Securing the Internet of Things using Tor Hidden Services, and Anonymity

[redacted]

Dr Mohammad Hammoudeh

BSc Computer Science

Abstract

Moving forward with Internet of Things (IoT) we are seeing a exponential increase in the amount of IoT devices that are in circulation. However, security has not kept pace with the growing challenges of privacy. This project aims to inform the reader of the issues surrounding IoT security, and to implement a solution using The Onion Router(Tor). Such a solution will protect IoT stakeholders privacy without the overhead of modifying application's underlying architectures or processes. Doing this I have found Tor to be a solution that would work in home and business use. Also finding that latency could be an issue for systems that are time sensitive.

Contents

List of Figures

Chapter 1

Introduction

Moving forward with Internet of Things (IoT) we are seeing a exponential increase in the amount of IoT devices that are in circulation. However, security has not kept pace with the growing challenges of privacy. With the wide verity of IoT devices and applications, it is much harder to implement solutions to mitigate security threats compared to classical networks. Whether accidental or malicious, interference with hospital monitoring devices, a car, or a nuclear reactor poses threat to human life. Currently, there is no holistic solution to securing the IoT.

In classical networks, the issues of privacy and security has been dealt with using tools such as Tor (TheTorProject 2016). In theory using the Tor network to advertise the IoT service as a hidden service will negate all previous possible attacks to an IoT service, and this is what I will explore in my project.

1.1 Project Aim and Objectives

1.1.1 Aim

The aim of this project is to produce a Tor driven anonymous service to secure IoT users and applications. The solution will abstract IoT devices and applications into hidden services that can be accessed securely and anonymously over the Internet. Such a solution will protect IoT stakeholders privacy without the overhead of modifying application's underlying architectures or processes.

1.1.2 Objectives

To achieve the aim of this project, the following objectives must be met:

- Conduct a literature review to identify the common critical IoT security threats that can be mitigated using hidden services.
- Investigate current tools/platforms to provide security to IoT devices using Tor.
- Identify an open source IoT connection-focused platform suitable to extend to provide hidden services to the IoT ecosystem.
- Implement a plugin to allow users to integrate their devices with their own hidden application or service.
- Configure and deploy the server infrastructure of the chosen open source platform into a local machine.
- Evaluate the new hidden service using real IoT objects.

1.2 Background

1.2.1 An Overview of the Internet of Things (IoT)

IoT is the networking and remote control of every day devices, ranging from a fridge or a family surveillance system to a whole building's central heating and ventilation system. The concept of IoT and the communication and networking of devices has been discussed as early as 1982. The first to implement this was a Coke machine that was able to report its inventory and if new stock in the machine was cold yet or not.

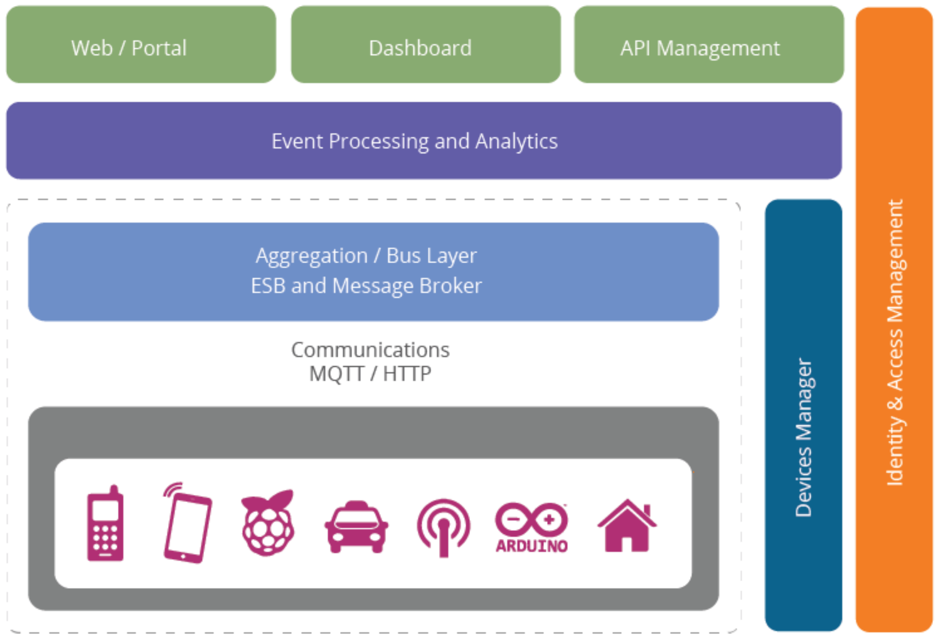


Figure 1.1: An illustration of the IoT architecture, adopted from (Fremantle 2016)

As seen in Figure 1.1, the IoT architecture can be broken down into five layers. The lowest layer being the device layer where many device types resides. Each device have to have radio equipment to attach it to the Internet to be considered an IoT device. The communication layer supports the connectivity of the devices, using a communication protocol such as HTTP/HTTPS, or MQTT. On the third layer the Aggregation/Bus Layer aggregates and brokers communications. The Event Processing and Analytics Layer then takes the events from the bus and provides the ability to process and act upon these events (Fremantle 2016). The IoT architecture will then need a way for the device used to communicate outside of its system, this is the purpose of the External Communications Layer. This layer breaks down into three components. Firstly, part of the layer deals with the interaction between websites/web-portals and the device. Secondly, the capability of the device communicating with Dashboards is needed. Finally, the API Management section manages device to API communication.

Using IoT, users are opening themselves up to a great deal of vulnerabilities, including the opportunity for surveillance from an entity to gain access and exploit private information against individuals or companies. This is why security is very important when using IoT. This is far from hypothetical as attackers stole 40 million credit card numbers after they hacked into a national retailer's HVAC system and used it to reach their computer system and their customers (Ailanthus 2016).

Added security is needed on the IoT system implemented in that company. IoT can also be abused in large scales, if you were to gain access to a large amount of the IoT devices these could be used to send requests to any server around the world. Put into practice this could shut down any server as the amount of requests that the server would receive far too many requests to handle, making that service unavailable to its intended user. Recently this has been brought to light by a group of hackers using IoT devices they accumulated using a piece of malware known as "Mirai". Using the devices gathered they were able to send requests to servers to shut them down "On Friday morning, someone targeted Dyn, a company that offers core internet services for popular websites such as Twitter, Spotify, Github"(FRANCESCHI-BICCHIERAI 2016) all of those services where taken down using only ten percent of the available "Mirai" botnet.

1.2.2 An Overview of the Tor Anonymity Network

Tor was developed in the mid 1990's by the United States Naval Research Laboratory, with the purpose of protecting U.S. intelligence communications online. In 2004, the Naval Research Laboratory released the code for Tor under a free license. In 2006, the Tor Project was founded as a research and education, non profit organization responsible for maintaining Tor. Since then, network anonymity has led many people to start using the Tor browser to navigate the "hidden service" found using Tor.

A hidden service is just a normal service that could be offered on the web, but if you do not have the URL you wont be able to see it, e.g., web publishing or instant messaging server. The hidden service can make itself available by advertising its existence to the Tor network. By contacting random relay nodes on the Tor network the service will now be available to view if you have the URL to view it on. An example of a hidden service would be DuckDuckGo, which is a search engine advertised as a hidden service on this URL: http://3g2upl4pq6kufc4m.onion/, only accessible when using Tor.

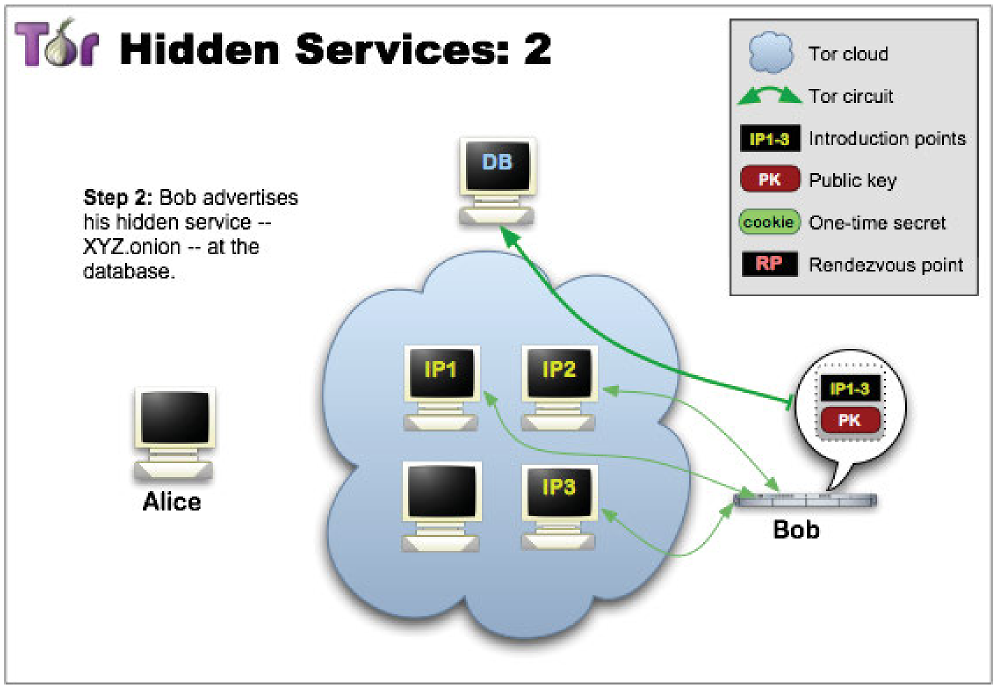


Figure 1.2: Hidden service availability, adopted from TheOnionProject (TheTorProject 2016)

In Figure 1.2 we can see the start of a server advertising its availability as a hidden service. Meaning it has just advertised its self to some nodes on the network for introduction points. The second step is assembling a hidden service descriptor which contains the public key and summary of each introduction point, and signs it with its private key. This will be found by clients requesting access to the server using a 16 character name derived from the service's public key. Now a client can initiate a connection to the server using this onion address.

"Using Tor protects you against a common form of Internet surveillance known as "traffic analysis" (TheTorProject 2016). This is a main benefit of using Tor, as using the network traffic is transferred between hundreds of volunteer-operated relays. As this data is encrypted, none of the volunteer nodes hold two crucial bits of information that could be used to track who is searching for what. The two crucial bits of information being; the source of information, and the destination of the information. Virtually removing your footprint from every node connecting you to the destination server.

Tor will obtain a list of Tor nodes from a directory server, then will build up a circuit of encrypted connection through relay nodes on the network. Put into practice this will make an "anonymous network". This is because between you and the site you are trying to access, if anyone were to intervene the network traffic they would only see parts of the information transmitted from you, on any given Tor relay. In Figure 1.3, we can see this in practice, the route through a Tor network. When established it becomes an anonymous network as stated before.

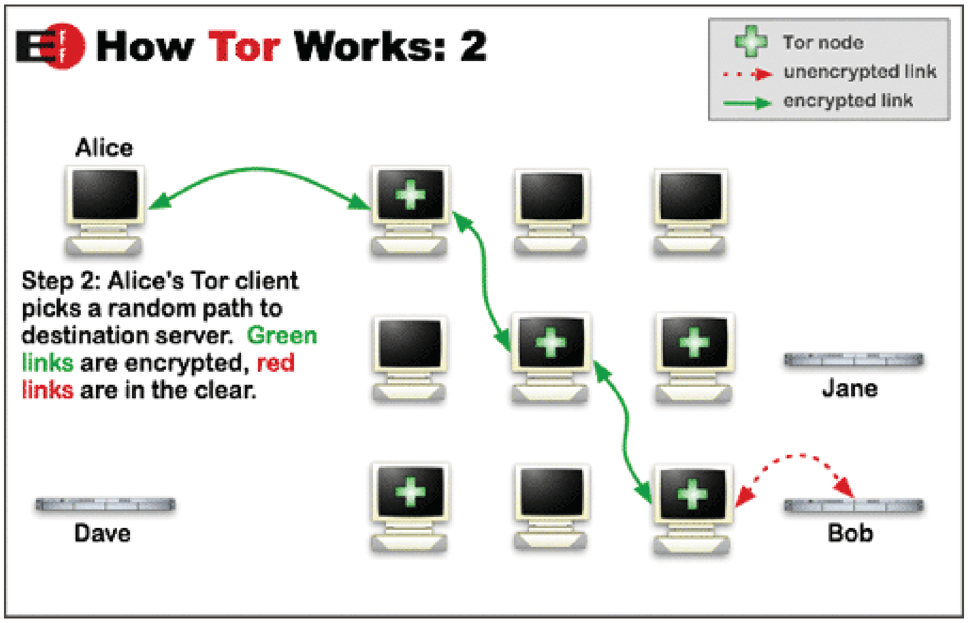

How Tor Works: 2
Tor node
unencrypted link
encrypted link
Alice
Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Jane
Dave
Bob

Figure 1.3: Navigation through Tor's network, (TheOnionProject)

1.3 Report Organisation

Each of the chapters and their contents are listed here:

1. Introduction: The introduction contains the aims and objectives of the project, also introducing the project with the background.

2. Literature Review, IoT Threats Landscape: In this section we evaluate common security threats for IoT devices, and it's threat landscape. Then evaluating current solutions before discussing the benefit of the Tor solution.

3. IoT-Tor (IoToR) Solution Design: This section introduces the reader to the requirements needed to complete the solution. Then evaluating the correct design model to use before implementing the IoToR design.

4. IoToR Implementation: In this section we implement the IoToR and test the solution.

5. Evaluation: Analysis of product as a solution and evaluation of testing.

6. Conclusion: Summarizing the project, including information regarding recommendations for future work.

Chapter 2

Literature Review, IoT Threats Landscape

2.1 Common Security Services in IoT

There are many common security aspects to devices that can be configured for use in many IoT devices. General security for IoT devices is implemented in a multi-layered approach, as not one single control will be able to secure the device. The security starts when power is applied to the IoT device.

In the following subsections we will be looking at different method of securing devices from the ground up, starting at the firmware. We then look at different method of securing devices from conventional methods.

2.1.1 Secure Booting

After power is first introduced into the device, the system firmware checks that the system boot loader has not been tampered with by checking the cryptographic key has been authorized by a database contained within the firmware. It will also prevent the loading of drivers that are not signed with a digital signature that it will accept (Niehus 2013). This will only be done on IoT devices with UEFI 2.3.1 or higher. This is to help prevent execution of un-signed code on the system. There are three main actions regarding Secure Boot on devices which are outlined in the diagram below:

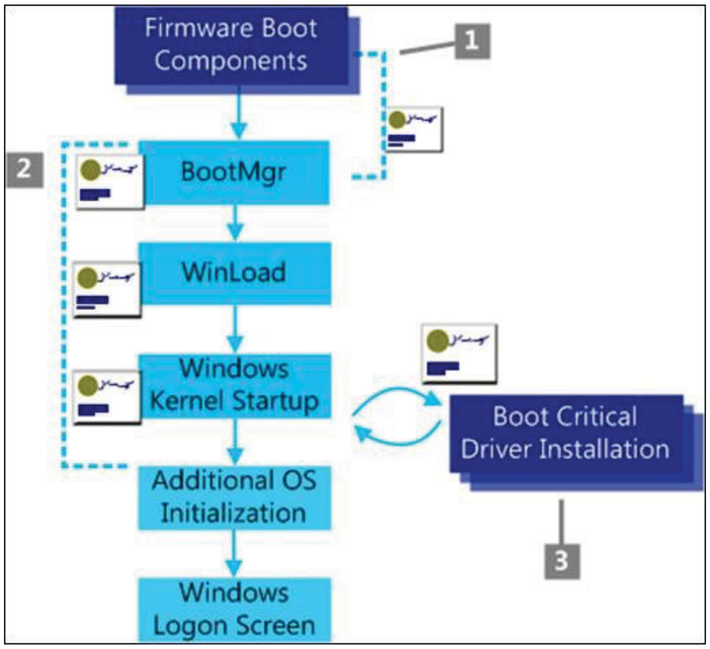


Figure 2.1: The main actions related to Secure Booting (Niehus 2013)

As seen in Figure 2.1 the three labeled main actions are as follows:

1. The system Firmware verifies all UEFI executable files and the OS loader, this is to ensure that they are trusted.
2. OS Boot Components then verify the signature of each of the components to be loaded, and any non-trusted components will not be loaded.
3. Finally, the signatures of all Boot Critical Drivers are verified before initialization to check for malware.

2.1.2 Access Control

Once the device has successfully secure booted different forms of resource and access control are applied. Mandatory access controls built into the operating system limit the privileges of device components, and applications so they can only access resources required to run (River 2013).

If the system is compromised access control is designed so that the user has minimal control on the other parts of the system. This is an important step as having root access to the device is very powerful. Therefore, when setting up an IoT device it is important to set different levels of access. Once in place, the device would be used under an account with no administrator privileges, giving less access to the device.

2.1.3 Device Authentication

When the IoT device is connected to a network, it will authenticate itself prior to the transmission or retrieval of data. As IoT devices often do not have users waiting to input credentials into the device this is important. IoT devices will benefit from this as machine authentication allows the device to connect to the network using a similar set of credentials a user would, and they are stored securely.

2.1.4 Firewall and IPS

IoT devices also need a firewall or a packet inspection capability so that the device can control the traffic that is destines to arrive at the device. The IoT device does not need to filter the higher-level protocols as the network appliances will take care of that. But, the device does need to filter the data destined to terminate at the device, in a way that makes use of the limited computational resources available.

2.1.5 Updates and Patches

Once in operation, the IoT device will start receiving patches and software updates. When the patches are rolled out the IoT device will need to authenticate it for use without consuming too much bandwidth, or without impairing the functional safety of the device.

2.2 Current IoT Threat Landscape

Currently, many security flaws arise from implementing an IoT service on the open web. In Figure 2.2 adapted from (Babar et al. 2011) we can see current possible attacks to IoT devices.

In the following subsections, we introduce different attack vectors for IoT devices. For each implemented IoT device the attack surface has to be established, so that the issues with the device security are re mediated, as the attack vectors are different per device.

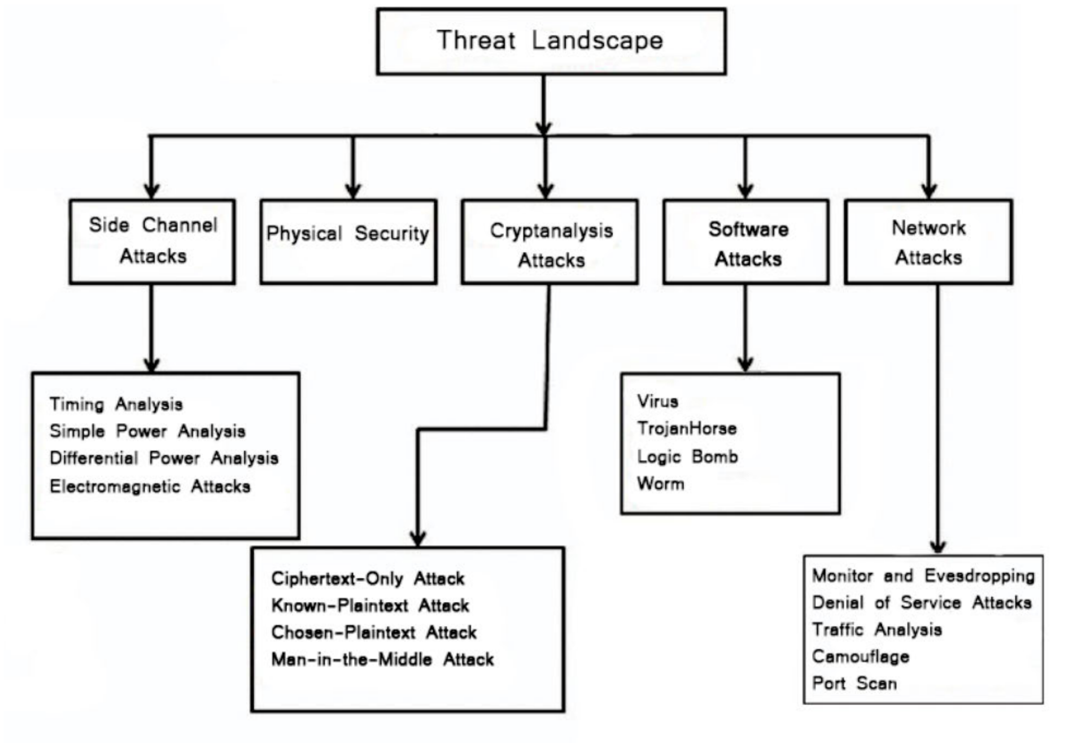


Figure 2.2: Available attacks on IoT devices

2.2.1 Side Channel Attacks

Side channel attacks are attacks that retrieve information from the data source, gathering data such as power consumption and temperature from the IoT device. Devices with encryption produce timing information that is measurable, using this information to recover the encryption key the device is using (Babar et al. 2011). In terms of IoT, all devices that send encrypted traffic are vulnerable, and if they do not they are vulnerable to different attacks that take less time. There are many side channel attacks that effect IoT devices including, Timing Attacks, Power Consumption Analysis, Differential Fault Analysis, and side channel attacks that use sound or electromagnetic emanations. These attacks are often best used on IoT devices that perform different operations based on the data currently being processed, such as a device that uses RSA due to the factorization complexity of the encryption keys.

Timing Attacks on IoT devices would find the time taken for the device to execute the encryption algorithm. This is because the operation will take time to execute and the only dependent will be the input. Working backwards from this the attacker could use the time taken for an operation to determine the input. Put into practice an attacker could measure how much time the IoT device takes to respond to certain queries, this is especially deadly in IoT devices as many devices are easy to access through a web interface.

Simple Power Consumption Analysis (SPA) involves analyzing power consumption over time via visual interpretation of the data. The user will see variations in the power consumption of the IoT device as it performs different operations, if RSA is implemented the squaring and multiplication operations can easily be distinguished. After the attacker has this information they can work out the secret key used. Because the hacker will have to be next to the device, this attack is impractical for large scale IoT device attacks. It can also be a problem if the hacker wants to infiltrate IoT devices in a secure building because the attacker will need to be next to the device.

Differential Power Analysis (DPA) is a more advanced side channel attack than the Simple Power Consumption Analysis. The concept is the same although sometimes there is a lot of noise within the plotted power analysis data, therefore making it hard to distinguish between the power spikes from multiplication and squaring operations and other operations. DPA uses statistical functions tailored to the target algorithm to gain the secret key (Kocher, Jaffe, and Jun 1999). As with the SPA attack the hacker will need to be next to the device so it is impractical for the same reasons. We can see in figure 2.3 the power analysis of the RSA encryption algorithm, and the difference in the CPU power spiking when processing the multiplication, which is on the right and noticeably longer, and the part of the algorithm not processing the multiplication, on the left. This would not be possible to see as clearly if the IoT device had any one of the countermeasures spoken above, but DPA is set out to remove the countermeasures.

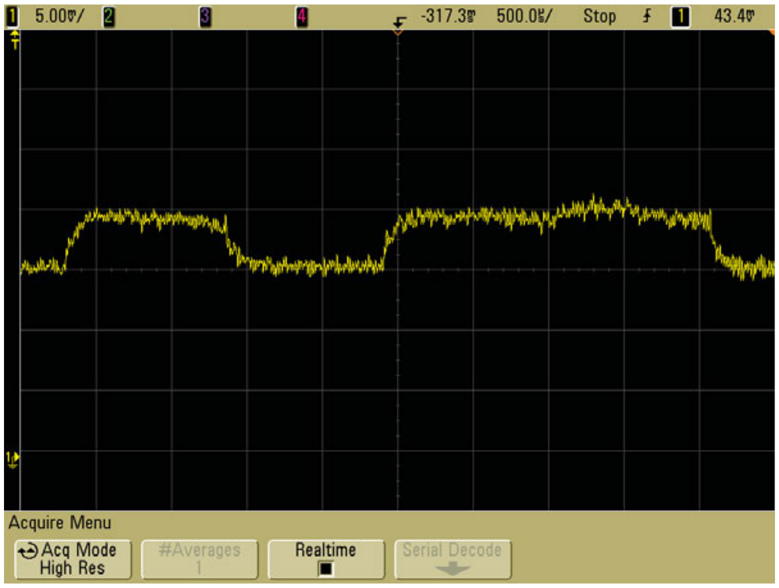


Figure 2.3: RSA power analysis where the left peak is the CPU spike whilst running the algorithm without the multiplication and the right is with, (Audriusa 2016)

Electromagnetic Attacks are directed at measuring the electromagnetic radiation emitted from a device. Using this information obtained from an IoT device an attacker will try to obtain the encryption key from the device. This attack is a non invasive passive attack, giving the hacker an advantage as the owner of the device wont know that it's being compromised. This attack is best performed with other side channel attacks suck as the Differential Power Analysis attack to obtain the private key quicker.

2.2.2 Physical Security

Any IoT device that is left physically exposed will be vulnerable, as then the item could be stolen. The attacker could then reverse engineer the device to obtain shared keys/passwords. Or the attacker could look for vulnerabilities in the software that can then be used to exploit operational products at a later time.

2.2.3 Cryptanalysis Attacks

Cryptanalysis attacks focus on decrypting the cipher text by obtaining the encryption key, using that paired with the encryption method used to gain the plain text. These types of attacks can only take place if the attacker knows what cryptographic algorithm was used to generate the cipher text on the IoT device, otherwise the attacks cannot be carried out by the hacker.

A common cryptanalysis attack that could be used on IoT devices is the Man-in-the-middle attack, often referred to as an MitM attack. MitM attacks exploits the fact that HTTPS servers send a certificate with it's public key to the Web browser. If the certificate is not trustworthy then the entire communication path is vulnerable (Callegati, Cerroni, and Ramilli 2009). The attacker would then replace the HTTPS certificate with a modified one onto the server. If the user then ignores the error warnings and continues to use the site, then the attack is successful.

Cipher text-only attacks are an option to getting onto an IoT device when only the only information the hacker has about the IoT device is a set of cipher text. With this attack while the attacker has no access to the plain text, in any successful cipher text-only attacks the attacker will still have some knowledge of the plain text. This knowledge could be as simple as the language the plain text is written in. Gaining any more information on the plain text than before conducting the attack is a success for the hacker, as this could then be used to conduct other attacks. Such as using the information to help with a traffic analysis attack (Biryukov and Kushilevitz 1998).

Known-plain text attacks can only occur if the hacker has access to an amount of plain text and cipher text, they will then use this information to try and gain more information such as the secret key used to encrypt the text. For IoT devices this could be used in conjunction with other cipher text attacks to gain access to information on the system.

Chosen-plain text attacks set out to gain the systems secret key, much like most cryptographic attacks. Using plain text of the hackers choosing to gain the cipher text of said plain text is the fist step of the attack. For an IoT device this attack would be hard to go through with because the attacker would need to get the device or web server to process a request and the attacker would then need to obtain the cipher text associated. The adaptive chosen-plain text attack is a variation of this attack where the attacker will chose different plain-text based on information learned from previous cipher-texts.

2.2.4 Software Attacks

Software attacks focus on the major source of security vulnerabilities on a system, as they exploit implementation vulnerabilities on a system using the systems own interface. Software attacks encompasses many attacks including using viruses to deliberately inject malicious code into the system. This would be used to take over said system to then use the system to send packets to a destination server to cause a denial of service for the service for example. Because of the hackers easy access to IoT devices over the web and the amount of devices that are left with the default user name and password, software attacks are easy to achieve.

Another software attack possible is infecting the target IoT device with a Virus. The hacker would need to gain access to the device to be able to put the virus onto the IoT device. The virus could then take many forms to execute code the hacker has implemented into the virus. The hacker could either want to take control of the IoT device to gain information or to use the device, or the hacker could want to shut the IoT device down. Using a virus they could infect the machine to achieve these goals by either creating a backdoor for the hacker to access the machine any time and take as much data as they want, or to corrupt the boot sector of the device.

Planting a Trojan Horse onto an IoT device would be the easiest of the software attacks to achieve. Using social engineering the hacker could disguise the Trojan horse as a perfectly safe piece of software that the IoT device needs. This could take the form of a security patch for example. Once executed the user will believe that the software is still genuine but the hacker could gain key data, they could use it as a backdoor to spy on the content flowing through the IoT device. They could also use it destructively, to take down the device or corrupt data. Trojan horses have been used before to use the IoT device as part of a botnet to aid in other attacks such as automated spamming, or distributed denial of service attacks.

Logic bombs are similar to viruses or malware such that they will infect the system with code and like the Trojan horse they will go undetected until the conditions for executing the code are satisfied. In 2013 a logic bomb was used to wipe hard drives, and master boot records belonging to three banks and two media companies on the same date and time(Zetter 2013). IoT devices are vulnerable to this attack as the ease of access and lack of security on devices means that hackers can access many devices at a time.

Worms in the IoT world are often designed to infect other Iot devices, dependent on the intent of the hacker. If there are many IoT devices on a closed system, the hacker will plant the worm on one IoT device, the worm will then spread to the other IoT devices on the network.

2.2.5 Network Attacks

Network attacks target the communication of networked devices. Network attacks are encompassed under two categories, the passive attack, and the active attack. Passive network attacks are attacks that are set out to find data without leaving any evidence that you were there. Examples of passive attacks include but are not limited to; Traffic analysis, Camouflage, and Monitoring and Eavesdropping. Active attacks are attacks that are designed to effect the operation of the system they are targeting, or to alter system resources. An example would be the Denial-of-service attack.

Monitoring and Eavesdropping on an IoT device is where the hacker, once connected to paths on the network that transmit the network traffic will be able to intercept these packages. Without strong encryption security measures such as end-to-end encryption the IoT devices that the hacker is targeting will be vulnerable to eavesdropping. Traffic Analysis is the act of monitoring the network traffic even if the traffic is encrypted. The hacker will look at the encrypted traffic on the IoT device to try and deduce information from patterns in the communication.

Denial of Service (DoS) Attacks have the intent of making the IoT device, or the network that it's running on unavailable to it's intended users. This is done by overloading the IoT device with requests so that it cannot take any more requests, this can be seen in figure 2.4(adapted from (Sullivan 2016)). After a user has taken over an IoT device or multiple devices, using different attacks mentioned above they could all be used to send packets to a chosen destination to cause a DoS/DDoS attack.

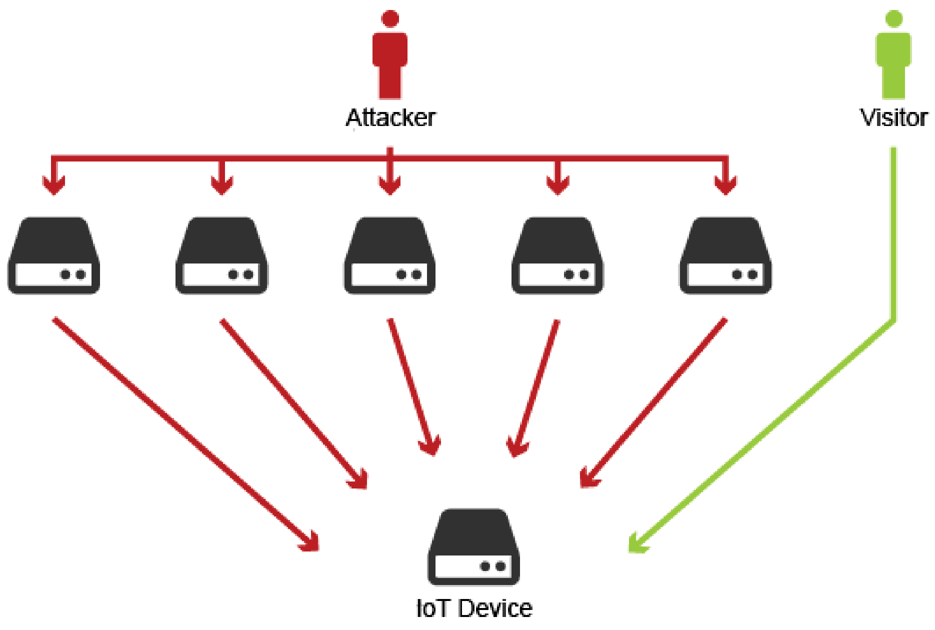


Figure 2.4: Example of a DDoS attack

After the hacker has gained access to the network and has access to the network traffic, the next step would either be to alter the traffic or corrupt the traffic. This is what's known as message corruption/modification.

2.3 Evaluation of Current Solutions

With the sheer amount of possible attacks available to IoT devices, there are solutions in place for most of these attacks. Such solutions are often convoluted and pairing all of the solutions for the different types of attacks would mostly slow down the IoT device with it's limited processing power. With over 6 billion devices to cater for (Meulen 2015), the specific security requirements will differ from device to device. This section discusses the exiting security mechanisms to mitigate the attacks mentioned reviewed in Section [number], whilst also looking at previous work to implement a security packages as a standard for IoT devices.

When using an IoT device the user will need to choose an IoT platform to use their device. The gap between the IoT sensor device and data network is filled by an IoT platform. The platform connects the device to a network to provide a middle ground to make sense of all the data generated by the sensors from the device.

Regarding side channel attacks, a way to prevent any information being obtained by hacker is to implement a standard amount of time that operations take to complete. This is not practical because it will mean that operations will run slower as all operations will need to be held for a small amount longer than it takes the longest operation to complete. To counteract most side channel attacks the person implementing the software for the IoT device could add noise to the algorithms used so that it is a lot more complex for anyone to deduce what is going on whilst analyzing the data. As they are more focused on being with the IoT device itself, the easiest method would be physical security.

Cryptanalysis attacks are in general easier to protect against if the correct precautions are taken with evolving encryption methods; if an encryption method, which is known to be vulnerable, is used then the IoT device is vulnerable. Using the Advanced Encryption Standard (AES) is one way to help as it is not currently known to be susceptible to known-plain text attacks. To prevent man-in-the-middle attacks both sides can compute a cryptographic hash function of the key exchange, sign it using a digital signature algorithm, and send the signature to the other side (Heward 2014). The receiving IoT device would then verify that the hash matches the locally computed hash and the signature came from the desired other party. Using brute force to try all the keys available is a common tactic so implementing a key that's long is crucial to keeping the data on the IoT device secure and keeping the integrity of data.

Security holes in the software and the OS the IoT device is running on can be exploited easily, but keeping the firmware and software up to date will negate most holes for the hacker to exploit. Specifically to software that has been planted onto the machine, it is crucial for the administrator of the device to run malware/virus scans regularly to check if the system has been compromised. An easy way to make sure the firmware on the IoT device has not been tampered with is using secure boot, which uses cryptographically signed code along with hardware support to make sure the firmware has not been tampered with (Grau 2016).

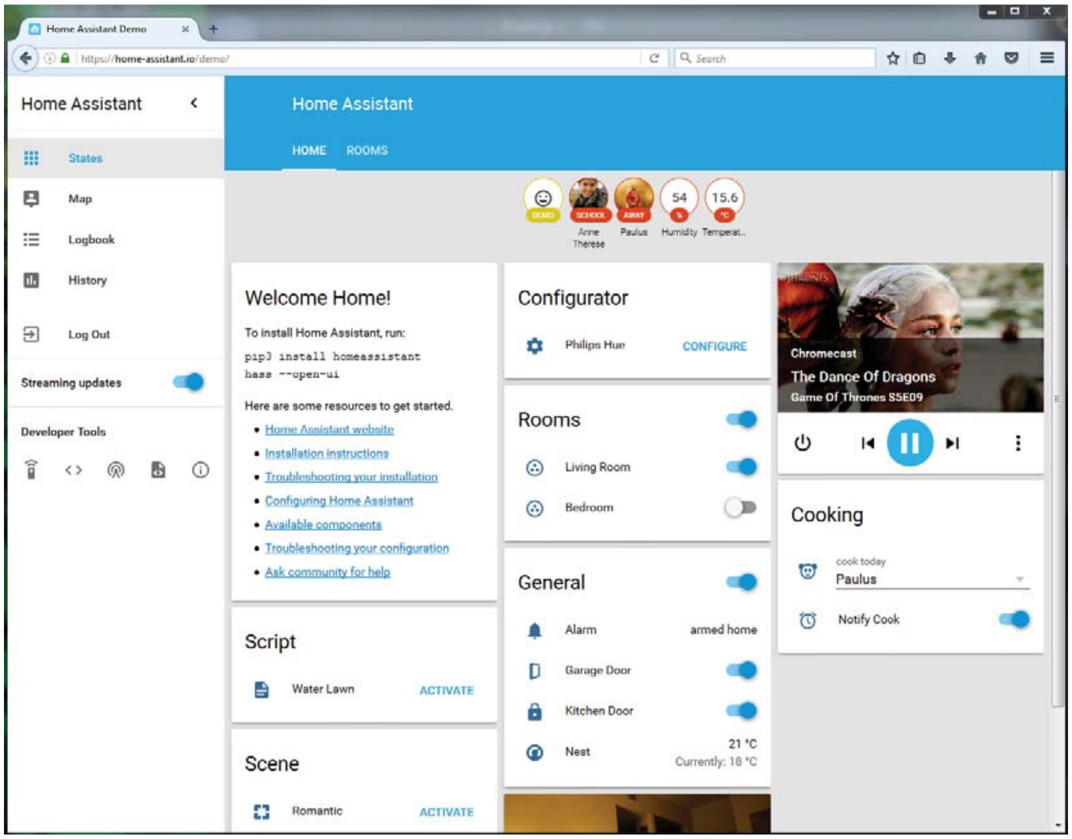
2.3.1 Current Solutions

Currently, there are many IoT platforms available to choose from. For these platforms there are security limitations and issues that arise. Most products are application and platform specific and don't adapt easily to other IoT devices. Therefore I will be focusing on a few existing solutions that meet the most of the security criteria.

Home Assistant

Currently, there are many IoT platforms available to choose from, one of them being Home Assistant. They provide an open source platform that users can use to strap onto a web server to attach an IoT device to. Currently, the packets sent from and to the web server are not secure as they are not encrypted with HTTPS. Meaning that if the device packets could be accessed, an intruder would have full control of your device and be able to work at obtaining access to other items on your network from there.

Hosting using HTTPS is the standard option that people and company's take to implement security to their IoT server. This brings many issues to the user hosting the service, one being that they would need to host with a dedicated IP address. Doing so brings anonymity concerns as people connecting to your resource will be able to obtain the IP. One attack that could be used after knowing this is a DDoS attack. After making the IP static the host needs a SSL certificate which is a task that takes time and money to complete. After setting up HTTPS successfully attackers will still be able to run port scanning on your network.



To install Home Assistant, run:

```
pip3 install homeassistant
hass --open-ui
```

Here are some resources to get started.

- Home Assistant website
- Installation instructions
- Troubleshooting your installation
- Configuring Home Assistant
- Available components
- Troubleshooting your configuration
- Ask community for help

Figure 2.5: Demo Splash screen for Home Assistant

Kaa

Kaa is an open source IoT platform that is hardware-agnostic(KaaIoT 2017). Almost any device can be implemented into the platform, starting from devices powered by operating systems, to resource-constrained micro-controllers. As Kaa is open source, anything that is not integrated with the platform can theoretically be implemented easily with the KAA endpoint SDKs. The same issues arise when looking at Kaa compared to Home Assistant. The solution by default is hosted using HTTP. As stated above this is something that should be avoided as any traffic to and from the server is un-encrypted. Meaning HTTPS is the next option to look at, and with the steps needed to implement HTTPS above, there are still downsides to using HTTPS over Tor.

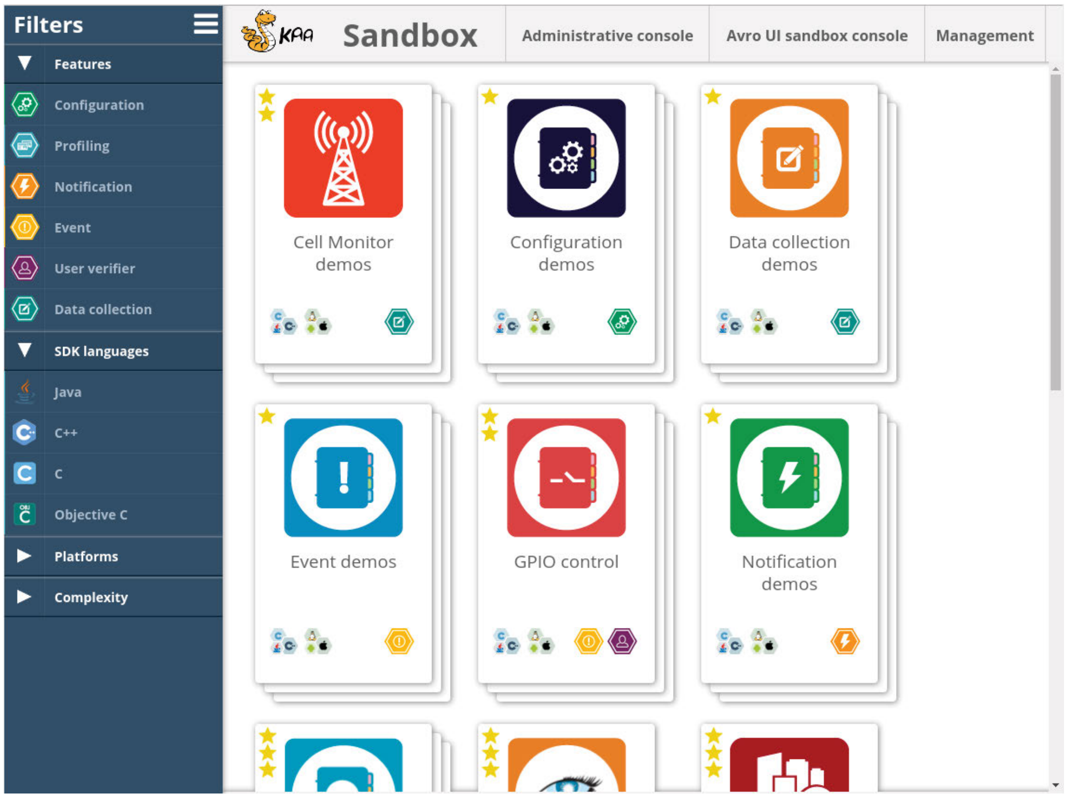


Figure 2.6: Demo Splash screen for Kaa

2.4 Tor as a Solution to IoT Security Threats

Tor is a well established, open source platform that has a proven effect in providing security and privacy in classical networks. As Tor is open source this makes it easier for researchers to look at Tor as a viable solution to a problem, and the product can be shared quickly. The end goal of the project is to produce a solution that uses Tor to drive an anonymous service to secure IoT users/applications. Establishing a security standard which uses the Tor network to house the IoT network would eliminate most issues that arise when implementing an IoT system.

Current solutions that are used are a compromise between speed and security. Using the Tor network will remediate the security issues by using end to end encryption. Also keeping anonymity by using relay nodes to mask the users IP address, what service they are hosting. Remote connections are secured by the same encryption and the nodes mask the incoming IP and what the user is trying to access.

Compared to the solutions that exist the Tor solution will negate most of the problems that they encounter. One issue that is eliminated is user/data privacy concerns, i.e., prevent anyone from accessing/scanning the IoT ecosystem ports and server fully, as they will not know where the service is being hosted. Hidden services can hide that they exist at all, if you do not know the correct authentication code. Therefore, you cant probe the service at all.

It is also useful to use Tor when not wanting to buy a HTTPS certificate and also do not know how to get an SSL/TLS certificate and how to correctly configure the service to run on HTTPS. There is also the added advantage of being able to access the service remotely without opening a port or setting up a virtual private network (VPN).

Chapter 3

IoT-ToR (IoToR) Solution Design and Analysis

The requirements of this project were to design and develop a Tor driven anonymous service to provide IoT users a safe platform to run their devices on. Currently, companies and single users buying IoT devices are using proprietary software for their device. Also, companies are stuck with designing proprietary software for their devices to connect to the Internet only for it to be insecure.

The solution proposed in this project is to use the Tor network to host a versatile web server that IoT consumers can attach their API to. This would make setting up a secure environment for anybody easier and more secure as the Tor network is encrypted so no information could be gained. In this chapter, the requirements analysis and IoToR design are given.

3.1 IoToR Requirements Analysis

There are multiple requirements that are needed to be met before the project can start. To start development on the IoTor I firstly need to pick the API/web server I wish to use. The platform Home Assistant are moving forward by focusing on auto-detection and implementation of many devices. Home Assistant already has support for many devices and the number is growing every day. Therefore making the process easier for the end user to set up common devices securely with the IoToR solution. Therefore I will be using Home Assistant in this project. I will then need an IoT device with a sensor that can record data to push to the web server. I will also need a device to host the server/API, this is so the hidden service can be locally hosted. As a lightweight device can be used on the same network to handle this, I will be using a Raspberri Pi model 3 b. To complete this project I will also need to gather information from a device to advertise on the hidden service. As this is the case an IoT device needs to be used to gather the information from the required sensor. For this I will be using an Elegoo UNO r3, and a standard sound sensor.

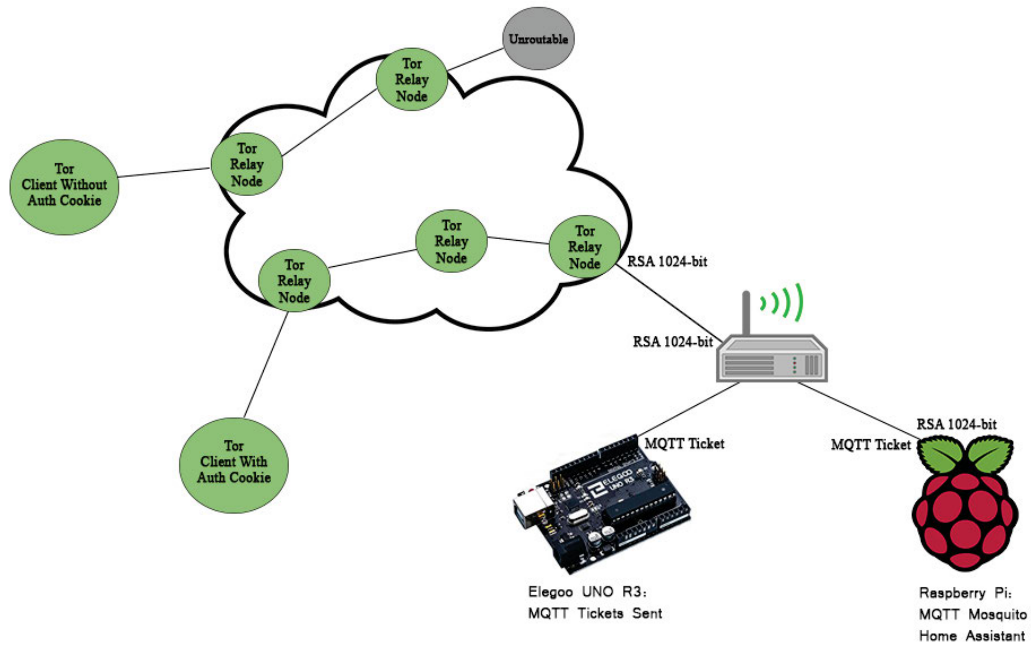


Figure 3.1: Design of interconnection of devices

As seen in Figure 3.1 the IoT device will be connected to the Raspberry Pi through the router. The information from the IoT device will be sent using the Message Queue Telemetry Transport(MQTT) protocol that runs on top of the TCP/IP protocol. With this a message broker is needed to handle MQTT tickets. The message broker is responsible for distributing tickets to interested devices based on the title of the ticket. Therefore I will be using the open source Mosquitto MQTT broker which will handle MQTT tickets over my network(Eclipse 2017). This will be triggered every time there is an event to send. The information is then aggregated so the API can use it to display information to the user. After the information has been aggregated and has been send to the API, the server will advertise this information to the hidden service over the Tor network. The hidden service is then protected by encrypting all packets with RSA 1024-bit encryption, which is an asymmetric encryption algorithm used by Tor(Roger Dingledine 2017).

We can then observe two clients connecting to the hidden service in figure 3.1. Both clients know the destination address for the service but only one had an authentication cookie for the service. Therefore the client without the authentication cookie will be re-directed to an unroutable address. With the authentic client being routed through the the service using Tor encryption.

3.2 Product Design Model

Whilst developing my proposed solution I will be using the Agile methodology for software development. Agile development is a set of principles for software development in which the solutions will evolve though collaborative efforts, I will be using this and adapting the solutions myself over time. Meaning after each step in implementation I will be evaluating and deciding if anything needs changing based on my prior research. This design model is suitable for my project as there could be issues where I would have to go back and review

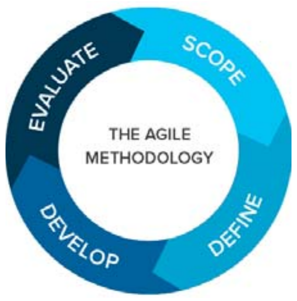


Figure 3.2: Agile software development methodology

Figure 3.2 shows how the agile methodology works conventionally.

3.3 IoToR Design

In this section I will be looking at the inter communication of the devices and services used. Once completed I will be able to use there as reference diagrams for how to set up the device physically, and the communication between the different aspects of the design.

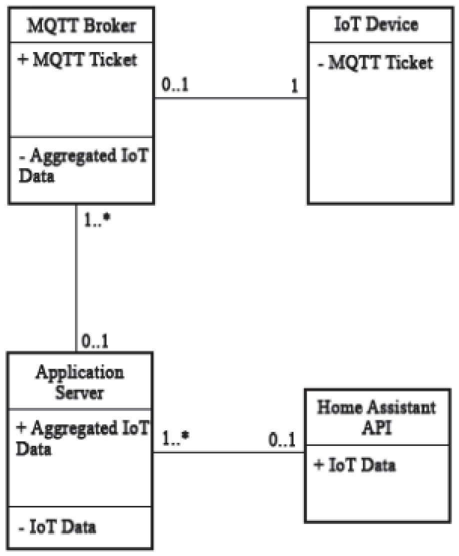


Figure 3.3: Design of communication between devices

In figure 3.3 we can see the design of how the products will communicate with each other to achieve the end goal of sending data to the API. We can see that the IoT device(Elegoo) will create an MQTT ticket that is then sent to the MQTT Broker. The broker will then take this information and aggregate it to get it ready to send down the line. but before it does if there is an issue with the ticket an error will be sent back to the device. If there are no errors then the title of the MQTT ticket is pulled to then define what to advertise. Then the ticket is advertised so anyone that is looking for a MQTT ticket with that header will receive the full ticket. In the case the application server sees a ticket with a header that it accepts it will accept the ticket. Then making the data readable to the API before sending it there to be displayed.

Table 3.1: Connecting Elegoo UNO R3 and Sound Sensor

Elegoo UNO	Sound Sensor
5V	+
GND	G
A0	A0
13	DO

In table 3.1 we can see how I have connected the Elegoo R3 to the sound sensor. The sound sensor needs to be connected to the `5V` out from the Elegoo into the sensors voltage in prong, which I have done by attaching it to the `+`. Then the ground for each of the devices needs to be connected. This is `GND` for the Elegoo and `G` for the sound sensor. The `A0` pins manage the analogue input for the device. The `DO` connects to the Elegoo's digital port 13.

Chapter 4

IoToR Implementation

In this section, I will be going through step by step how I produced my product. This will start with the configuration of the Raspberry Pi. In this section there will be many command denoted by the `$` symbol, this just denotes that the prompt is ready to take commands, therefore do not include this if implementing this specific solution. Firstly changing the configuration of the pre-installed OS. Then Installing dependencies for Home Assistant to run. I will then implement the solution as a hidden service before implementing the IoT device and sensor.

4.1 Raspberry Pi Development

4.1.1 Connecting to the Pi

Home Assistant requires the Raspberry Pi to be running the Raspbian Lite OS. The package bought for this project was a Raspberry Pi model 3B which came with a micro SD with the OS pre-installed onto. Once I connected the Pi to all the peripherals needed, I then needed to enable SSH so I could remotly connect to the Raspberry Pi for remote configuration.

As of November 2016 the SSH server is disabled by default so I had to go into go into the raspi-config (Raspberry Pi configuration command) to change this.

```
$ sudo raspi-config
```

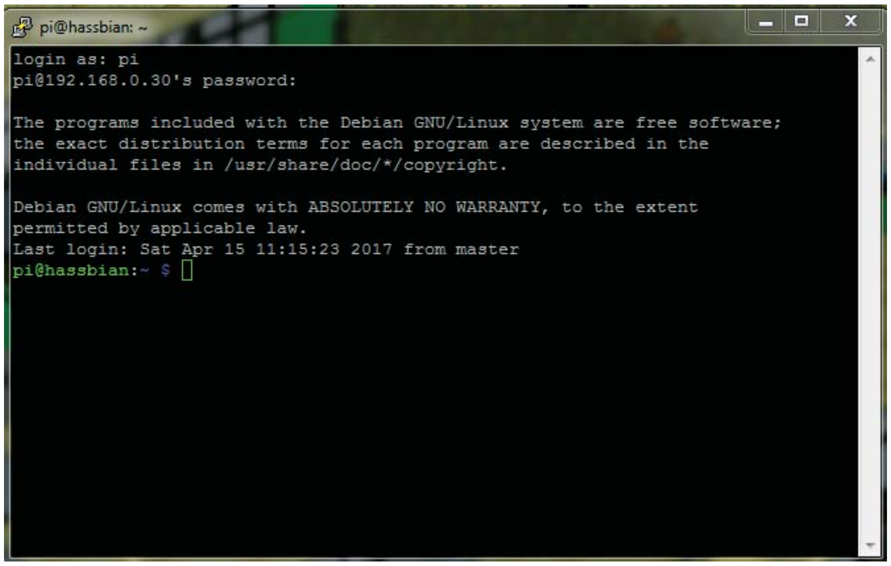
After running this command you will need to select `Interfacing Options`, then navigating to `SSH`, press `Enter` and select `Enable or Disable SSH server`. After doing this SSH is now ready to be used.
I now needed to know the Pi's IP address so I can SSH onto the Pi. To do this I will open up a terminal on my Pi and type:

```
$ hostname -I
```

Or alternatively by using a tool such as `nmap` on the network your pi is running on. As I now know what IP my Pi is on I can now connect using:

```
$ ssh pi@ipaddress
```

Replacing the `ipaddress` with the IP address that I obtained from the Pi. I personally used Putty to connect to the Raspberry Pi using the settings stated here. Once connecting to the Pi has been initialized there is a security warning for the first connection which I only needed to type `yes` to get to the next step. I then was prompt for a password which is `raspberry` on a fresh Raspbian Lite install. After connecting to the Pi a prompt will appear which is identical to the one found on the Raspberry Pi itself as seen in figure 4.1.



```
login as: pi
pi@192.168.0.30's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Apr 15 11:15:23 2017 from master
pi@hassbian:~ $
```

Figure 4.1: Connecting to the Raspberry Pi over SSH with putty.

It is then advised to change the default password by using the command bellow.

```
$ passwd
```

4.1.2 Installing Dependencies

Now to use the Pi I will need to install the dependencies needed for the operation of Home Assistant.

The first dependency that is needed is python so we will install that.

```
$ sudo apt-get install python3 python3-venv python3-pip
```

Now we will need to add an account for the Homr Assistant called `homeassistant`. I will also be using the -rm argument to create a system account and home directory for the user.

```
$ sudo useradd -rm homeassistant
```

The next step is to create a new directory to house the Home Assistant.

```
$ cd /srv
$ sudo mkdir homeassistant
```

Then the directory needs to be owned by the homeassistant user.

```
$ sudo chown homeassistant:homeassistant homeassistant
```

Next I will need to change to a virtual environment for Home Assistant, which will cause minimal overhead to the Raspberry Pi. Also the virtual environment will keep this Python environment isolated within a single directory tree. This means the Python package installed with Home Assistant will not interact with the rest of the system. With no interaction the system is isolated therefore no interference and no chance of the installation being broken by another program.

```
$ sudo su -s /bin/bash homeassistant
$ cd /srv/homeassistant
$ python3 -m venv .
$ source bin/activate
```

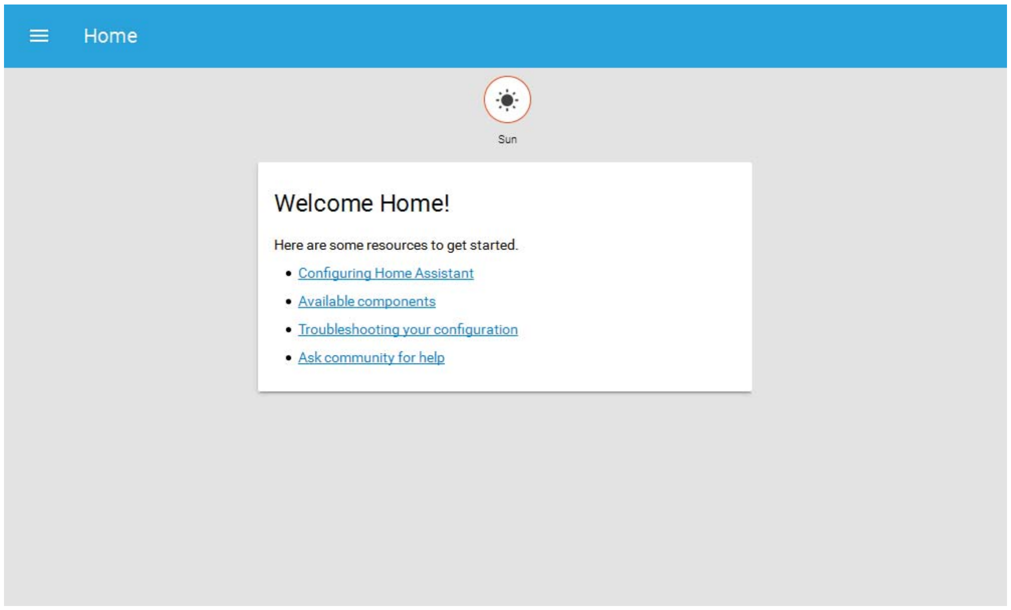
The prompt will then change to show us that the environment has been set up and is ready to use. I will then install Home Assistant.

```
(homeassistant) homeassistant@raspberrypi:/srv/homeassistant $ pip3
install homeassistant
```

Once installed Home Assistant can be run for the first time, which will complete the installation. This is because it will create a configuration directory `.homeasssistant` inside the `/home/homeassistant` directory and install any last dependencies.

```
(homeassistant) $ hass
```

Now that the dependancies have been installed and Home Assistant is now installed, the installation the the Raspberry Pi can be reached over a web interface on `http://ipaddress:8123`. Which looks like the image in figure 4.2



Welcome Home!

Here are some resources to get started.

- Configuring Home Assistant
- Available components
- Troubleshooting your configuration
- Ask community for help

Figure 4.2: Splash screen for first time connecting to Home Assistant.

4.1.3 Autostart

Having the Home Assistant service autostart on boot of the Raspberry Pi will be very useful to me and for anyone running a similar configuration. As I am running Debian I will be using systemd to autostart the Home Assistant instance. A service file is needed to be created to start the Home Assistant instance automatically using systemd. This will need to be implemented over SSH therefore we will be using the command line to create and save this file. Therefore we can use the following command to create the file.

```
$ sudo nano -w /etc/systemd/system/home-assistant@homeassistant.service
```

Then entering the following into the file before pressing CTRL-X then pressing Y to save and exit.

```
[Unit]
Description=Home Assistant
After=network.target

[Service]
Type=simple
User=%i
ExecStart=/srv/homeassistant/bin/hass -c "/home/homeassistant/.homeassistant"

[Install]
WantedBy=multi-user.target
```

Whereby the ExecStart is the path to the directory where Home Assistant was installed. Then I needed to reload `systemd` to make the environment aware of the changes.

```
$ sudo systemctl --system daemon-reload
```

After reloaded to get Home Assistant to start automatically I will use the following command.

```
    $ sudo systemctl enable home-assistant@homeassistant
```

Then finishing the configuration by using the `reboot` command to reboot the device.

4.1.4 Hosting as Hidden Service

Tor allows clients and relays to offer hidden services. As such we will be using Tor to provide our server with anonymity and security by advertising Home Assistant with Tor. Firstly I will need to install Tor onto the Raspberry Pi. This can be done easily with the following command.

```
    $ sudo apt-get install tor
```

Next I will need to configure the hidden service to point to the local web server. This is done in the `torrc` file which is generated when Tor is installed. Therefore I will navigate to `/etc/tor/torrc` and use nano to open the `torrc` file. Once open I will need to navigate to the section that starts with this:

```
############### This section is just for location-hidden services ###
```

This section of the file consists of hidden services, which are represented by groups of lines. By default all the code is commented out, as denoted by the `#`, therefore hidden services are disabled. There are two commands that initiate the hidden service that need to be present in the torrc file to start the hidden service.

- HiddenServiceDir: This points to the directory in which Tor will store information about that hidden service. Tor will also create a file in this directory called `hostname` which we will use later to gain the onion URL the service is hosted on.

- HiddenServicePort: This lets me specify which virtual port(the port people connecting to the hidden service will believe they are using) to redirect the traffic through, also stating the IP.

Using the template above we can implement the following code into the `torrc` to produce the hidden service.

```
HiddenServiceDir /var/lib/tor/homeassistant/
HiddenServicePort 80 127.0.0.1:8123
HiddenServiceAuthorizeClient stealth clientname
```

I have also added the last command, this ensures all traffic to and from the Home Assistance instance is run over the Tor network. This makes the traffic hidden to even other nodes on the Tor network. Also including a generic client name that can be modified. I will then restart Tor to produce a new authentication cookie from the `hostname` file.

```
    $ sudo /etc/init.d/tor restart
```

Now I can read the newly generated Tor authentication cookie from the `hostname` file using the following command.

```
    $ sudo more /var/lib/tor/homeassistant/hostname
```

With the output of the command being similar to the output on the following line. Although the generated .onion domain and authentication cookie will be unique to each hidden service. Unless you generate multiple cookies, in which case there would be multiple authentication cookies for one domain. Meaning that if a service that multiple users need to use is hosted as a hidden service, then multiple authentication codes can be generated. This can be used to manage access to the service in such environments.

```
abcdef1234567890.onion ABCDEF1122334455667789
```

After this the Home Assistant Tor configuration is complete. I can now input the new authentication cookie into the Tor client on a machine on my network. This will be to connect to the hidden service, and with this configuration I can access the Home Assistant instance over Tor on any network.

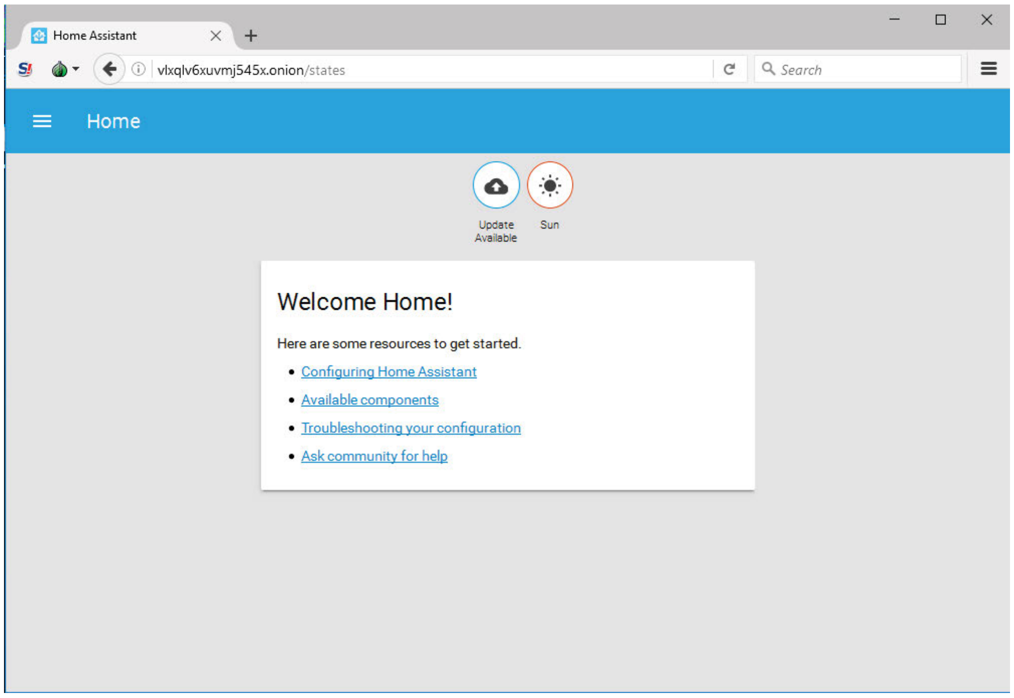
To do this I will need to browse to my `torrc-defaults` file which can be found in the tor browser install directory as follows:

```
"Tor install directory"/Browser/TorBrowser/Data/Tor/torrc-defaults
```

Where the "Tor install directory" is the directory in which you installed the Tor client onto the machine that you are trying to connect to the hidden service. Once in the file I have appended the authorization cookie to the file as follows:

```
HidServAuth abcdef1234567890.onion ABCDEF1122334455667789
```

After this I restarted the browser and browsed to the site to see it complete, as seen in figure 4.4.



Welcome Home!

Here are some resources to get started.

- Configuring Home Assistant
- Available components
- Troubleshooting your configuration
- Ask community for help

Figure 4.3: Splash screen for first time connecting to Home Assistant over Tor.

4.1.5 configuration.yaml

Home Assistant needs direction on how to handle information from devices to then push this to the server. The configuration.yaml file consists of all the code that builds the API. Meaning everything in this file is used to create the API. This can be found and edited on the Raspberry Pi, whilst in an SSH session by using the following commands:

```
cd /home/homeassistant/.homeassistant
sudo nano configuration.yaml
```

Home Assistant can also look for devices on your network and if it knows how to implement them into the API it will. This is useful for people that want to use the product with standard devices such as the Philips Hue, which would automatically be recognized. In this file I have specified an API password for the service. This is done using the `api_password` parameter, the splash screen for Home Assistant is now the API password screen, which is shown in figure ??. Also having `history` enabled is needed to track state changes over time.

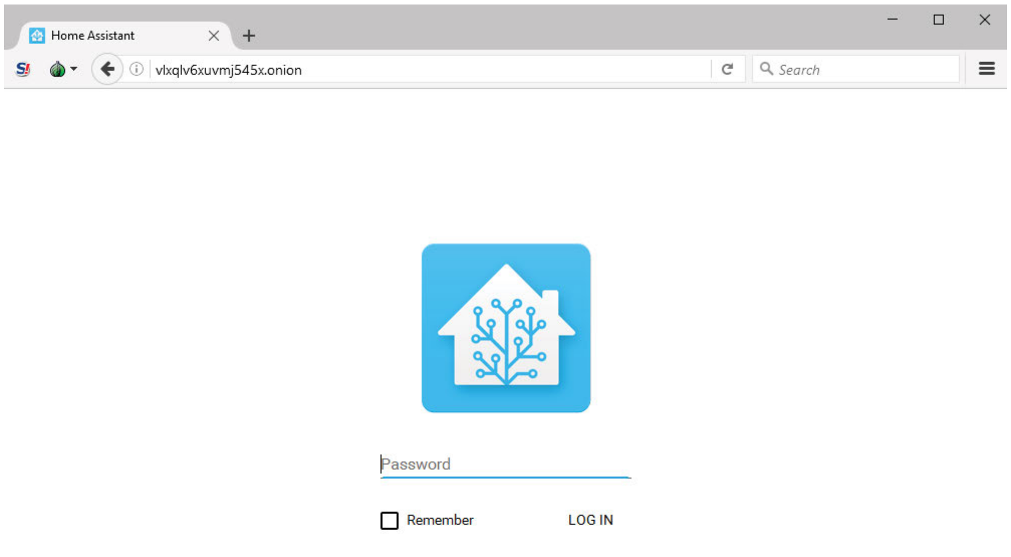


Figure 4.4: New API splash screen, generated using the api_password parameter.

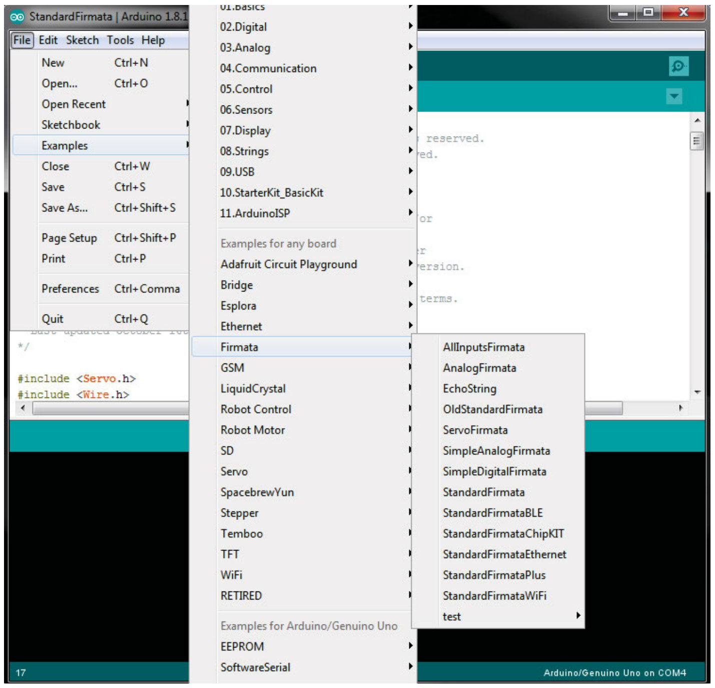
```
mqtt:
  discovery: true
  username: homeassistant
  password: api123

arduino:
  platform: mqtt
  state_topic: "sound"
  unit_of_measurement: "DB"
```

Using the code above I generate output from the device and push it to the API. The `mqtt` on the first line denotes the MQTT broker, with corresponding parameters tabbed on the following lines. using `discovery` will look for MQTT tickets that the server can process. As the brokers is responsible for distributing messages to the API, the broker will need the username and password associated with the API.

Also I have input information about the Arduino clone, whereby the `platform` is the protocol in which the device will be using to communicate. The `state_topic` parameter denotes the topic to look for to store that with the `unit_of_measurement` parameter.

4.2 Elegoo Implementation

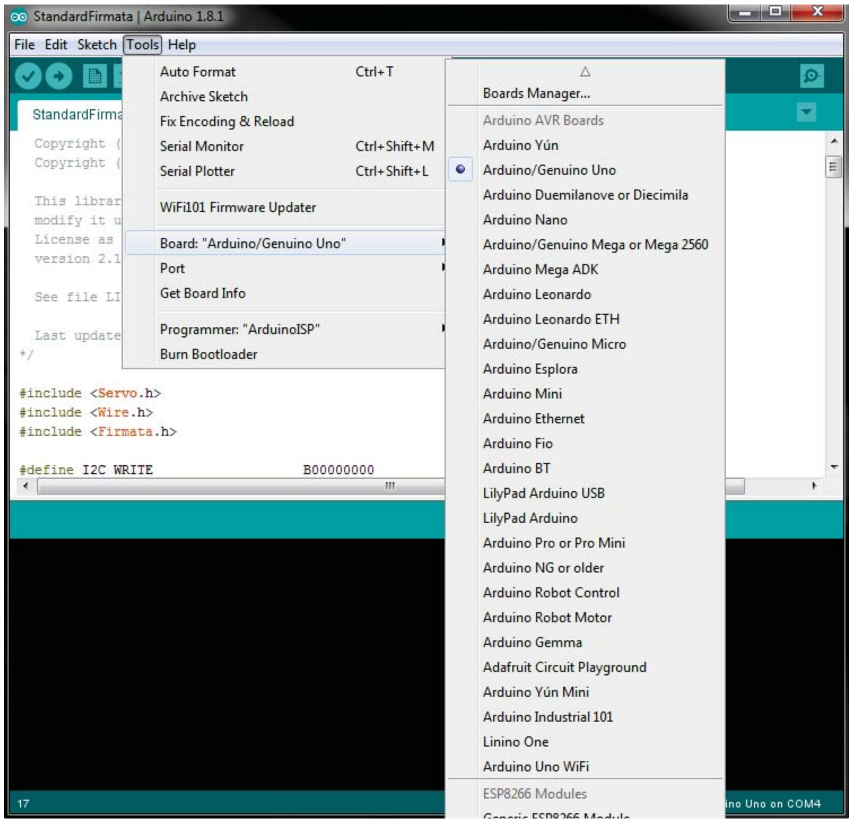
As stated in the design I am using an Elegoo R3 which is comparable to the Arduino Uno. Also using a sound sensor to read in sound levels over time(in decibels). Firstly I will need to ready the Elegoo R3 for passing information from the Device to the server. To pull the data from the sound sensor I have used the StandardFirmata firmware on the board, this firmware is designed to selectively send and receive data to and from the sound device. Using the Arduino software to upload the firmware to the board. As we can see in figure 4.5 I have opened the software and navigated to the location of the sketch I will be using.



```
*/

#include <Servo.h>
#include <Wire.h>
```

Figure 4.5: Using Arduino software to upload firmware to Elegoo R3.

Once opened I can then plug the Elegoo into my machine. Once the device has been recognised and drivers have been downloaded we can see which COMM port it is using by checking in `Tools > Port`. Then choosing the correct port and board type before uploading the firmware. The settings can be seen in figure.


StandardFirmata | Arduino 1.8.1

Board: "Arduino/Genuino Uno"

Programmer: "ArduinoISP"

```
  Copyright (
  Copyright (

  This librar
  modify it u
  License as
  version 2.1

  See file LI

  Last update
*/

#include <Servo.h>
#include <Wire.h>
#include <Firmata.h>

#define I2C_WRITE                   B00000000
```

Figure 4.6: Selecting settings in Arduino application ready for firmware upload.

Before uploading the firmware on the Arduino application, it is required that the Elegoo be restarted. Therefore I will restart the device then upload the firmware.

Once uploaded I can reconnect the Elegoo R3 along with the sound sensor back to the the Raspberry Pi to power the device.

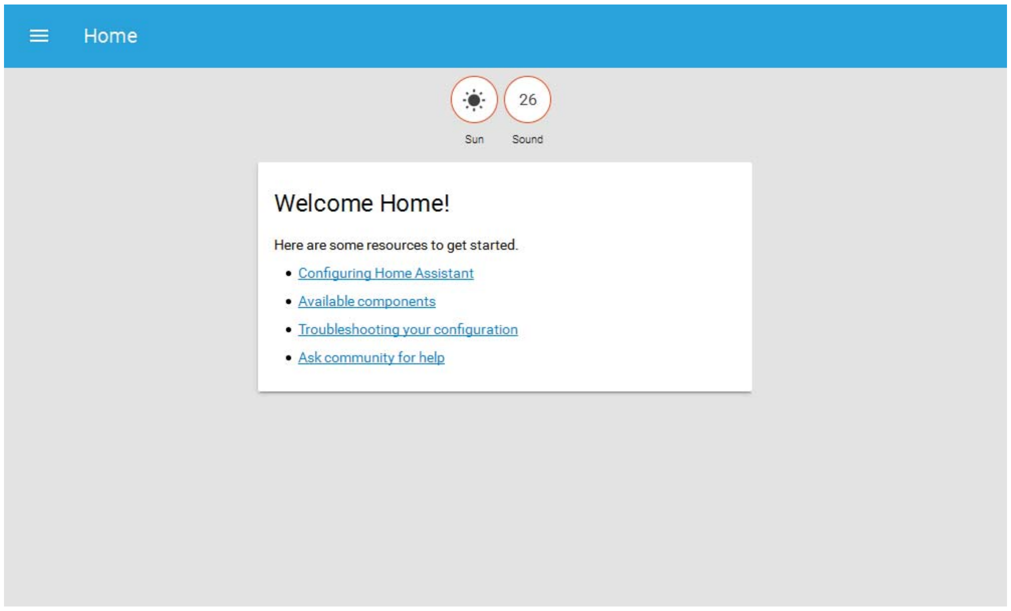
4.3 IoToR

Now that the product has been implemented I will check the functionality to make sure there are no errors. In the case that the product is not showing error logs can be looked at to see what is happening. In the `configuration.yaml` section there are issues that can arise that cause the product not to show. A quick way to check for errors from that file is to use the following:

```
hass --script check_config
```

Which will output any issues Home Assistant is having whilst processing the file.

The splash screen for the device can be seen in figure 4.7. We can now see the sound device showing on the splash screen.



Welcome Home!

Here are some resources to get started.

- Configuring Home Assistant
- Available components
- Troubleshooting your configuration
- Ask community for help

Figure 4.7: Home Assistant splash screen with new sound device.

If we click the device a box will appear showing the last 24 hours for the device. Showing the change in sound over time for the device as seen in figure ??.

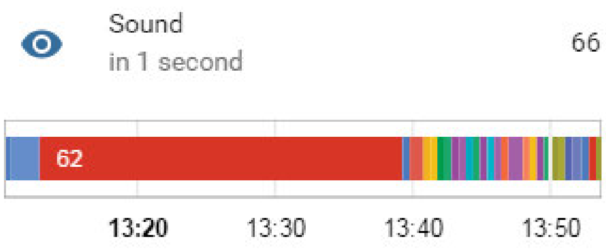
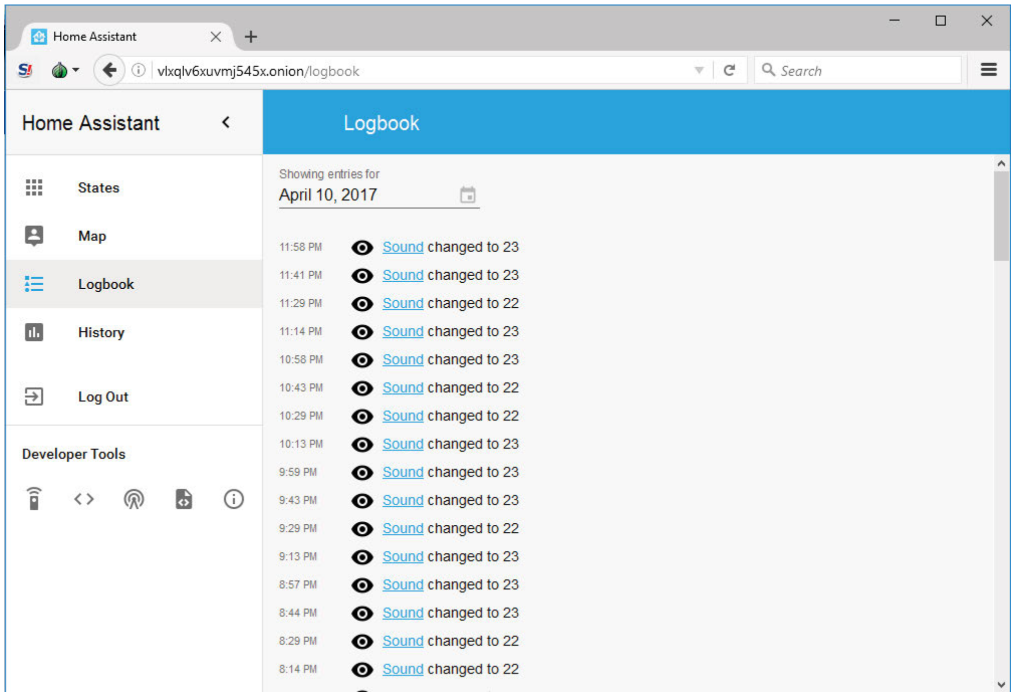


Figure 4.8: Home Assistant showing the values for sound over time for the device.

In this specific case this information is not that useful as the sound over time changes so much that it is difficult to distinguish between the sound and time. There are `History` and `Logbook` tabs that can be explored in Home Assistant to see the data represented differently as seen in figure 4.9.



Showing entries for
April 10, 2017

11:58 PM	Sound changed to 23
11:41 PM	Sound changed to 23
11:29 PM	Sound changed to 22
11:14 PM	Sound changed to 23
10:58 PM	Sound changed to 23
10:43 PM	Sound changed to 22
10:29 PM	Sound changed to 22
10:13 PM	Sound changed to 23
9:59 PM	Sound changed to 23
9:43 PM	Sound changed to 23
9:29 PM	Sound changed to 22
9:13 PM	Sound changed to 23
8:57 PM	Sound changed to 23
8:44 PM	Sound changed to 23
8:29 PM	Sound changed to 22
8:14 PM	Sound changed to 22

Figure 4.9: Home Assistant Logbook showing values for the device.

In figure 4.9 we can see the change over time represented in rows, filtered by the date.

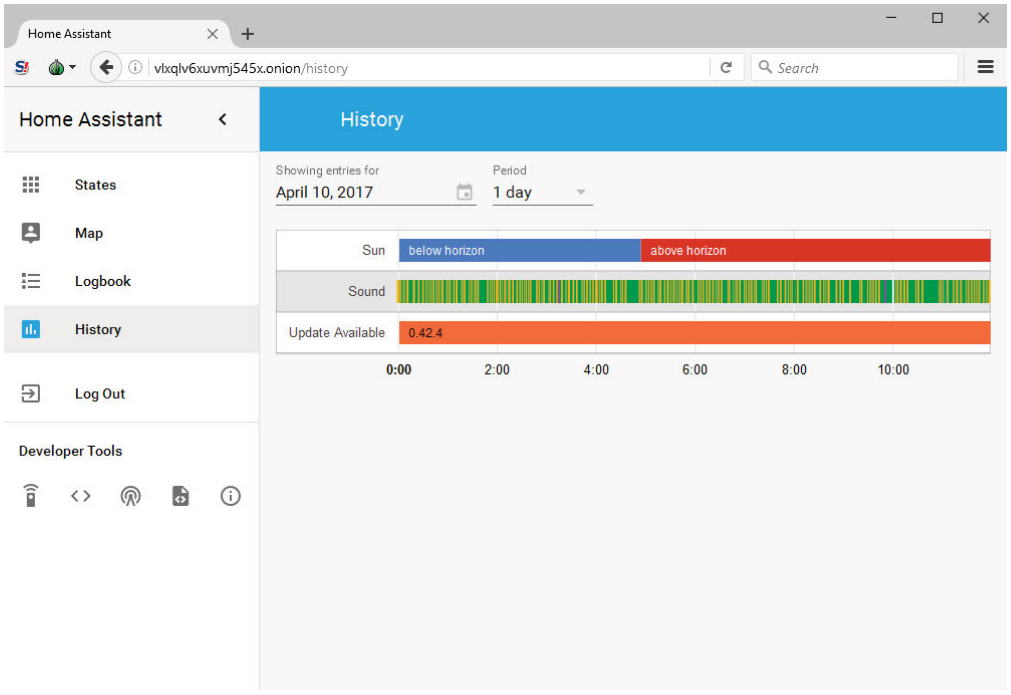


Figure 4.10: Home Assistant History showing values for the device.

In figure 4.10 we can also see the sound change over time for the device. On this specific date the sound input for the device fluctuates between a few values with a few anomalies from when I entered the room with the device in.

4.4 Testing

Now that the product has been implemented I can look at different aspects of the Tor network and measure these to evaluate the product. This will be done predominantly looking at the performance of the Tor network and comparing this to the open web. With Tor routing all traffic through nodes it can be seen that Tor runs slower than using the open web with a standard browser. There may be latency issues whilst connecting to services, which could turn away potential IoT users from routing their service using Tor. Using the Tor Metrics project I have been able to run various tests on the Tor network(TorMetrics 2017). Tor use data from various sources to produce this data, one main source being CollecTor. This is a Tor driven service which collect various information about nodes on the network.

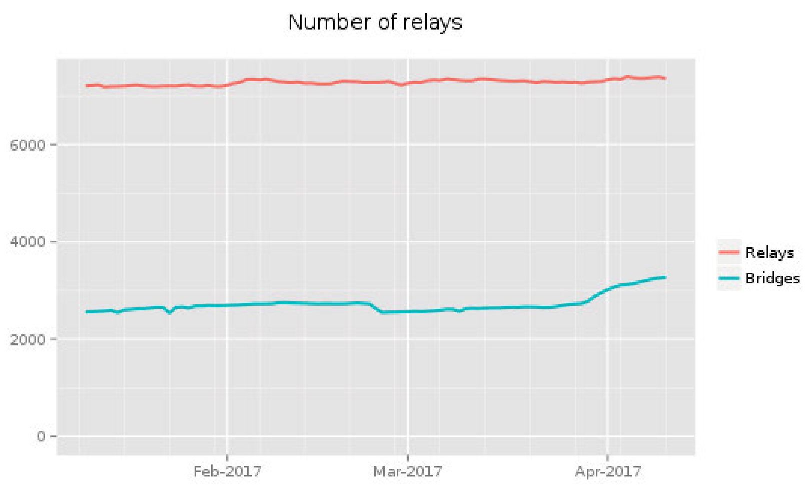


Figure 4.11: Tor metrics showing the number of running relays and bridges in the network(TorMetrics 2017).

In figure 4.11 we can see the number of relay nodes and bridges on the Tor network. Whereby bridge's are relay nodes that are private therefore they can be used to provide access for blocked clients. This graph shows the data from $10-01-2017$ to $10-04-2017$. The number of relays in 4.11 stays above $6,500$ nodes at the time the graph was produced. A standard interaction on Tor takes only 3 nodes to transfer the data through.

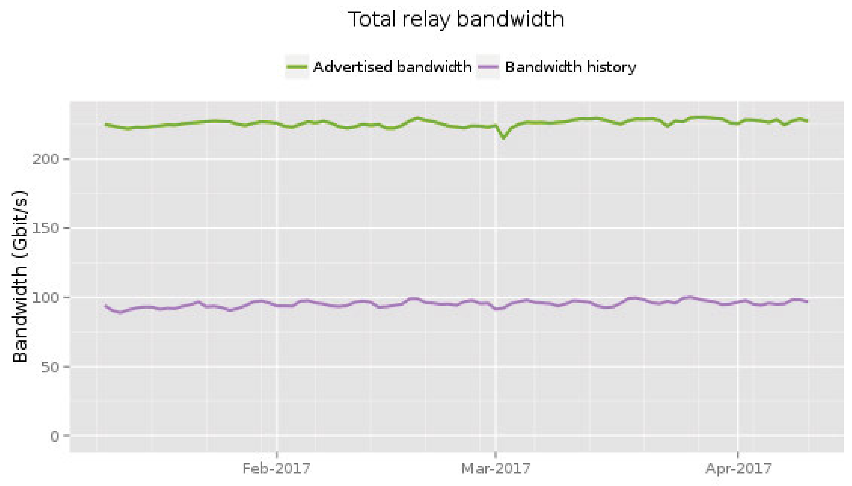


Figure 4.12: Tor metrics showing the bandwidth of running relays in the Tor network(TorMetrics 2017).

In figure 4.12 we can see that the advertised bandwidth is more than double that of the consumed bandwidth of all relays. Meaning that those 6, 500 relay nodes can withstand the pressure applied to the Tor network. Therefore this is not an issue for users hosting services over Tor.

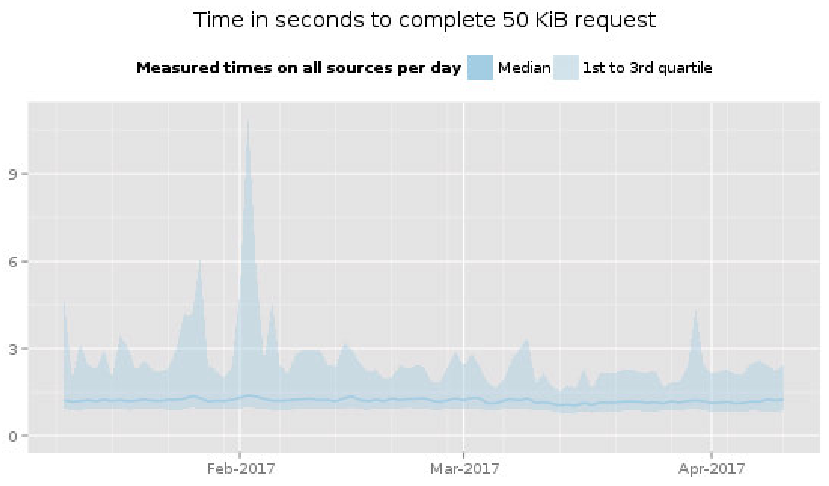


Figure 4.13: Tor metrics showing the time in seconds to complete a 50KB request(TorMetrics 2017).

The main issue raised by people when using Tor for security is for it's high latency. Looking at figure 4.13 we can see the overall performance when downloading files of size $50KB$. For this Tor have only taken the $1st$ to $3rd$ quartile, therefore omitting the slowest and fastest quarter of the measurement. We can see from this graph that the time taken ranges from 1 second to 10.5 seconds, with the median being stable at just above 1 second. In regards to latency anything in the range $5 - 40ms$ is considered normal for the open web, meaning that Tor on average is twice as slow compared to the open web (PingMan 2014).

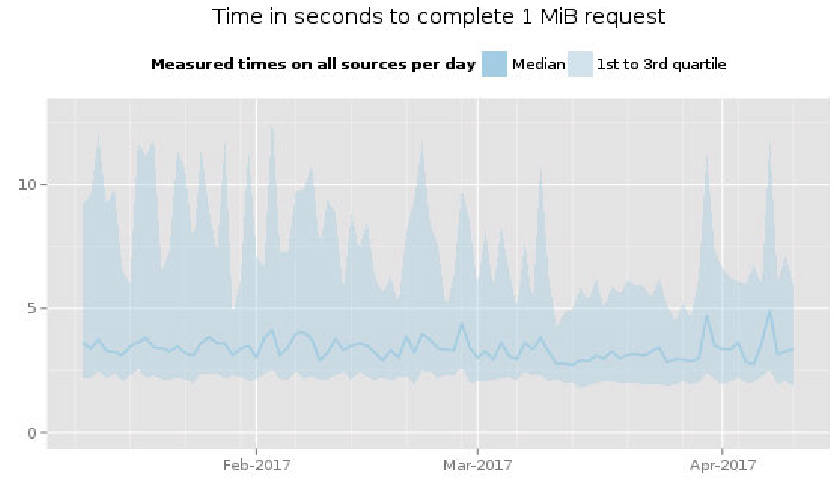


Figure 4.14: Tor metrics showing the time in seconds to complete a 1MB request(TorMetrics 2017).

We can also look at the time in seconds to complete a $1MB$ request as shown in figure 4.14. The range for this is just under 2.5 seconds to 15 seconds, with the median fluctuating between 2.5 seconds and 5 seconds. Looking at the graph we can also see that the distribution for the time taken is a lot more erratic than that of what can be observed using a significantly smaller file. As seen in figure 4.13, which is to be expected. As the Tor solution runs off light weight MQTT tickets, there wont be many instances where this will be an issue in this case. Although for users with larger data sets being sent over the network there could be an issue.

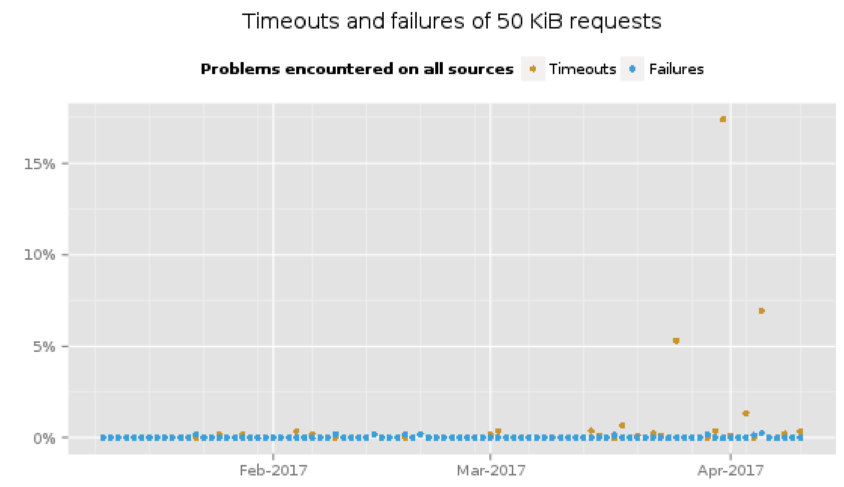


Figure 4.15: Tor metrics showing the time in seconds to complete a 1MB request(TorMetrics 2017).

In figure 4.15 we can see a graph which plots the fraction of timeouts and failures when downloading a $50KB$. A timeout occurs when a download does not complete in the time allocated, and a Failure occurs when the download completes, but the file retrieved is smaller than expected. There are 20 days out of the 90 days of data where timeouts have happened on files, and 9 days where there where failures. In regards to the timeouts the majority of them effected are negligible, as the amount of files on the day the files timed out was below 2%. Although there are 3 anomalies which show a range between 5% and 17.5%. But as the data is taken from 90 days, having 3 anomalies with an average of 9.8% of files timing out over 3.3% of the total days tested is a small amount for the Tor network. As the failures only happen to less than 1% of the data used this can be seen as negligible.

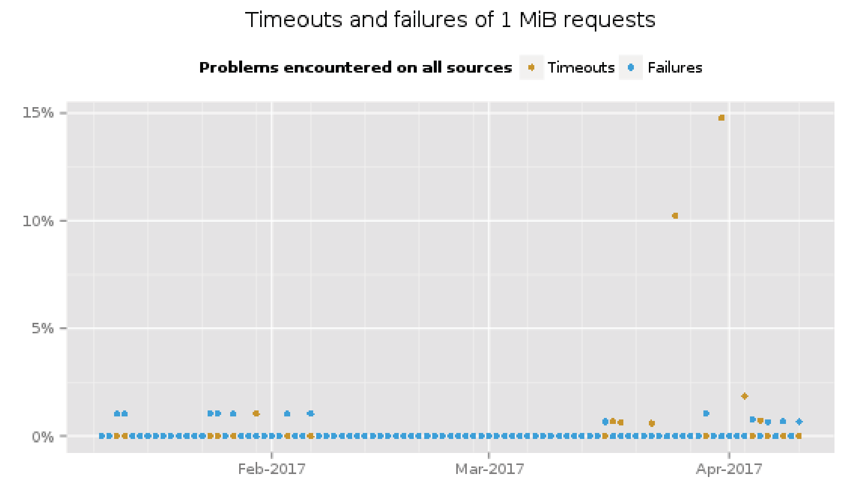


Figure 4.16: Tor metrics showing the time in seconds to complete a 1MB request(TorMetrics 2017).

I have then looked at the fraction of timeouts and failures on files that have a size of $1MB$. The results for the timeouts are close to those achieved by changing the size of the file to $50KB$, as seen in figure 4.15. As there are 2 anomalies that range from just over 10% to just under 15%. The failures in this case occur a lot more frequent on the larger file, as there are 12 readings above 0% all around 1% as seen in figure 4.16. This only shows that larger files are more prone to failure in transmission, but as the increase in failure is 1% of the data used, and this being over 12 days out of 90 this is negligible.

Chapter 5

Evaluation

At the beginning of my project there were specific aims and objectives to complete. In this section I will be evaluating the completeness of the product by critically analyzing the work and results achieved.

5.1 IoToR

Firstly I will evaluate the product that was produced for this project against the aim and objectives. The aim for the project was to produce a Tor driven anonymous service to secure IoT users and applications. It also states that the solution will abstract IoT devices and applications into hidden services, that can then be accessed securely and anonymously. The product produced meets these criteria completely as I have an IoT device set up to transfer information to a hidden service running over the Tor network. It was also stated that the solution must not modify the underlying architectures or processes. This is also the case as the product implements the hidden service on top of completed product.

One objective that I needed to meet was to conduct a literature review, to then identify the common critical IoT security threats. This objective was met as this has been produced, focusing on the common critical threats. I then investigated the current platforms that provide security to IoT devices. Therefore meeting my second objective. This was done by critically analyzing some current solutions to then implement the Tor solution on this platform. Bringing me to my next objective which was to identify an open source platform, suitable to extend to provide a hidden service solution to. I also believe I met this as the implementation used Home Assistant, which is open source, and suitable to implement hidden services with. The next objective was to implement a plugin to allow users to connect their own devices with their own hidden service. As the Tor solution differs per install, I do not believe a plugin can be developed to implement a wide variety of products as a hidden service. Although I believe a standard install package could be developed, which would be system independent, which would set up default configuration for my product. Meaning there would be no added information about the IoT devices inside the configuration.yaml file, but it would be set up ready to take input. This would mean that anyone wishing to implement an IoT device into their home/business would be able to do so quickly, with added security, and anonymity. The next objective was to deploy the server infrastructure needed to implement the Tor solution to the chosen open source platform, into a local machine. The implementation of the chosen platform has been completed, as I am hosting the Home Assistant platform. It is also hosted on a local machine on my network on a Raspberry Pi model 3. The last objective being to evaluate this product which is being covered in this section.

5.2 Testing

For the testing I can deduce that the network is stable in terms of number of relays and bandwidth allocation to the relays. This is simply because the amount allocated is far greater than that of those resources needed. Looking now at the latency of the network when downloading a file of 50KB, we can see that the mean is double that of the open web latency. This is to be expected as the process through Tor's network will be higher due to the amount of encryption happening. Also having to pass through random nodes around the world will be effecting the speed at which Tor can process the $50KB$ file. The difference of 0.5 seconds will not effect any home use of this system, and only effect the most time strenuous implementations of this product. By looking at the data from Tor metrics using the same test but with a $1MB$ file instead, we can see that the median time has increased by an average of 3.25 seconds. Looking at the data spread we can conclude that larger files produce a more erratic distribution, meaning this could be unsuitable for time critical applications e.g. health care. Although this is only the case if the information transmitted has to be received withing a time frame that is less than the range, and if the product uses files above $1MB$ often.

Looking at the results from the timeouts and failures of a $50KB$ request, we can see that there were 3 anomalies for timeouts that average 9.8%. Although as this only occurs during 3.3% of the total days from the dataset we can deduce that this is an acceptable amount of timeouts. Comparing this to the same test but with a $1MB$ file we see around the same distribution but with more failures. This is to be expected as the file size is larger therefore there is a greater chance of failure. And as failure on increase by 1% for 12 day's out of 90 the results are negligible.

Chapter 6

Conclusion

This project was undertaken to produce a Tor driven anonymous service to secure IoT users and applications. The results of this project revealed that the IoToR solution would be beneficial for home and business use. Although Tor have made efforts to speed up the latency of the network, the IoToR solution may not be suitable for applications that are time sensitive. This is due to the nature of Tor taking longer to send packets than using the open web. In regards to future work, testing the solution with a wider range of devices would be a starting point. This would give a clearer indication as to what extent this product can be used.

Bibliography

[1] Ailanthus. *A Quick, Simple Guide to Tor and the Internet of Things (So Far)*. 2016. URL: `https://blog.torproject.org/blog/quick-simple-guide-tor-and-internet-things-so-far`.

[2] Audriusa. *Recorded by student in ETH (Zurich) during system security laboratory work. - Own work, GPL*. 2016. URL: `https://commons.wikimedia.org/w/index.php?curid=9762292`.

[3] Sachin Babar et al. "Proposed embedded security framework for internet of things (iot)". In: *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*. IEEE. 2011, pp. 1–5.

[4] Alex Biryukov and Eyal Kushilevitz. "From differential cryptanalysis to ciphertext-only attacks". In: *Annual International Cryptology Conference*. Springer. 1998, pp. 72–88.

[5] Franco Callegati, Walter Cerroni, and Marco Ramilli. "Man-in-the-Middle Attack to the HTTPS Protocol". In: *IEEE Security and Privacy* 7.1 (2009), pp. 78–81.

[6] Eclipse. *Mosquitto*. 2017. URL: `https://mosquitto.org/`.

[7] LORENZO FRANCESCHI-BICCHIERAI. *Blame the Internet of Things for Destroying the Internet Today*. 2016. URL: `http://motherboard.vice.com/read/blame-the-internet-of-things-for-destroying-the-internet-today?utm_source=vicenewsfb`.

[8] Paul Fremantle. *A Reference Architecture For The Internet of Things*. 2016. URL: http://wso2.com/whitepapers/a-reference-architecture-for-the-internet-of-things/.

[9] Alan Grau. *What is Really Needed to Secure the Internet of Things?* 2016. URL: http://www.iconlabs.com/prod/internet-secure-things-%E2%80%93-what-really-needed-secure-internet-things.

[10] Giovanni Heward. *Cryptanalysis and Attacks*. 2014. URL: https://www.experts-exchange.com/articles/12460/Cryptanalysis-and-Attacks.html.

[11] KaaIoT. *Kaa IoT Development Platform overview*. 2017. URL: https://www.kaaproject.org/overview/.

[12] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential power analysis". In: *Annual International Cryptology Conference*. Springer. 1999, pp. 388–397. URL: https://goo.gl/jg0zLx.

[13] Rob van der Meulen. *Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015*. 2015. URL: http://www.gartner.com/newsroom/id/3165317.

[14] Oliver Niehus. *My Top 3 Trusted Boot: Secure Boot Measured Boot*. 2013. URL: https://blogs.msdn.microsoft.com/olivnie/2013/01/09/windows-8-trusted-boot-secure-boot-measured-boot/.

[15] PingMan. *What is normal for latency and packet loss?* 2014. URL: https://www.pingman.com/kb/42.

[16] Wind River. *SECURITY IN THE INTERNET OF THINGS*. 2013. URL: https://www.windriver.com/whitepapers/security-in-the-internet-of-things/wr_security-in-the-internet-of-things.pdf.

[17] Nick Mathewson Roger Dingledine. *Tor Protocol Specification*. 2017. URL: https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt.

[18] Nick Sullivan. *DDoS Prevention: Protecting The Origin*. 2016. URL: `https://blog.cloudflare.com/ddos-prevention-protecting-the-origin/`.

[19] TheTorProject. *Tor Overview*. 2016. URL: `https://www.torproject.org/about/overview.html.en`.

[20] TorMetrics. *Tor Metrics*. 2017. URL: `https://metrics.torproject.org/`.

[21] Kim Zetter. *Logic Bomb Set Off South Korea Cyberattack*. 2013. URL: `https://www.wired.com/2013/03/logic-bomb-south-korea-attack/`.

Securing the Internet of Things using Tor Hidden Services, and Anonymity

[redacted] [redacted]

Contents

Terms of Reference

1 Project Background

The Internet of Things (IoT) is the networking and remote control of every day devices, ranging from a fridge or a family surveillance system to a whole building's central heating and ventilation system. The concept of IoT and the communication and networking of devices has been discussed as early as 1982. The first to implement this was a Coke machine that was able to report its inventory and if new stock in the machine was cold yet or not. Using IoT you are opening yourself up to a great deal of vulnerabilities, including the opportunity for surveillance from an entity to gain access and exploit private information against individuals or companies. This is far from hypothetical as attackers stole 40 million credit card numbers after they hacked into a national retailer's HVAC system and used it to reach their computer system and their customers (Ailanthus 2016). Implementing IoT over Tor is a viable solution for added security features as Tor encrypts traffic of the network the IoT web server is hosted on.

Tor was developed in the mid 1990's by the United States Naval Research Laboratory, with the purpose of protecting U.S. intelligence communications online. In 2004 the Naval Research Laboratory released the code for Tor under a free license. After development for a few years in 2006 The Tor Project was founded, which is a research and education, non profit organization responsible for maintaining Tor. Since then network anonymity has led many people to start using the Tor browser to navigate the "hidden service" found using Tor. A hidden service is just a normal service that could be offered on the web, but if you don't have the URL you wont be able to see it e.g. web publishing, instant messaging server. The hidden service can make it's self available by advertising it's existence to the Tor network. By contacting random relay nodes on the Tor network the service will now be available to view if you have the URL to view it on. "Using Tor protects you against a common form of Internet surveillance known as "traffic analysis."" (TheTorProject 2016) This is a main benefit of using Tor, as . Using Tor, the network traffic is transferred between hundreds of volunteer-operated relays. As this data is encrypted, none of the volunteer nodes hold two crucial bits of information that could be used to track who is searching for what. The two crucial bits of information being; the source of information, and the destination of the information. Virtually removing your footprint from every node connecting you to the destination server.

This is put into practice by Tor in a smart way. Tor will obtain a list of Tor nodes from a directory server, then will build up a circuit of encrypted connection through relay nodes on the network. Put into practice this will make an "anonymous network". This is because between you and the site you are trying to access, if anyone were to intervene the network traffic they would only see parts of the information transmitted from you, on any given Tor relay.

In conclusion I will create a web server with a device connected to it, and publish it as a hidden service. Meaning that the data is encrypted by Tor making it harder for people to exploit if they find it. This would be beneficial for anyone to use ranging from big companies to small families to protect their devices/information.

2 Course Specific Learning Outcomes

In doing this project I will be developing on, and gaining more understanding with:

- Studying the management and security of networked systems.
- Evaluating existing work and my own work.

- Studying the fundamentals of computer network communications and communication protocols.
- Developing a knowledge of Tor, IoT, and the anonymous network.

3 Aim

The aim of this project is to produce a Tor driven anonymous service to protect the users ”Internet of Things” application.

4 Objectives

During my project my objectives will be:

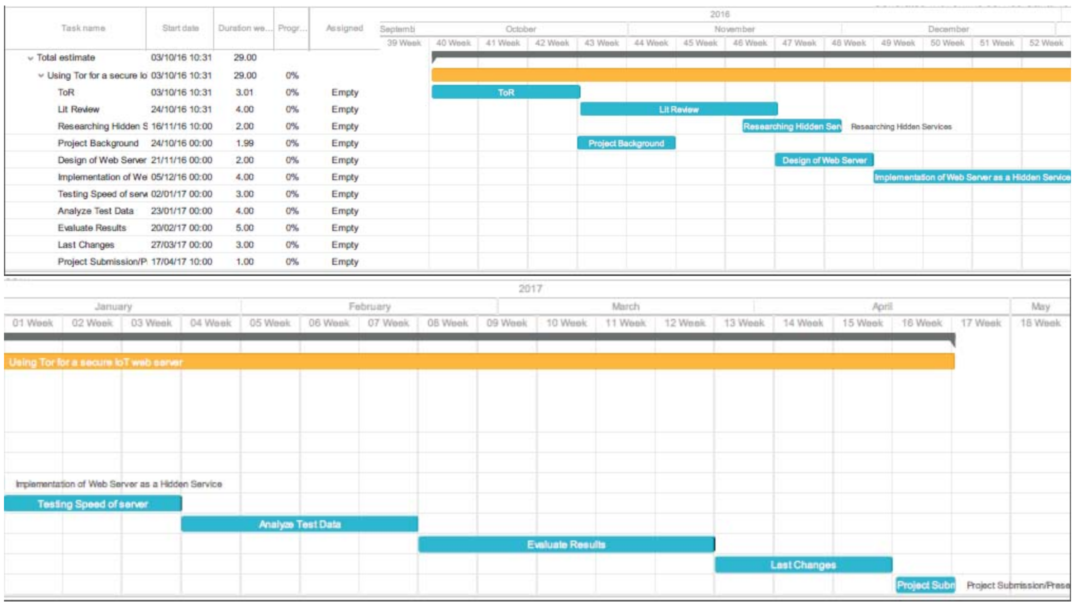
- Conduct a literature review to identify common critical IoT security threats that can be addressed using hidden services.
- Investigate current tools/platforms to provide security to IoT devices using Tor.
- Identify an open source IoT connection-focused platform suitable to extend to provide hidden services to the IoT ecosystem.
- Implement a plugin to allow users to integrate their devices with their own hidden application or service.
- configure and deploy the server infrastructure into a local machine
- Evaluate the new hidden service using real IoT objects.

5 Problems

During my project it is likely that I could encounter a problem, the problems/solutions are outlines below:

- The one problem I can see happening is if I can’t get the web server up and running on my network as a hidden service. If this happens I will need to use university resources to get around this.

6 Timetable and Deliverables



Task name	Start date	Duration we...	Progr...	Assigned
Total estimate	03/10/16 10:31	29.00		
Using Tor for a secure Io	03/10/16 10:31	29.00	0%	
ToR	03/10/16 10:31	3.01	0%	Empty
Lit Review	24/10/16 10:31	4.00	0%	Empty
Researching Hidden S	16/11/16 10:00	2.00	0%	Empty
Project Background	24/10/16 00:00	1.99	0%	Empty
Design of Web Server	21/11/16 00:00	2.00	0%	Empty
Implementation of We	05/12/16 00:00	4.00	0%	Empty
Testing Speed of serv	02/01/17 00:00	3.00	0%	Empty
Analyze Test Data	23/01/17 00:00	4.00	0%	Empty
Evaluate Results	20/02/17 00:00	5.00	0%	Empty
Last Changes	27/03/17 00:00	3.00	0%	Empty
Project Submission/P	17/04/17 10:00	1.00	0%	Empty

7 Required Resources

In terms of Hardware, I will need a wireless communication device that I can connect to IoT, a desktop that I can work from, and a network that I can publish the web server from.

References

Ailanthus (2016). *A Quick, Simple Guide to Tor and the Internet of Things (So Far)*. URL: `https://blog.torproject.org/blog/quick-simple-guide-tor-and-internet-things-so-far`.

TheTorProject (2016). *Tor Overview*. URL: `https://www.torproject.org/about/overview.html.en`.

ETHICS CHECKLIST

This checklist must be completed **before** commencement of **any** research project. This includes projects undertaken by **staff and by students as part of a UG, PGT or PGR programme**. Please attach a Risk Assessment.

Manchester Metropolitan University

Please also refer to the University's Academic Ethics Procedures; Standard Operating Procedures and the University's Guidelines on Good Research Practice

Full name and title of applicant:	[redacted]
University Telephone Number:	
University Email address:	[redacted]
Status: All staff and students involved in research are strongly encouraged to complete the Research Integrity Training which is available via the Staff and Research Student Moodle areas	Undergraduate Student ✓ Postgraduate Student: Taught ☐ Postgraduate Student: Research ☐ Staff ☐
Department/School/Other Unit:	Computing, Mathematics & Digital Technology
Programme of study (if applicable):	Computer Science
Name of DoS/Supervisor/Line manager:	Dr Mohammed Hammoudeh
Project Title:	Securing the Internet of Things using Tor Hidden Services, and Anonymity
Start & End date (cannot be retrospective):	26/10/2016 to 28/04/2017
Number of participants (if applicable):	1
Funding Source:	

Brief description of research project activities (300 words max):

In my project I will be identifying common critical IoT security threats that can be addressed using hidden services. Investigating current tools/platforms to provide security to IoT devices using Tor. Identifying an open source to extend to provide hidden services to the IoT ecosystem. Implementing a plugin to allow users to interrogate their devices with their own hidden application or service. Then to configure and deploy the server infrastructure into a local machine. And finally evaluate the hidden service using real IoT objects.

	YES	NO
Does the project involve NHS patients or resources? If 'yes' please note that your project may need NHS National Research Ethics Service (NRES) approval. Be aware that research carried out in a NHS trust also requires governance approval. Click here to find out if your research requires NRES approval Click here to visit the National Research Ethics Service website To find out more about Governance Approval in the NHS click here	☐	✓
Does the project require NRES approval?	☐	✓
If yes, has approval been granted by NRES? Attach copy of letter of approval. Approval cannot be granted without a copy of the letter.	☐	✓

NB Question 2 should only be answered if you have answered YES to Question 1. All other questions are mandatory.	**YES**	**NO**
1. Are you are gathering data from people?	☐	✓
For information on why you need informed consent from your participants please click here		
2. If you are gathering data from people, have you:	☐	☐
a. attached a participant information sheet explaining your approach to their involvement in your research and maintaining confidentiality of their data?	☐	☐
b. attached a consent form? (not required for questionnaires)	☐	☐
Click here to see an example of a participant information sheet and consent form		
3. Are you gathering data from secondary sources such as websites, archive material, and research datasets?	✓	✓
Click here to find out what ethical issues may exist with secondary data		
4. Have you read the guidance on data protection issues?	✓	☐
a. Have you considered and addressed data protection issues – relating to storing and disposing of data?	✓	☐
b. Is this in an auditable form? (can you trace use of the data from collection to disposal)	✓	☐
5. Have you read the guidance on appropriate research and consent procedures for participants who may be perceived to be vulnerable?	✓	☐
a. Does your study involve participants who are particularly vulnerable or unable to give informed consent (e.g. children, people with learning disabilities, your own students)?	☐	✓
6. Will the study require the co-operation of a gatekeeper for initial access to the groups or individuals to be recruited (e.g. students at school, members of self-help group, nursing home residents)?	☐	✓
Click for an example of a PIS and information about gatekeepers		
7. Will the study involve the use of participants' images or sensitive data (e.g. participants personal details stored electronically, image capture techniques)?	☐	✓
Click here for guidance on images and sensitive data		
8. Will the study involve discussion of sensitive topics (e.g. sexual activity, drug use)?	☐	✓
Click here for an advisory distress protocol		
9. Could the study induce psychological stress or anxiety in participants or those associated with the research, however unlikely you think that risk is?	☐	✓
Click here to read about how to deal with stress and anxiety caused by research procedures		
10. Will blood or tissue samples be obtained from participants?	☐	✓
Click here to read how the Human Tissue Act might affect your work		
11. Is your research governed by the Ionising Radiation (Medical Exposure) Regulations (IRMER) 2000?	☐	✓
Click here to learn more about IRMER		
12. Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind?	☐	✓
Click here to read about how participants need to be warned of potential risks in this kind of research		
13. Is pain or more than mild discomfort likely to result from the study? Please attach the pain assessment tool you will be using.	☐	✓

Click here to read how participants need to be warned of pain or mild discomfort resulting from the study and what do about it.		
14. Will the study involve prolonged or repetitive testing or does it include a physical intervention?	☐	☑
Click here to discover what constitutes a physical intervention and here to read how any prolonged or repetitive testing needs to managed for participant wellbeing and safety		
15. Will participants to take part in the study without their knowledge and informed consent? If yes, please include a justification.	☐	☑
Click here to read about situations where research may be carried out without informed consent		
16. Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?	☐	☑
Click here to read guidance on payment for participants		
17. Is there an existing relationship between the researcher(s) and the participant(s) that needs to be considered? For instance, a lecturer researching his/her students, or a manager interviewing her/his staff?	☐	☑
Click here to read guidance on how existing power relationships need to be dealt with in research procedures		
18. Have you undertaken Risk Assessments for each of the procedures that you are undertaking?	☐	☑
19. Is any of the research activity taking place outside of the UK?	☐	☑
20. Does your research fit into any of the following security sensitive categories: • commissioned by the military • commissioned under an EU security call • involve the acquisition of security clearances • concerns terrorist or extreme groups If Yes, please complete a Security Sensitive Information Form	☐	☑

I understand that if granted, this approval will apply to the current project protocol and timeframe stated. If there are any changes I will be required to review the ethical consideration(s) and this will include completion of a 'Request for Amendment' form.

☐ have attached a Risk Assessment
☐ have attached an Insurance Checklist

If the applicant has answered **YES** to **ANY** of the questions **5a – 17** then they must complete the MMU Application for Ethical Approval.

Signature of Applicant: [redacted] [redacted] Digitally signed by [redacted] [redacted] Date: 2017.04.24 21:36:21 +01'00' Date: 28/10/2016 (DD/MM/YY)

***Independent Approval** for the above project is (please check the appropriate box):*
Granted
☐ I confirm that there are no ethical issues requiring further consideration and the project can commence.

Not Granted
☐ I confirm that there are ethical issues requiring further consideration and will refer the project protocol to the Faculty Research Group Officer.

Signature:____________________ Date:__________(DD/MM/YY)

Print Name: ____________________ Position:____________
Approver: Independent Scrutiniser for UG and PG Taught/ PGRs RD1 Scrutiniser/ Faculty Head of Ethics for staff.