retrieve potential information which thus can be analyzed. To retrieve the information from the devices controlled by the internet of things and also further analyzed to see what other information can be retrieved not only from renowned data centers but also apart from them which will be the main concern and focus of this research. The potential information cannot only be found from the servers, web header and fingerprinting operating systems but also can be retrieved from the user’s cell phone storage.

This study will proceed further with a literature review and other related and previous works done by the researchers to discover the IoT devices and retrieve the information. This study will also provide the challenges in discovering IoT devices as well as put forth the design solution and implementation of the provided technique and solution. The second part of the study will emphasize the policy-based identification of IoT devices. In the end, consolidated results and analysis supported with a conclusion and suggestions for other researchers for future work will be provided.

2. Related work

Internet of things is an emerging and newly introduced area which is still finding its implementations in different areas which are already established due to this reason while searching for the studies which have already addressed the issue are not much probable and merely few articles have been written in the area of device discovery, annotating devices and response on the occurrence of the event in IoT. In this section, first, the techniques will be discussed already used to annotate and device discovery on the application layer based protocol, then this study will further discussed some protocols of application layers available and already been developed for the accomplishment of the said purpose and then this study will present the motivation for using the proposed application layer based protocol and traffic analysis technique for annotating the devices and their discovery for the investigators to prevent the potential attacks and investigating the traffic.

2.1. Fingerprinting

Fingerprinting refers to the technique of identifying mobile and other devices on the network. This technique works by matching the snapshots of the user's device when it is clicked. Fingerprinting technique when used to take snapshot can fetch the information of mobile devices such as Internet Protocol, user agent, the operating system of the device and web or store application. It can also capture the screen size and screen resolution and other deep details when parameters are altered to do so especially with the application layer protocols. For example, if an advertisement on the web page or application is clicked by the user, and this advert took the user to install any application, a snapshot is taken, and afterward when the application is installed and opened by the user a similar snapshot is taken again and then matched with the earlier snapshot. In this case, if the device founds a similar snapshot in its database which was stored when the first snapshot was taken then the advertiser gets attributed to installing the application.

Shamsi, Cline, and Loguinov discussed the fingerprinting technique for operating systems. This study focused on coping up with random distortion in the network along with the scanning of the user's features in the course of internet-scale SYN. The classification techniques discussed in this study only comes true when it is assumed that parameters of the reported network are termed as a-priori – the possibility of the modification in the default value of TCP/IP header by the user along with occurrence of packet loss, how much the operating system used is popular and how the network is delay is distributed among the packets. This study also addresses the issue which was not properly addressed in the previous studies, that for public internet how a realistic version of mentioned parameters can be obtained to analyze the particular network by customizing those parameters. This study also proposed a technique called *Faulds* which was a non-parametric Expectation-Maximization estimator. This study claimed that this technique is for the involvement of distributions that are not known, in the single-probe operating system fingerprinting and also emphasizes that robustness to noise of the proposed EM estimator is significantly higher than the studies in the previous works. However, this study did not particulate the types of the devices on which fingerprinting was carried out that how was the compatibility of the devices with IoT and also applied fingerprinting technique to get the operating system information and it was not in the scope of study that how fingerprinting can grab the other details of the devices like vendors, version of the application and IP addresses [4].

Caballero et al. in his research also used the fingerprinting technique to generate automatic and accurate fingerprints for the classification of the software pieces. This study summarized that, with the aim of the classification of the pieces of software into classes, the fingerprint technique can be used which is fundamentally and principally composed of the set of inquiries and the responses on the inquiries, followed by the classification function that can be implemented to the responses. This study presented the approach consists of 3 phases for automatic fingerprint generation that are candidate query explorations, machine learning and the testing phase. In the first phase, the candidate query set is presented which can yield dissimilar responses from the hosts which belong to, unlike classes. This study used machine learning techniques so that it can categorize the valid queries and machine learning techniques can also be used to learn the adequate classification function. Classification of each software piece can be achieved easily by automatic generation of accurate fingerprints and it does not exploit the search space while querying exploration which is beneficial for other queries that are awaiting discovery. However, this study did not discuss details about the queries generated from the IoT devices and also how fingerprinting technique can help the investigators to find the vendors, IP address and applications in the devices which are on IoT networks [5].

Shamsi et al. in his study indicates that traditional operating system fingerprinting tools are not effectively used for analyzing a large amount of traffic and attacker interventions and their probes. This study also proposed that if remote servers can produce the feature vector with a single SYN packet then above mentioned problem can be resolved which was not doable with the help of traditional TCP/IP fingerprinting tools for example nmap. These traditional models were not able to classify the difficult problems because of instability and unpredictability of the features and also not possible because of very limited information data encapsulation in there. This study proposed the stochastic theory of OS fingerprinting using singly packet SYN. However, this study did not indicate that how the device features can be grabbed by the investigators on the application layer and what details can be fetched to investigate the traffic. Moreover, using Single-packet SYN a large amount of data flow cannot be monitored while the NGWN consists of 5G technology which can transfer up to 100gbps, and the number of devices will be exponentially increased in the 5G networks. This technique of producing vector features with single packet SYN is good for lower data rates and networks with a not dense population of devices [6].

From the above studies, it is clear that to fingerprint an IoT device requires two main things, queries and responses, queries generated from devices and responses from the network servers. To fingerprint and grab the device the conversational information between device and network is mandatory. The other requirement to fingerprint the IoT device is the class label which is also known as category or target, is the class to which the IoT device belongs.

2.2. Deficiencies of fingerprinting

The main feature of securing the device and investigating the information flow through the device is the time. Fingerprinting technique works well when the period between clicking the link and the installation of the application is less i.e. within few hours. The main flow of work done by the fingerprinting technique can be summarized as:

- A web fingerprint is generated by the network which consists of transitory and short-lived mobile data that includes IP information, OS information, Version details, Vendor details and more on the click of the link by the user.
- A followed up and subsequent fingerprint is again generated by the network when the application is installed and opened for the first time.
- The user is deep linked with the application servers if the fingerprints matched.

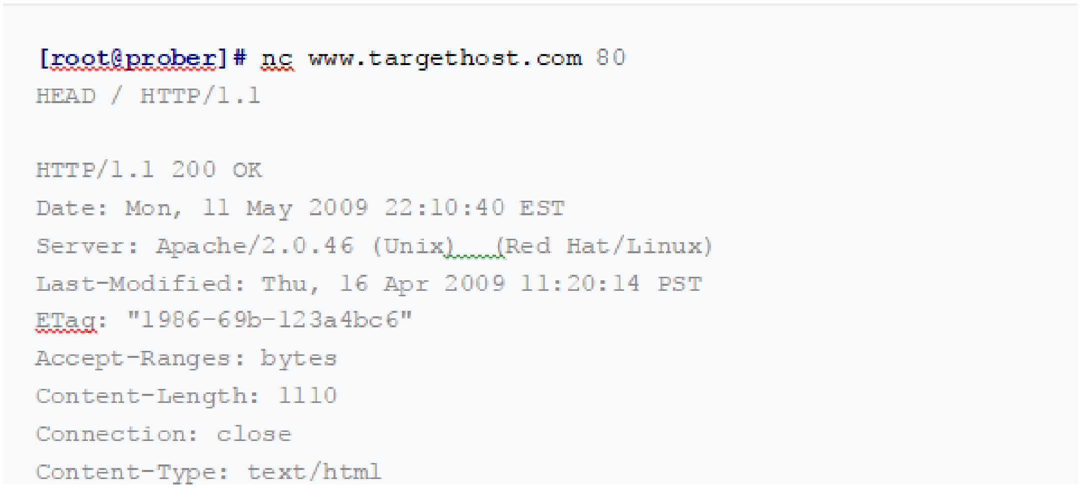
If analyzed deeply there can two problems which may arise, one is time span, if the time span is becoming high enough like 3, 4 hours the finger print will become useless because the decomposition of fingerprints starts as soon as they are taken. The second problem arises when the fingerprints generated from different devices on the same network are highly alike. These devices get paired to a completely different device on the network because of the similarity in the fingerprints. Though this happens with 30 – 40 percent of devices even then this percentage is relatively when considered in ultra-dense networks of 5G. However, the attribution networks claim that there will be an advanced filtering technique for fingerprinting to avoid wrong pairing of devices even though the whole inherited process is week so it will become impossible to extract gold out of the dirt.

Moreover and more importantly, fingerprinting requires training data as the classification model relies on the training data to input in the class labels. To achieve high precision and coverage it necessitates that the training data must be large for a large number of classes. As there is very limited training data available for the IoT devices and the number of IoT devices is large as compared to the classes of the operating system. So it is not feasible to collect the training data for the IoT devices manually, because the number of IoT devices will increase exponentially in the ultra-dense 5G networks. So collecting training data for the devices in such a network will become a difficult task.

2.3. Banner grabbing

The banner grabbing technique is used to fetch textual information about mobile and computer devices. Using banner grabbing can also provide information about the network and its open ports on which the device is connected and sharing information. This technique can be used to get the list of devices connected to the particular network to make an inventory or to check the network resources and services being used by them. This technique can also be used to get information about the hosts, the protocol usage, open ports, operating system running on the host and application version and details. Banner grabbing is used with the protocols of the application layer to extract the information about the application running on the device and their version along with the operating system details, these protocols can be HTTPS, FTP, SMTP on port 80, 20 and 25 respectively. Nmap, telnet, zmap and netcat are some tools to perform the banner grabbing of the devices generally from the Linux environment.

Using any of the tools, the connection to the webserver can be established and sending HTTP request afterward, to which web server will respond with the following details shown in [Figure 1].



```
[root@prober]# nc www.targethost.com 80
HEAD / HTTP/1.1

HTTP/1.1 200 OK
Date: Mon, 11 May 2009 22:10:40 EST
Server: Apache/2.0.46 (Unix)  (Red Hat/Linux)
Last-Modified: Thu, 16 Apr 2009 11:20:14 PST
ETag: "1986-69b-123a4bc6"
Accept-Ranges: bytes
Content-Length: 1110
Connection: close
Content-Type: text/html
```

Figure 1. Information from Webserver

This information from the web server response can be used by administrators to log the system and to know which services are running. However, this information, in the same way, can be retrieved by the intruder from which one can extract the information about the open port to probe an attack [7].

Banner grabbing is another technique which is been used by investigators to discover the devices over the network instead of fingerprinting. The limitations of the fingerprinting lead the researchers to use banner grabbing especially due to the relatively short time span of short lived snapshots and due to a large number of IoT devices compared to inadequate training data availability. Banner grabbing technique enables the investigators to extract from the application layer in the textual form to labeling the internet of things devices.

Antonakakis et al. in his research along with other techniques used the active scanning technique. In this study, it is proposed that to determine the effects of Mirai on the manufacturer and model of the devices Censys is been used because it actively scans the IP4 space, and the data on the application layer of the hosts present on the internet is aggregated. In this study, the protocols of the application layer been scanned were HTTPS, SSH, FTP, CWMP and Telnet. However there was a number of challenges were faced by the researchers in labeling the devices accurately like while scanning HTTPS protocol, the outwards facing services were stopped by the servers of the infected device due to which infected with Mirai devices were prevented from scanning. While the second problem faced was that when scanning with Censys, it took approximately 24 hours to complete during which the infected device may change its IP address. To overcome this problem the researcher used the regular expressions of the nmap service probes to fingerprint the devices to identify the manufacturer and model of the devices. In this study, it is claimed that during scanning with regular nmap rules 98% of SSH, and 81% of FTP banners were matched and extracted data of the device used to label them. However, only 7.5% of the telnet banners were matched. Moreover, the nmap could not handle the CWMP and HTTPS banners, to accommodate these two protocols and increase the coverage of the banners, researchers constructed their expressions to map the banners to the manufacturer and model of the device. Despite all these efforts the researcher finally figured out that the data extracted from the application layer protocols SSH, Telnet, CWMP and FTP is not enough to determine the manufacturer and the model of the device because nmap could only make it to extract the generic description of the device. This study only makes it possible for up to 35% of banners to identify the device type, model and manufacturer [8].

Fachkha, Bou-Harb, Keliris, Memon, and Ahamad in their studies used the darknet preprocessing model and CPS probing orchestration fingerprinting techniques to identify the devices. In their study, they used the manual rules and wrote expressions through which it becomes possible to identify the industrial control system in cyberspace. However, in both studies presented the results were true to this extent but are not applicable and no discussion has been provided in the case of ultra-dense networks and IoT devices [9].

As a part of the literature review of banner grabbing, the researcher found two search engines, Shodan and Censys which are equally popular for discovering online devices. Both search engines use SSH, HTTPS, Telnet and FTP protocol of the application layer to execute the internet-wide scan. However, Shodan search engine for discovering the online devices uses nmap tools along with a manual collection set of rules [10]. On the other hand, Censys search engine uses Ztag; a utility for annotating raw scan data with additional metadata. As, both search engine uses a set of rules, nmap and Ztag in shodan and Censys respectively along with their manual set of rules but rule generation in the banner grabbing is manual. For this purpose, technical skills are required to write the rule for the banner grabbing. However, in the NGWN the 5G ultra-dense networks will support more IoT devices will make it difficult to write manual rules for the identification of the devices through banner grabbing [11].

3. Design and approach

The design and approach presented in this study have two dimensions, in the first phase, a tool for the incidence response of IoT devices will be identified in this study. To discover IoT devices on the network a tool is needed to start incidence response. The basic idea presented in this paper is based on the scanning of available ports to identify IoT devices. As the IoT devices are different from the other devices so as their way of communicate is also different. So in this study, the port scanning technique will be used for the classification of the IoT devices. And secondly, that tool will be evaluated in the simulated environment to ensure its functionality and validity.

Internet of Things devices can be controlled using a web browser as the Web APIs are available for the IoT. This may be achieved through different languages like JavaScript and HTML5 bindings. This also enables to integrate it further, but the language chosen for this study is Python due to its dynamics characteristics and fast development and python also supports different third party installation packages. Moreover, python is a socket level programming language along with numerous GUI libraries for the development of the interface easily.

3.1. Initial trials of tools

In this phase, different libraries of device discovery will be tested to get better results in the prescribed environment. At first, the NetDisco is tested and analyzed because it is a relatively fast program which is a non-trivial characteristic of this research. To scan the whole network (depending upon the no. of devices) it took around 10 seconds. This library is pre-designed to do certain tasks like finding the 'homekit' on the network. This feature of NetDisco however reduces the flexibility and its usefulness. In the prescribed environment of this study, this library was tested on smart home devices like Google Home devices, Chrome cast and light bulbs on the network. All these devices are IoT devices, thus this environment covers the IoT aspect of the study. However, as a result, only chrome cast was discovered and identified by the NetDisco. On trying again with some changes and further inspection, it manages to discover all the Google home devices and to surprise it also grouped all the Google devices as one device with the name of Google Cast. Thus making NetDisco library less useful in the case of this study, because it only managed to discover some devices while leaving all other possible Internet of Thing Devices and this study emphasizes and focused on discovering all the IoT devices.

In suggesting and proposing a tool it is never enough or worthy to test only one library as NetDisco, so taking this into account other libraries were also tested. Nmap library for python is another option that was tested in building the tool. This study requires only a simple ping scan of the network to discover the IoT devices, but it is better to be done at a pace. This library at first appears to be very slow that it was thought as not useful and practical in this environment. However, the basic configuration of the flags to find the IoT devices the process of discovering the devices can be paced up to the acceptable extent. However, ping scan time is highly unpredictable as sometimes it is 3 seconds and sometimes it can be up to 10 seconds. Due to this the average time for the ping scan was taken in 10 rounds by building a small script which is 6.14 seconds and it was 4.9 seconds for another 10 rounds. When calculated for 100 rounds the average ping time was calculated as 8.7 seconds.

After completing the ping scan the port scanning was required, for which the Nmap tool was tested and it seems to be failed in the port scanning because it was taking too much time in scanning the ports of the devices available on the network. It becomes infeasible because of its extremely slow speed in scanning the devices. This motivated the development of the customized network library because during the testing of the Nmap and NetDisco library it was also discovered that, when the program is transported to another system it becomes mandatory for the Nmap to set up the variables according to the new environment. The developed custom network library featured all solutions to the problems and also works with built-in python libraries.

3.2. Machine learning

The main focus of the study is to identify and classify the IoT devices, so after gathering all the information from the network the following step is to identify that whether the device is IoT or not. The idea earlier was to collect the information from all the open ports of available devices and classify them based on nature if these are IoT devices or not. One such idea was also in consideration that is Bayesian classifier. In the functionality and success of the Bayesian classifier, the major influenced factor is the nature of the dataset thus did not work effectively. DTA algorithm is another technique that can be used for classifying the devices and this technique helps reduce the running time of the program. The necessity of scanning every device every time would be reduced by eliminating the devices while scanning each port every time as this requires a significant amount of time.

To educate the algorithm or train it so that it can classify the devices is the largest issue in identifying the devices that whether the device is IoT or not. It requires a significantly large amount of device data as a data set so that it can learn from it or simply classify them. As there are not many IoT devices available, that is why a significantly large dataset was not found to work for the identification and classification of the IoT devices. So the developed algorithm works only for the known and limited available IoT devices and so built that it can be quickly reconfigured when enough large dataset will be found.

3.3. Concise approach

Initial testing of different network libraries to identify and classify the IoT devices is learned that for network discovery a custom is required to begin the development. The basic rudiment of this class is that it must be portable enough to run on multiple operating systems. It must also not take much time to scan, identify and classify the IoT devices and it also supports the configurability according to the wish of the investigator and datasets available.

3.4. Discovery of devices

Discovering the device on the network is achieved using ping as mentioned earlier. However, without determining the active devices on the network Ping can be a time killer technique. Pinging the devices for the whole network and subnet and then waiting for the reply from the devices to determine the active status of the devices can be very time consuming. In this regard, thanks to the network Classless Inter Domain Routing protocol that can help to determine the active devices without pinging the devices on the whole network. Thanks to python’s socket library that will do the work to find the individual IPs of the active devices within the range of the CIDR.

3.5. Port Scanning by TCP

As in the initial testing phase it was established that the port scanning technique can be used to scan for the open ports of the IoT devices, present on the network, this is also done using Python’s socket library. This is lower level scanning but quick and effective because it can be done on systems of different nature.

4. Implementation

To detect the IoT devices on the network for which the above experiments were carried out and the results are presented in the implementation section.

4.1. Overview

The netscan.py file is the main rudiment of the proposed idea of this research. This file contains all the required tools for scanning the devices available on the network. The following screenshot shows the NetScanner class is detailed in [Figure 2].

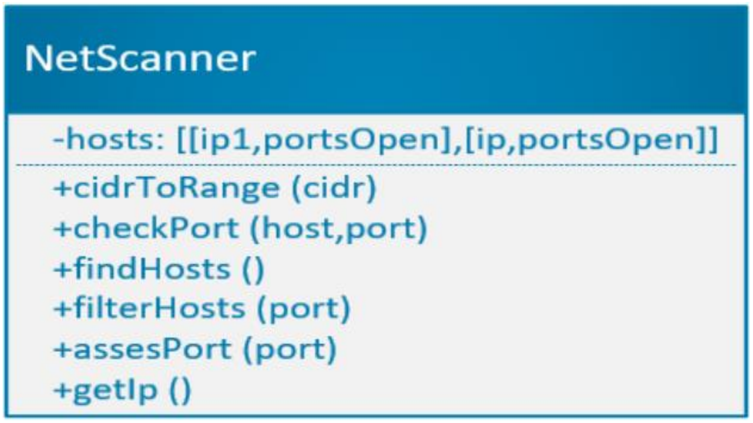


Figure 2. NetScanner class

As from the figure above, the “hosts” is the only variable of the NetScanner class. This variable in the NetScanner will show only the ports which are found open during the scan and it will contain the number of all hosts available on the network. All the functions this class will perform are significant and the main features of the class. However, this class and its functions are so built that some of these will be internal while others will be used externally.

4.2. Runtime

Internet Protocol (IP) or Classless Inter Domain Routing Protocol (CIDR) range is given to initialize this class. However, IP and CIDR range have slightly different functionality thus, in the case of CIDR range is given another functional CIDRTORANGE function is used that will set the variable of the class as hosts, but if the IP address is given then this class will determine and identify the active devices present on given IP. If the CIDR range is given then the active devices present within that range will be discovered by the function. The flow of the instruction is given in the flow chart below in [Figure 3].

After discovering the devices available on the network by scanning ports, the next step is to determine the IoT devices. The hosts that have supplied ports can be eliminated from the list using the filter hosts function. For instance, the device can be removed whose port number 80 is active. This process is further detailed in the flow chart in the subsequent [Figure 4]. The specified port can be accessed through the asses hosts function. The 2nd element of the host's array will be altered by doing so. This way the number of checked ports can be determined. This is also further detailed in the flow chart in [Figure 5]. The basic idea behind this is to identify and determine the probability of the presence of the IoT device for which the decision tree is built.

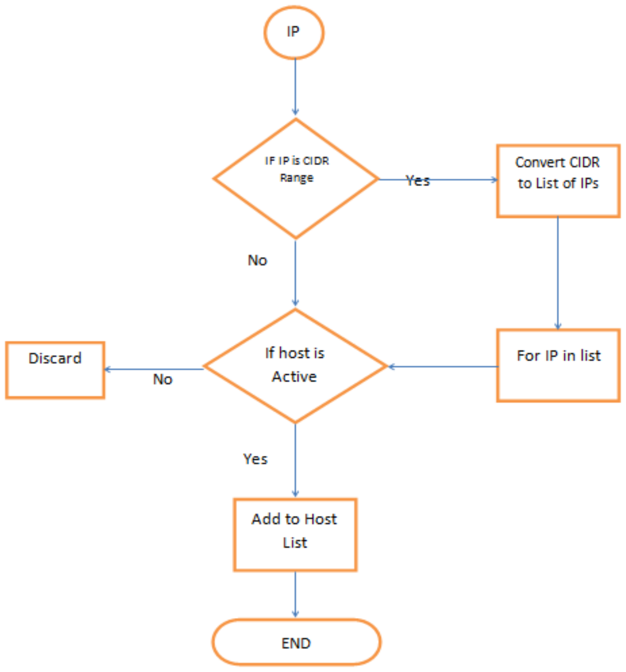


Figure 3. Determining Host

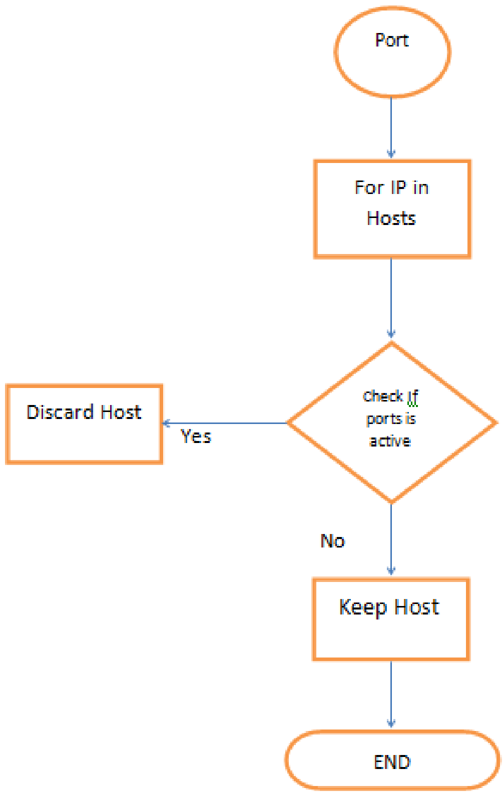


Figure 4. In case of filter ports

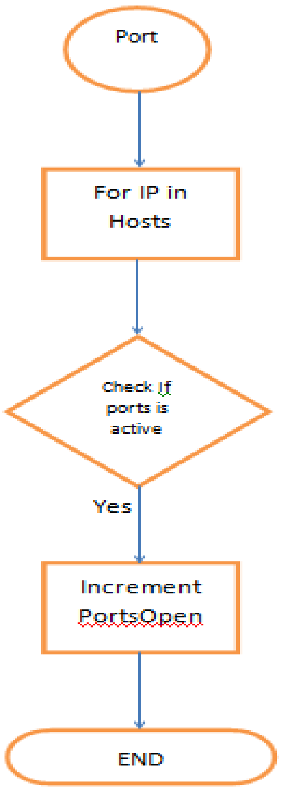


Figure 5. Specified port case

4.3. Selected ports

As earlier, it is mentioned that not enough data set was found to test for the identification and classification of the IoT devices because of the common guidelines for the elimination of the devices from the list. The filtration process starts from two application layer protocols for web traffic and SSH i.e. port 80 and port 22 respectively.

The reason for choosing these ports to be filtered first is quite simple that IoT devices will more probably not use these ports for web traffic and SSH. Like these are some more ports that are not likely to be used by IoT devices such as port 445 that is used for the Windows SMB.

During the assessment of port scanning, it is noticed that port 8008 is the useful port to accomplish the task of this research. This port is an alternated of HTTP port 80 and port 8080 on the application layer. This port is used by most of the Cast enabled devices used in IoT. This established the fact that port 8008 is the open port on all types of IoT smart devices such as chrome cast, Fire TV stick and Google home. Moreover, during the study, it is also noticed that most of the mentioned devices used in this case port 8009 is also identified as an open ports. This port is used for jserver protocol. In addition to these ports, 9999 is also found to be an open port when the network is scanned for the IoT devices available.

Test results show that when the filters mentioned above in the figures are used, it was able to successfully identify the three Google Home devices. The implementation of the whole process is illustrated in [Figure 6] below.

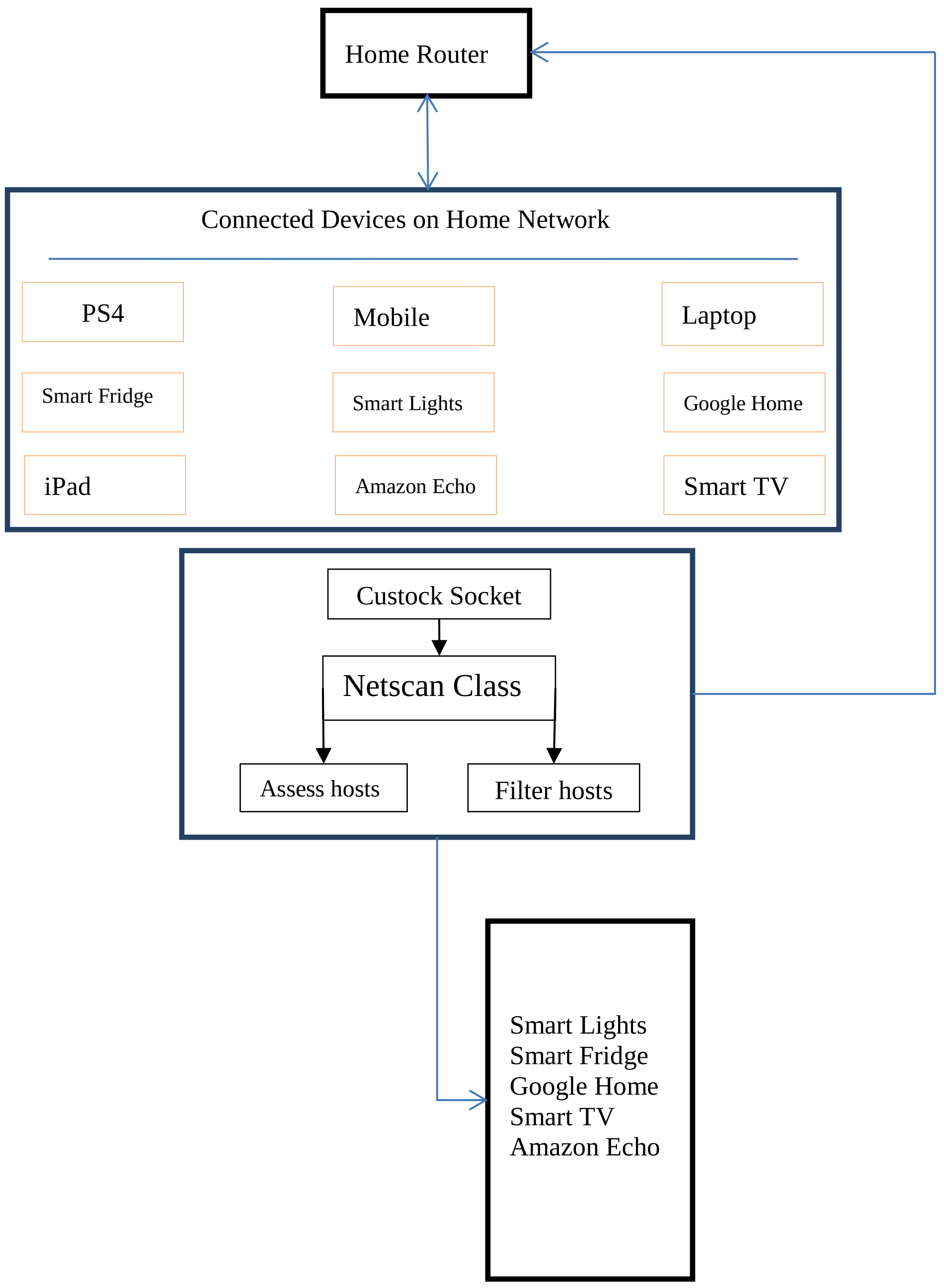


Figure 6. Comprehensive Process

5. Result outcomes and analysis

Insight of the user's daily life can be investigated and through gathering its information like its scheduled tasks and routines in their IoT enabled devices. As far as the incidence response of the IoT devices is concerned having information about the controlled devices is the major aspect. For instance, the routine of any person entering his/her room or leaving the room can be quite obviously observed by the routine of smart lights turned on/off. In the same way surplus information about the user can be unveiled if analyzing the traffic from the smart fridge, TV sticks and other Google Home products even the user's eating habits. Google Home products as earlier said are the major source of information about any user, as it only takes "hey Google" or "okay Google" to send the voice command to do the particular task. For instance, if the user says "Okay Google Turn on the Fan", this voice command will be interpreted by Google and will perform the task whatever it was specified in the voice command. The device after listening to the voice command will send this command to the remote servers of the Google Home whichever is responsible for turning the Fan On and communicating with the particular device and then from that server the turn on command is sent to the particular appliance.

Another aspect is that whatever voice command or signal is sent to the device in this respect is handled by the Google Assistant and is then stored in the Google, disregard the mobile devices or Home products. This stored information is available to the user in the history of their respective accounts. The information of whatever is sent to it, not only the command which was sent to it, but also the time and date when it was run, the voice command audio clip itself, and the process of the command and respective service triggered by the command. If this information is accessible to the investigator it will be quite valuable for the investigation. However, companies like Google have no access policies to the private data of the user that is why it is quite difficult for the investigators to retrieve data.

6. Single filter approach

According to the working of the code, a mathematical model is presented in this study. IoT devices will be shown in the list of output devices when an input is provided consists of a list of IPs and the open ports available on those IPs associated with the input IPs, this will compare with the list of ports which are known to be associated with non-IoT devices along with the list of open ports that are known to be associated with the IoT devices. This means that the result will be only those ports that are associated with the IoT devices and the IPs that will be shown in the list will only be the IPs that are confirmed to be hosting the IoT devices. The single filter approach is advantageous in terms of single statement usage for the iteration of the complete list of given IPs, this also increases the efficiency of the code and the utmost runtime of the code is O(n). On the other hand, this approach necessitates that the given IPs and associated ports are IoT ports whether known or unknown and should be presented in advance as illustrated in the equations below.

$$IoT\_Devices = \forall IPs \in IP\_List: (S\_Ports \cap IoT\_Ports)\ andand\ (S\_Ports\ ! \cap Non\_IoT\_Ports) \quad (1)$$

$$IoT\_IPs = \forall IPs \in IoT\_Devices: IP\_List(IoT\_Devices) \quad (2)$$

The variables involved are shown in [Table 1] below.

Table 1. Definition of variables used in equations

Variable	Definition
IoT-Devices	Array of IoT devices with associated IP list according the model's output
Potential-IoT-Devices	Array of potential IoT devices indexed in the IP list according the model's output
IoT-IPs	Resultant array of IPs of confirmed IoT devices

IPs	IPs include in the list
IP-list	IP array generated due to network discovery scan
S-ports	ports related to the current IP being compared to
IoT-ports	All known IoT ports
Non-IoT-Ports	All known non-IoT ports

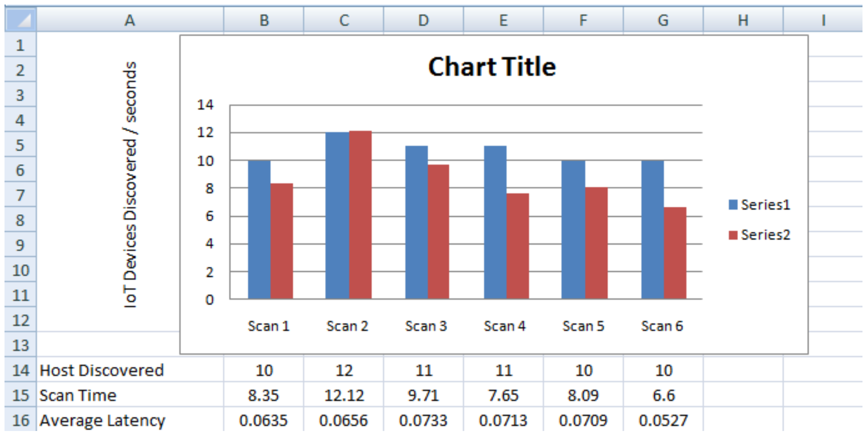
7. Double filter approach

The double filter approach can also be used to filter out the IoT devices. Following equations is the mathematical expression to know the potential and confirm IoT devices.

$$Possible\_IoT\_Devices = \forall IPs \in IP\_List: (S\_Ports\,! \cap Non\_IoT\_Ports) \quad (3)$$

$$IoT\_Devices = \forall IPs \in Possible\_IoT\_Devices: (S\_Ports \cap Known\_IoT\_Ports) \quad (4)$$

In this approach, the non IoT devices will be filtered out first against the ports that are known to be non IoT ports. The rest of the devices left in the output array will most probably be IoT devices. According to the equation (3) and (4) the filtered devices will again be compared to the list of known IoT ports. Due to the double filtration of the devices, the total run time might increase i.e. *O(n+m)*, but it is necessary because it might be possible that in the first filtered list some IoT devices are still filtered out even compared to the known non-IoT devices.



A	B	C	D	E	F	G
Host Discovered	10	12	11	11	10	10
Scan Time	8.35	12.12	9.71	7.65	8.09	6.6
Average Latency	0.0635	0.0656	0.0733	0.0713	0.0709	0.0527

Figure 6. Result of discovered devices

In this research 6 test scans of the network were conducted and the resulting graph is shown in the figure above. This scan shows at least 10 devices on each network scan out of 256 IP networks. This scan also provided the manufacturer names e.g. Apple, Samsung, etc. As far as the investigation is concerned the information extracted is very alluring and as shown in the results above that, not more than 10 seconds were taken for each network scan. Manufacturer name extraction is very important because on these bases the device can be assessed for instance, if the manufacturer appears in the scan is Chrome cast paired with the host closer like by the investigator can be taken in the network as an IoT device and some possible clues might be found.

Table 2. Definition of variables for probability determination

Variable	Definition
X	No. of devices on network
Y	No. of open ports associated with the device
P	No. of IoT ports known

To find out the probability of any one port scanned being an IoT port it will first be assumed that all the networks will have x_n number of connected devices having y_m number of associated open ports. While on the other hand "p" in the above [Table 2] shows the number of known IoT open ports associated with the devices. So the probability can be calculated as:

p/65536------- probability of anyone port being an IoT port

The probability of having the IoT open port on the network can be calculated for each connected device as follows:

$$\boldsymbol{y}.\frac{\boldsymbol{p}}{\boldsymbol{65536}} = probability\ of\ presence\ of\ IoT\ device\ on\ the\ port \tag{5}$$

$$(\textstyle\sum_{\boldsymbol{i=1}}^{\boldsymbol{x}} \boldsymbol{y}i\,.\frac{\boldsymbol{p}}{\boldsymbol{65536}}) = probability\ of presence\ of IoT\ port\ on\ Network \tag{6}$$

The mathematical model presented above can determine that any fruitful results can be expected or not if netscan code is running on the network. For device isolation and investigation running the netscan code will be the next logical step in case of more probability of finding the IoT device.

8. Future work and conclusion

Additionally, in this research, the device taken into account were only those that were connected to the Wi-Fi network. However, not necessarily all the devices connected to the Wi-Fi devices may connect using other wireless technology, so it is to be accounted for in further research. Using Web APIs for IoT helps find all the devices present on the network and also in determining the connection type being used worth further investigation. During this research increasing incidence of smart devices on IoT networks in routine life. Another aspect that worth research is to conduct an in-depth investigation focused on the control devices and the type of information that can be retrieved from them. Complete control over the IoT devices is allowed by the Web APIs for the IoT, on the application layer using adapted Web Runtime Engines running on the IoT devices which means that the adapted version of the presented code enables the investigator for having extended access and control of the device. The study shows that the information extracted from the investigation of the IoT network can be very useful for the investigator for incidence response. In this study, the tools are proposed which is helpful in the investigation process. The tool to discover the internet of things devices present on the network effectively the presented program produced the tool for this purpose. Moreover, the identification and classification of the potential open ports that are associated with the IoT devices is also achieved, and also determine that the associated devices are IoT or non IoT devices.

Acknowledgment

The author of the paper would like to thanks Dr.Yongle Chen for his kind supervision and support through the completion of this study. Without his efforts and worthy contribution, this study would never be produced with effective results. The author would also like to thanks the fellow authors for their support and contribution.

References:

[1] Huffstutler A., “Video of aggravated assault shared on social media leads to arrest,” Retrieved 18 july, 2019, http://www.wrcbtv.com/story/37420342/video-of-aggravatedassault-shared-on-social-media-leads-to-arrest, **(2018)**

[2] Pandya J. A, “Changing internet: The convergence of blockchain, internet of things, and artificial intelligence,” AI and Big Data Retrieved 18 July, 2019, from https://www. forbes.com/sites/cognitivewo rld/2019/07/05/a-changing-internet-the-convergence-of-blockchain-internet-of-things-and-artificial-intelligence/#34d9a7297c58, **(2019)**

[3] “How Big is IoT?” 20.6 Billion Connected Devices By 2020, Retrieved 18 July, 2019, from https://mitechnews.com/internet-of-things/how-big-is-iot-20-6-billion-connected-devices-by-2020/, **(2019)**

[4] Shamsi Z., Cline D. B., and Loguinov D., “Faulds: A non-parametric iterative classifier for Internet-wide OS fingerprinting,” Paper presented at the Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, **(2017)**

[5] Caballero J., Venkataraman S., Poosankam P., Kang M. G., Song D., and Blum A., “FiG: Automatic fingerprint generation,” **(2007)**

[6] Shamsi Z., Nandwani A., Leonard D., and Loguinov D., “Hershel: Single-packet OS fingerprinting. IEEE/ACM Transactions on Networking (TON),” vol.24, no.4, pp.2196-2209, **(2016)**

[7] McClure S., Scambray J., and Kurtz G., “Hacking exposed fifth edition: Network security secrets and solutions: mcgraw-hill/Osborne, **(2005)**

[8] Antonakakis M., April T., Bailey M., Bernhard M., Bursztein E., Cochran J., and Kallitsis M., “Understanding the mirai botnet,” Paper presented at the 26th {USENIX} Security Symposium ({USENIX} Security 17), **(2017)**

[9] Fachkha C., Bou-Harb E., Keliris A., Memon N. D., and Ahamad M., “Internet-scale Probing of CPS: Inference, Characterization and Orchestration Analysis,” Paper presented at the NDSS, **(2017)**

[10] Durumeric Z., Adrian D., Mirian A., Bailey M., and Halderman J. A., “A search engine backed by Internet-wide scanning,” Paper presented at the Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, **(2015)**

[11] Durumeric Z., Bailey M., and Halderman J. A., “An internet-wide view of internet-wide scanning,” Paper presented at the 23rd, **(2014)**

[12] {USENIX} Security Symposium, Security, 14