School of Computer Science and Engineering

The University of New South Wales

Machine Learning based classification of botnet network traffic over Tor

by

Abrar Amin

z5018626@ad.unsw.edu.au

Thesis submitted as a requirement for the degree of Bachelor of Science (Honours) in Computer Science and Engineering

Submitted: 23rd November, 2022

Supervisor: Dr. Nadeem Ahmed

Assessor: Dr. Jawad Ahmed

Abstract

Botnets are a collection of bots usually at scale that are controlled by an adversary, the botmaster using Command and Control (C&C) servers. In the past botnets have been used for malicious activities such as email spamming, distributed denial of service attacks, cryptocurrency mining and click fraud. In order to build more resilient botnets and avoid detection, C&C servers have been hosted on and accessed through the Tor anonymity network. Due to network traffic over Tor being anonymized and encrypted we cannot rely on existing botnet Intrusion Detection Systems (IDS). This thesis involved setting up a testbed to capture botnet traffic over Tor, creation of a Tor botnet traffic dataset and applying several Machine Learning Classifiers to fingerprint botnet communication using the Tor network. We have been able to achieve a 88% accuracy score using an Extreme Gradient Boost (XGBoost) classifier to identify if a Tor-encrypted network trace contains botnet communication.

The Tor-botnet dataset produced as a part of this thesis consists of 2.4 GB of packet captures which includes over 1,300,000 records. This was done due to the absence of any publicly available datasets containing Tor-encrypted botnet network flows, and this was required for our Machine Learning application. This dataset and our Machine Learning models would be contributing to other future research on classification and fingerprinting of malicious botnet traffic and creation of network-based IDS. Furthermore we have labelled 3 distinct botnet lifecycle phases in our dataset - "Infection", "Connection" and "Attack". Future work can be done to enrich our dataset and identify the stage of the botnet lifecycle from network traffic traces.

Tags: botnet, botnet detection, Tor, machine learning, network traffic analysis, Intrusion Detection System, de-anonymisation, network security, website fingerprinting

Acknowledgement

This work has been inspired by the hard work of Ishan Karunanayake and Nadeem Ahmed who are actively working on research on network traffic analysis, website fingerprinting and de-anonymization attacks on Tor. They have provided me with support and suggestions which has been immensely helpful for me to get started on my first academic research project and have a better understanding of the required background knowledge. I would like to thank them for their time during our weekly catch-ups.

Abbreviations

BYOB - Build Your Own Botnet, an open-source post-exploitation tool used to implement our botnet.

C&C - Command and Control.

DDoS - Distributed Denial of Service Attacks.

HS - Tor Hidden Service/ Onion Service.

IDS - Intrusion Detection System.

IoT - Internet of Things.

IP - Introduction Point, a relay node on the Tor network.

ML - Machine Learning.

OP - Onion Proxy.

OR - Onion Router.

PCAP - Packet Capture, API for capturing computer network traffic.

RP - Rendezvous Point, a relay node on the Tor network.

Contents

1 Introduction

The word botnet is formed by combining the words robot and network. Bots are used to run automated tasks over the internet, an example of this would be web crawlers used to index web pages. Bots usually play the role of clients in a client-server model. Botnets are usually malware-infected computers (clients) that are controlled by an adversary, called the botmaster or bot-herder using a Command and Control (C&C) server [2]. Botnet clients can run silently in the background without the knowledge of the device's owner. Botnets can also be designed to be self-propagating, that is when a device is infected it can attempt to infect other vulnerable devices within its network [7].

Common uses of botnets include DDoS attacks, sending spam and phishing emails, cryptocurrency mining, password cracking and stealing personal data. In 2021 the largest DDoS attack was recorded by DDoS protection and mitigation company, Cloudflare which involved 17.2 million requests per second at its peak with a total of 330 million requests sent over a space of a few seconds targeting a financial institution which they had attributed to a botnet comprising of 20,000 bots across 125 countries [6].

According to an FBI estimate in 2014, every year around 500 million devices are infected by botnets globally which incurs US $110 billion in financial losses [3]. Botnets are often sold or leased by the botmasters in the black market, with the price of the botnet depending on the size, being more expensive the larger the botnet footprint [4, 5].

More recently botnet authors have been using complex architectures including the use of P2P protocols and anonymity networks like Tor in order to build more resilient C&C servers that are harder to locate and take down by law enforcement authorities [2]. Tor is an acronym for The Onion Routing (Tor) project which is the most popular implementation of onion routing. Tor provides anonymity to the client by wrapping the request in layers of encryption as it is passed through a circuit of proxies before communicating with the server as opposed to the client communicating with the server directly [8]. This makes detecting the IP address of the C&C server a more difficult task from the client side.

The second generation of Tor network which was proposed in 2004 also included the ability to anonymously host Hidden Services (HS), known as an Onion Service within the Tor network itself through the use of rendezvous point [9]. We have already noticed instances of botnet C&C servers being hosted as HS on Tor, as early as 2012 [2, 10, 11]. This move makes it significantly more challenging to locate C&C servers as all Tor traffic is encrypted and are of fixed sized cells, furthermore HS are non-IP based and don't use conventional DNS requests [2].

In Section 2 of this paper we will discuss in depth on Tor-based botnets and existing work that has been done in detecting botnets and other malware traffic over Tor. We will also discuss data collection methods and testbed architectures we have considered and chosen to generate data for our research to use data mining techniques on in Section 3. In Section 4 we will discuss our data collection and preprocessing procedure. We will go into Machine Learning models we have developed to perform experiments and analyse our results in Section 5. We discuss future improvements and work that can be done based on our research and draw a conclusion in Section 6.

2 Literature Review

In the following section we explore in detail the background knowledge acquired throughout Thesis Part A and related academic research to our thesis topic. The following section covers the fundamental knowledge on Tor, Onion Service, Machine Learning (ML) techniques used for malware traffic classification in depth. We will further discuss identified gaps and well as the aims and outcomes of this thesis.

2.1 Tor

Tor, which stands for "The Onion Routing" is the most widely used implementation of onion routing which was first proposed by researchers at U.S. Naval in 1995. Tor network was initially deployed in 2002 under a free and open source licence as it would maximise transparency and decentralisation [13]. Further improvements were made with the second generation Onion Router in 2004 by Dingledine et. al. [9] including the addition of hosting web services within Tor. Tor Project Inc. a non-profit organisation was formed in 2006 to maintain Tor's development as well as the Tor browser.

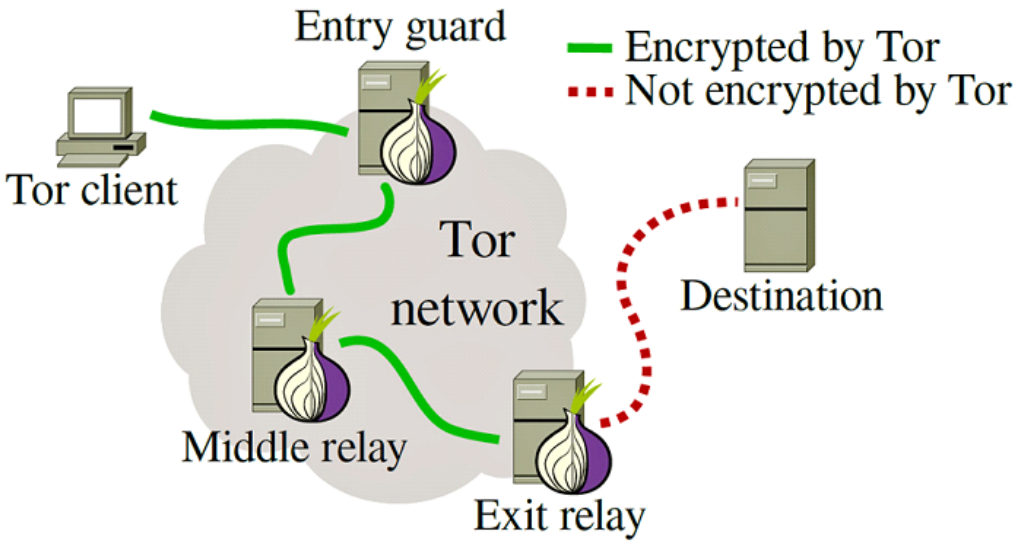

Figure 1: Components of the Tor Network [14]

In order to use the Tor network to access a web service first the client will be required to install an Onion Proxy (OP) on their device, this can be done by using the Tor browser. The main components of a Tor network consists of a circuit composed of a Guard node which serves as the entry point to the Tor network, a Middle relay node and an Exit relay node. These nodes or Onion Router (OR) are usually run by volunteers who donate their bandwidth.

The Tor client downloads a list of all ORs from the directory server (not shown in Figure 1), then it establishes a 3-hop path, also called a circuit consisting of 3 ORs as seen on Figure 1. The guard node gains access to the IP address of the client, and must be a trusted Tor relay. Within the circuit the ORs are only aware of what node lies before it and after it and do not have information on the full circuit, source and destination. Same ORs are never picked when establishing a Tor circuit.

OP uses Diffie-Helman key exchange to share distinct keys with each of these ORs. OP transmits the data across the circuit encrypted with all 3 keys in order. All data within the Tor network is sent in 512-byte cells. Each OR can use their key to only decrypt a single layer in an onion-like fashion. Exit node decrypts the final layer and has the fully decrypted data to deliver to the server the OP wanted to communicate with. When the data is sent back from the server to the exit node the process described above is performed in reverse where each OR encrypts the data with their keys and only the client can decrypt the entire response with all 3 keys [12, 15].

Furthermore, there is another type of node called Tor bridge that is designed to be used in circumstances where connection to Tor network is blocked and are not publically listed in the directory server. These bridge nodes act like guard nodes [1].

2.1.1 Tor Hidden Services

Tor also provides an option for users to host services over the Tor network, which are called Hidden Services (HS) or Onion Services. Using Tor hides the location (IP address) of the HS, they are usually identified with a .onion pseudo domain and can only be accessed using the Tor network.

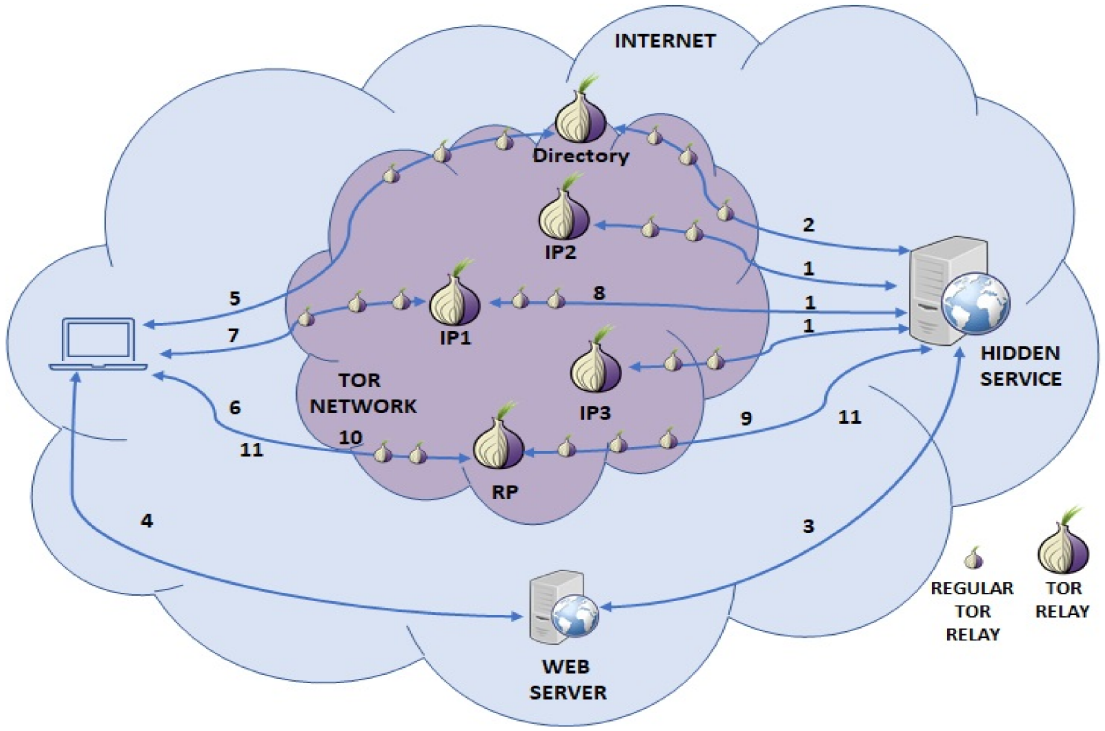

Figure 2: Establishing connections to a Tor Hidden Service [16]

Tor Hidden Service (HS) establishes long-term circuits with 3 relays randomly which are called Introduction Points. The HS then publishes the descriptor containing the Introduction Points to a Distributed Hash using an anonymized Tor circuit and signs the descriptor with its private key, the public key of which typically forms its .onion address.

The client acquires the .onion address associated with the HS and accesses it using an Onion Proxy. The client establishes a Tor circuit to a node which becomes the Rendezvous Point (RP), the client provides a one-time secret to the RP. The Introduction Point then provides the HS information on the client secret and the RP. The HS then creates a Tor circuit to the RP, which matches the two secrets and forms a new circuit between the OP and the HS. All subsequent communications take place using the RP [17]. Figure 2 shows all the Tor nodes that are involved in this process and the sequence diagram.

Using the above protocol, the client does not know the location of the Tor HS, and the HS does not know the location of the client, but both know the location of the RP.

2.1.2 Attacks on Tor network

Tor provides users with anonymity that is useful to whistle-blowers, journalists and residents living under totalitarian governments and users who are conscious of their privacy or want to circumvent censorship on the internet. The anonymity provided by Tor has also been misused to host websites selling drugs, distributing illegal content such as child pornography and for botnet C&C servers [1]. Due to the criminal activities mentioned above, deanonymization of Tor network traffic has become an active area of research by Law enforcement and government agencies.

The most common form of attack on Tor are deanonymization attacks which are performed in an attempt to de-anonymise the users and HS within the Tor network [1].

Traffic Correlation Attacks assume that the attacker has control of the guard node and the exit node of the Tor circuit and can simply try to correlate calls going to the guard node from the client and the calls going out to the server from the exit node in order to deanonymize the client [18].

On Website Fingerprinting attacks, attackers rely on encrypted traffic to have distinctive characteristics and aim to link the user to the website or Tor Hidden Services by looking at leaked side channel information like packet statistics and burst of packets. This is typically done by training supervised Machine Learning (ML) models by first collecting fingerprints from different websites and then using this as training data for ML models. Sirinam *et. al.* utilised a Convolutional Neural Networks (CNN) and achieved an accuracy of 98% on Tor traffic without defence in 2018 [19].

Circuit Fingerprinting Attack was first described by Kwon *et. al.* in 2015 which attempts to fingerprint Tor circuits created from clients to communicate with HS [20] by classifying circuits into 5 categories including HS-IP, Client-IP, HS-RP, Client-RP using features of the circuits such as number of incoming and outgoing cells, duration of activity and the sequence of the first 10 cells. They used Tree-based and k-Nearest Neighbours (k-NN) classifiers for this circuit classification task.

2.2 Botnet

As discussed earlier, Botnets consist of a number of malware-infected computers (clients) that are controlled by an adversary, called the botmaster using a Command and Control (C&C) server [2]. IoT devices like routers have become key targets for attackers due to poor security and factory default or unprotected passwords, as seen by the Mirai botnet in 2016 targeting devices with ARC processors which have been used to perform DDoS attacks [21, 22, 23]. In the section below we explore botnet lifecycle, architectures used by botnet authors over the years and current detection mechanisms.

2.2.1 Botnet Lifecycle

According to Silva *et. al.* [24] and Ghafir *et. al.* [21] there are 5 stages of the botnet lifecycle, as shown in Figure 3.

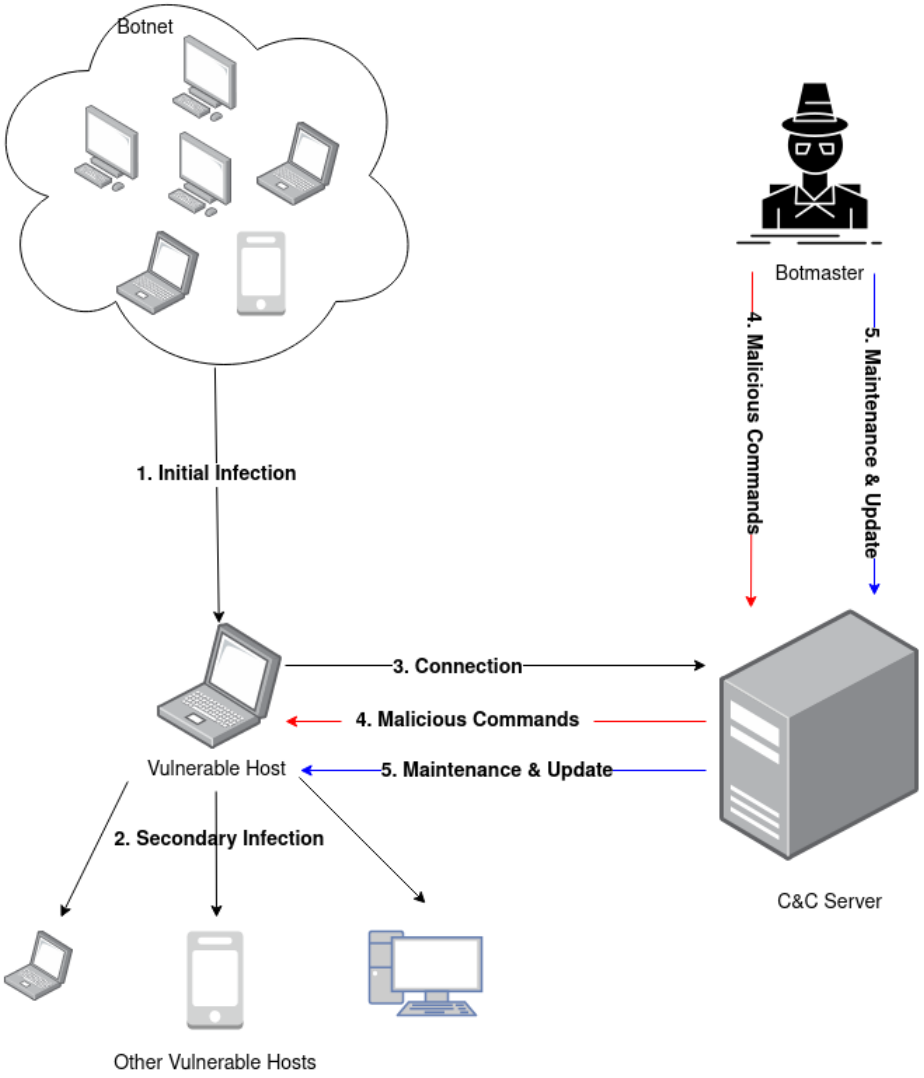

Figure 3: Typical Botnet Lifecycle Stages

During the initial infection phase, the host gets infected through drive-by-download, malicious attachments in email or infected removable storage devices and becomes a potential bot. The host then moves on to the secondary infection phase where it downloads and installs the complete botnet binaries usually from a static location through HTTP, FTP or P2P protocols.

After the secondary infection phase, the botnet client connects to the C&C server in a process called rallying. After connection is established with the C&C server, the device becomes a part of the botnet and can be used for future malicious activities by the botmaster who can now send commands to it through the C&C server. Finally, the botmaster provides

updates to the botnets in order to ensure resiliency and avoid detection, for instance the botnet binaries can be updated to connect to a different C&C server [21].

2.2.2 Botnet Architectures

Some of the common botnet architectures used over the years have been noted below.

Centralised Botnets

Internet Relay Chat (IRC) is a network protocol for text based chat systems for instant messaging in channels was first created in 1988. The first non-malicious botnet was created in 1989 which allowed bots to play the video game, Hunt the Wumpus on an IRC channel. The first malicious botnet using IRC was detected in 1998, named GT-Bot which had several variants [2, 25]. The bot is then controlled remotely using a preconfigured IRC server and channel. One of the weaknesses of using IRC was, Network Analysts can easily detect and block IRC traffic.

HyperText Transfer Protocol (HTTP) protocol was also soon adopted by botnet creators since it was easier to mask the malicious network traffic as regular web traffic compared to IRC and became popular in the mid-2000s [2, 26]. The botnet clients would connect to preconfigured C&C web servers using HTTP.

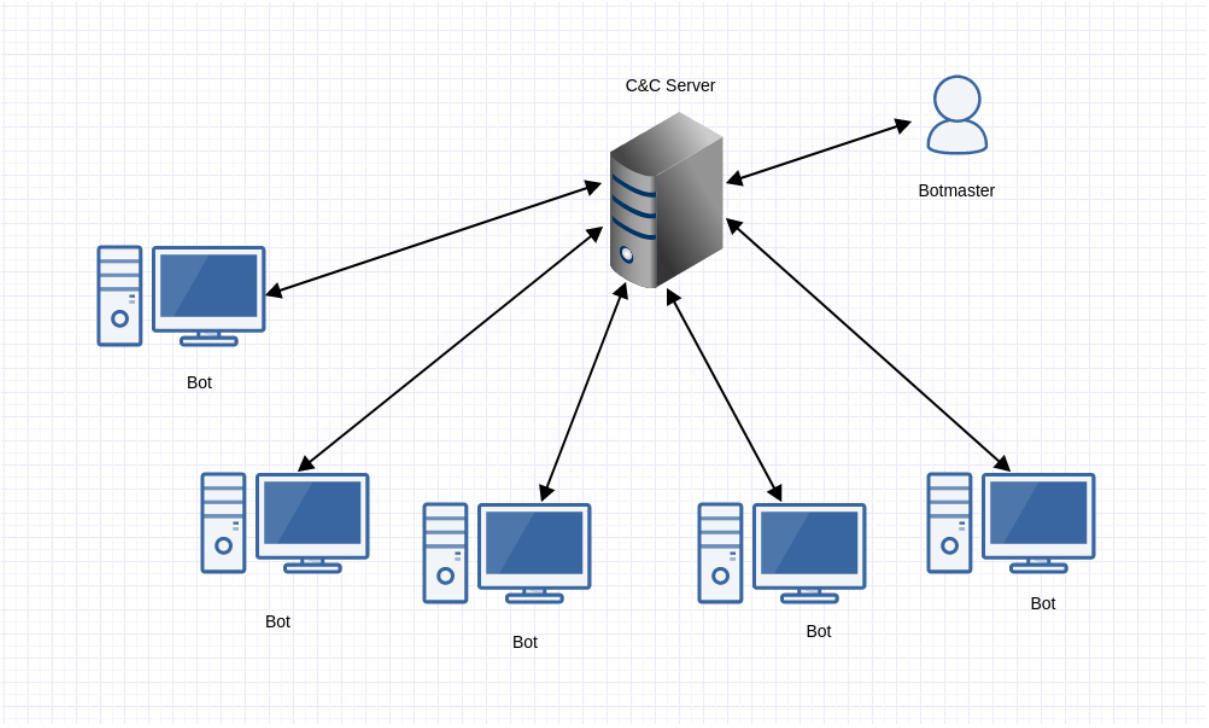

Figure 4: Centralised Botnet Architecture

In Figure 4 we can see a Centralised C&C server which is common for HTTP and IRC based botnets, it is a basic Client-Server model and as mentioned earlier the web servers are usually hard coded into the botnet binary. Here the C&C server is a single point of failure, if it is taken out the botmaster loses control of the botnet.

Decentralised Botnets

The first botnet with a Peer-to-Peer (P2P) architecture emerged in 2003 [27].

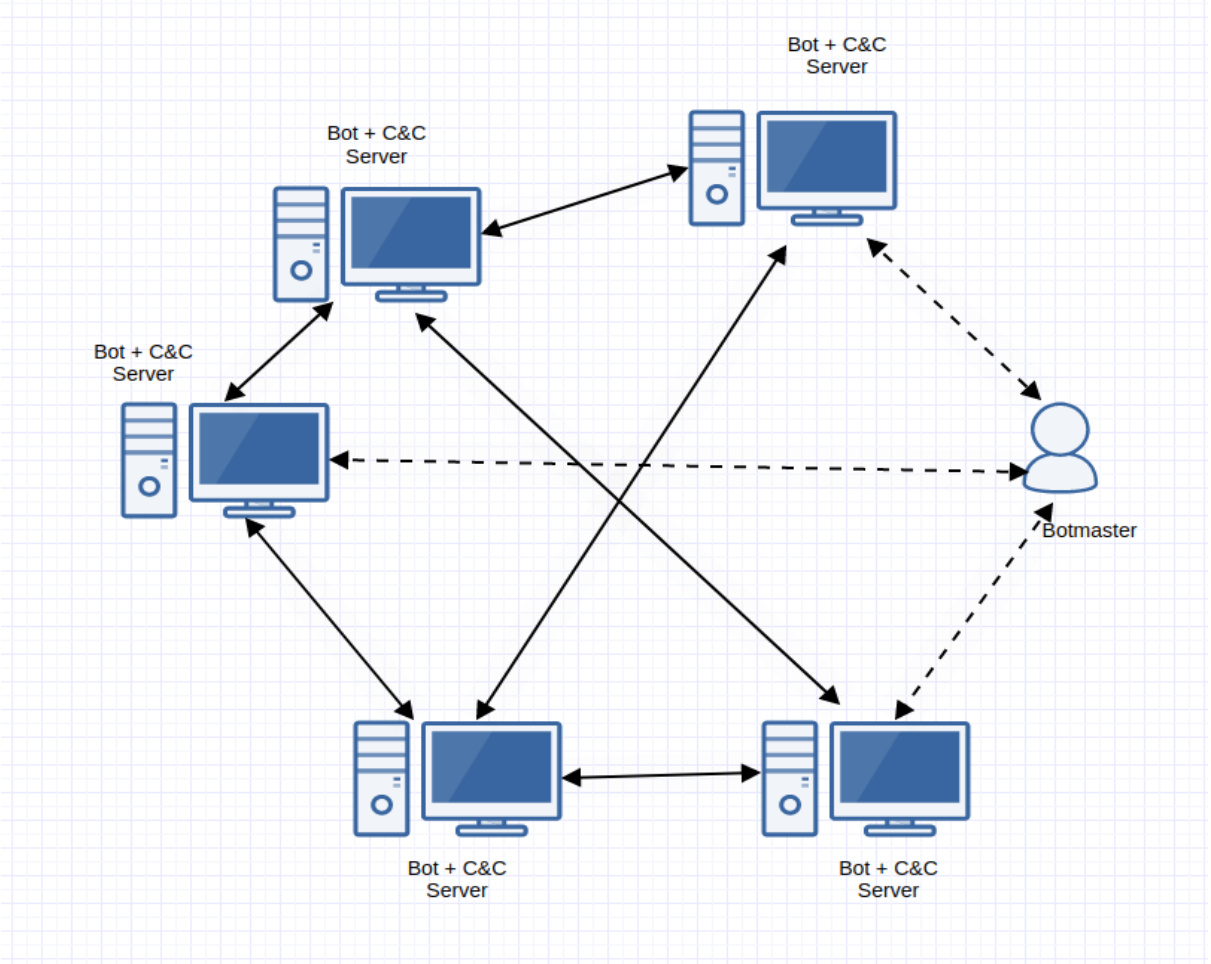

Figure 5: P2P Botnet Architecture

As we can see on figure 5, using P2P networking protocol we remove the need for a centralised C&C server making it more resilient to being taken down. Using a P2P architecture each bot in the botnet maintains responsibility for routing commands and information to each other [26]. Each of the bots are effectively acting as both client and C&C server. Hybrid Architectures that combine the ideas of Centralised and Decentralised architectures are also theoretically possible [32].

DNS Fast Flux

DNS Fast Flux is a technique that involves using several IP addresses to connect to a single domain name and changing these IP addresses rapidly. This is a technique commonly used by botnets to turn static domains into moving targets [28]. The attacker will associate multiple IP addresses with one domain name and frequently update the DNS record associated with it. Storm botnet was using Fast Flux as early as 2007 [29].

Domain Generation Algorithm (DGA)

Bots can generate a large number of domain names in a predictable manner that can be used as rendezvous points to the C&C server. Kraken was the first botnet to use DGA in 2008 and it was later used by Conficker [30].

2.2.3 Botnet Detection Techniques

While there are many different botnet detection techniques, according to Zeidanloo *et. al.* there are two main approaches to botnet detection - A Honeypot based approach and an Intrusion Detection System (IDS) based approach [31].

Honeypots are computers used by the security community to purposefully trap malware for security monitoring and to divert it away from real targets [33]. For our research we are more interested in IDS based techniques that can be further subdivided into Signature-based and Anomaly-based botnet detection approaches.

Signature-based IDS approaches are normally used against identified botnets, as they rely on a known list of threats and signatures. Anomaly-based IDS approaches can be used to detect suspicious activities from known and yet undiscovered botnets and Machine Learning (ML) approaches to train the IDS [34]. Anomaly-based IDS can be subdivided further by a Host based technique which monitors the internal activities of the computer or a Network based technique which monitors external network traffic of the computer [31]. A passive Network-based approach would be the main focus for our research.

2.2.4 Botnets over Tor

Theoretical infrastructures for botnets over Tor was discussed as early as 2010 by Dennis Brown at DefCon 18 Security Conference on his talk titled, “Resilient Botnet Command and Control with Tor” [36]. In his talk he proposed implementing a Tor proxy similar to Tor2Web

and running the C&C server as a Tor HS instead of running Onion Proxy (OP) on the client which might become easy to detect using Host-based Anomaly Detection IDS. He added that the client might get suspicious if all of their network traffic was passed through an OP, as they might end up retrieving web pages distinctive to the country of the Tor exit node (e.g. Google Homepage in Czech language). He further suggested in his talk on adding SOCKS support or setting up a private Tor network with the infected clients in order to make the botnet C&C more resilient.

In 2012 the first botnet using Tor, named Skynet was discovered which was a Tor enabled ZeUS based bot using IRC. The C&C server was hosted within the Tor network as a HS which was capable of performing DDoS attacks and bitcoin mining on the infected hosts, the size of the botnet was estimated to be around 12 to 15 thousand compromised bots [37].

In 2013 security researchers noticed a huge spike in the Tor network usage which has been attributed to the Mevade botnet (also known as Sefnit) which was used for click fraud and cryptocurrency mining which was also hosting it's C&C server as a Tor HS [38]. This increase in traffic to Tor also created latency for all legitimate users of the Tor network [10].

In more recent times we have seen botnets targeting IoT devices being used over Tor network, with BrickerBot being discovered in 2017 which seems to be using a Tor exit node in order to communicate with the C&C server while not using a Tor HS. And in 2019 a variant of Mirai botnet was discovered using Tor HS for it's C&C server [39, 40].

On the other hand Casenove *et. al.* argues that using Tor would only give an illusion of being stealthy to the attacker and mentions that the Tor HS can still be identified using techniques like traffic correlation and fingerprinting that we mentioned before. Furthermore, they argue moving botnets with a large footprint could cause instability in the Tor network leading to their detection much like Mevade [41].

2.3 Related Work

To the best of our knowledge the first work done to identify malicious traffic over Tor was by Ling *et. al.* [12] in 2015 who developed a system known as TorWard which integrated an IDS to a Tor exit node at their university to detect and classify malicious traffic. Their research indicated that using their set up 10% of the Tor traffic to their exit node was malicious traffic

and they were able to detect 200 malware. TorWard also utilised a sentinel agent to help block suspected malicious traffic using a modified version of the Tor network.

The TorWard approach relies on a signature-based IDS, so it might not be able to detect undiscovered malware traffic. Furthermore it looks at exit node communication to web services on the internet, which might not be effective against C&C servers that are being hosted as Tor Hidden Services which was the case for almost all documented occurrences of botnets over Tor since they would not use Tor exit nodes as shown on Figure 2. Since TorWard only works on the exit node it is not useful for detecting malicious traffic across all other Tor nodes.

In 2018, Fajana *et. al.* developed TorBot Stalker which utilises website fingerprinting on the extracted Tor circuit level data. This research involved creating a synthetic dataset to use for ML. The classifier built for this research classifies the malicious network traffic into the following classes - Normal, HTTP botnet, TCP botnet, Telnet botnet, IRC or Web. The best accuracy was obtained using a Random Forest classifier at almost 99.6% [10].

While the result of TorBot Stalker is promising, specifics about the malware binaries used are not mentioned. One of the limitations of this research is detecting long periodic intervals as circuit path ID is changed and another relay is used, and as Fajane *et. al.* states botmasters can circumvent this by changing its circuit path when sending data. Furthermore they also state that network packet analysis can be used to improve their solution and be more persistent even if the circuit changes.

In 2020 Robles *et. al.* created a ML classifier to classify malware and non-malware traffic over Tor. They created their own dataset including malicious traffic from the following Malwares - Dexter, Kazy, Locky, Parite and WannaCry variants from VirusTotal. Using packet-based statistics as features they were able to use Decision Trees, k-Nearest Neighbour, Naive Bayes and Random Forest classifiers to achieve high accuracies [11]. While this research is promising and it goes into depth on the data collection procedures used, it does not take into account C&C servers hosted using Tor Hidden Service.

2.4 Problem Statement

We noticed that there is a lot of research being done to identify ways to attack the Tor network as well as studying botnets. We have seen how botnet authors have continued to evolve the architecture of botnets in order to make them more resilient from being taken

down, and we have seen the rise of botnets over Tor anonymity network since it was first discovered in 2012.

Since there has not been much research done in passive network-based IDS over Tor for botnet traffic this presents us with an unique opportunity to experiment with and create new algorithms and Machine Learning models in order to detect botnet traffic.
The related works to our project mentioned above have not published the datasets they had used to train and evaluate their ML Classifiers, this meant that we would be tasked with creation of the dataset first in order to apply ML/ Data Mining. We intended to make this dataset we created publicly available so any future research work can benefit from it and contribute to it, and use it as a standard to compare the performance of different ML models or any other Botnet detection systems created in the future. We discuss more on data collection procedures on Section 3.

As a deliverable for this thesis, we will be attempting to answer the following questions - Can we construct a labelled synthetic dataset consisting of malicious and non-malicious traffic which simulates real network traffic as closely as possible to use with supervised ML models and Neural Networks? Can ML be effectively used to classify Botnet network activity over Tor from other legitimate Tor and Non-Tor traffic? We discuss further on Machine Learning models we have created, experiments we have performed and our results on Section 4.

3 Methodology

In this section we discuss a few experimental set-up for our project that we had considered for capturing malicious and benign Tor-encrypted network traffic. We will also discuss the botnet binaries we had considered in order to generate the network traffic.

3.1 Existing Botnet and Malware datasets

We have first tried to identify if we could use any datasets that currently exist publicly for our project, partially or fully. We came across a few datasets that contain botnet network traffic and Tor network traffic, summarised in the Table below:

Dataset Name	Description	PCAP Size
ISCX Botnet Dataset (2014) [45]	Contains non-malicious network traffic from a wide range of network protocols as well as botnet traffic from IRC, HTTP and P2P based botnets. In both the training and test datasets around 45% of the traffic is malicious the rest is benign.	Training: 5.3 GB, Test: 8.5 GB
ISCXTor2016 (2016) [42]	Contains PCAP files captured using TCPDump and Wireshark for Tor and Non-Tor network traffic across several network protocols including Browsing, email, chat, P2P, etc.	22 GB
Trojan T.Bot (2012) [44]	PCAPs originally posted by contagiodump from the Skynet Tor-based botnet from 2012, the first shared by the author of contagiodump blog [43]. The dataset acquired also contains the original botnet binaries but they seem to be encrypted.	32 MB
CTU-13 (2011) [51]	Contains botnet traffic mixed with normal and background network traffic across 13 different scenarios, varying with the botnet used and the number of bots.	1.9 GB
The Bot-IoT Dataset (2021) [45]	Synthetic dataset combinatining botnet and normal traffic from IoT devices, contains over 72,000,000 records.	69.3 GB

Table 1: List of popular botnet datasets

Looking at the table above we can see that despite numerous datasets existing for Tor network traffic and botnet network traffic, very limited data is available for botnet traffic over Tor. We have managed to acquire packet captures from the original Skynet Tor-based botnet dating back to 2012 which will be extremely valuable for us to analyse and try to replicate at

a larger scale using different variants of botnet binaries, however it is too small just to use on its own.

The datasets created and used by Fajana *et. al.* [10] and Robles *et. al.* [11] would have been useful for this thesis had it been made publicly available.

3.2 Testbed Set-up

In this section we discuss network testbed architectures we have considered and the advantages and disadvantages they present.

3.2.1 Capturing outgoing traffic to Guard Node (Live Tor network)

The set up we can use to capture Tor encrypted traffic in this case would be similar to the approaches used by Fajana *et. al.* [10] and Robles *et. al.* [11] . In our research, since we intend to run the C&C server locally we can capture traffic going into the Guard Node from both the Tor client (infected bot) and Tor Hidden Service (C&C server). This will also give us a unique opportunity to experiment with de-anonymization attacks like circuit fingerprinting attacks as described in Section 2.1.2 of this report.

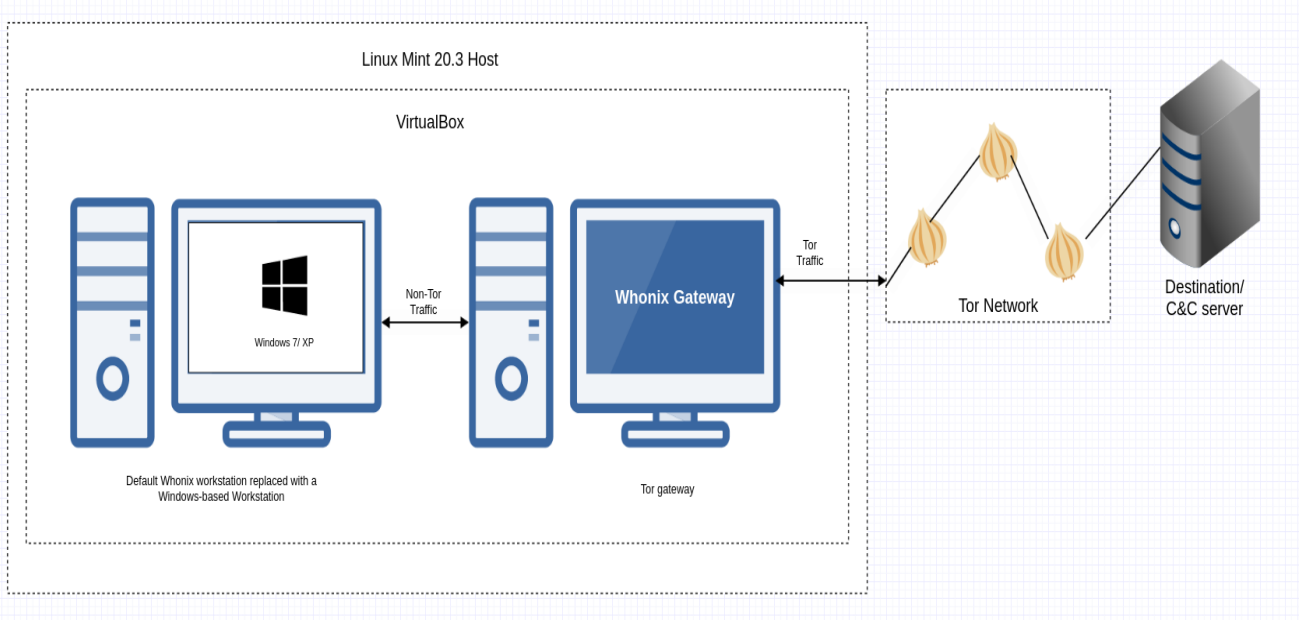

Figure 6: Testbed set-up using Whonix and Live Tor network

Using an approach similar to Robles et. al. we can use a Whonix gateway which will relay all network traffic from a Windows 7 or XP workstation, both of them running as Virtual

Machines within a Linux host. Whonix is an OS which utilises Tor and runs within the user's computer as a virtual machine (VM).

Whonix consists of 2 separate VMs - Whonix Gateway and Whonix Workstation. The Whonix Gateway runs the Tor process while the Workstation runs applications that use the internet on an isolated network. All network traffic from Whonix Workstation goes through the Whonix Gateway [52]. Hence we can use Whonix to ensure all network traffic generated by the botnet client is going through the Tor network.

One of the main motivations of using a Windows workstation instead of the default Whonix workstation is because most botnet binaries were targeting Windows OS. However, for the final set up we had used for this thesis we used the default Linux Whonix Workstation instead of using a Windows workstation, as mentioned on Section 3.3.2 of this report.

3.2.2 Capturing outgoing traffic at every node (Private Tor network)

The second approach we suggest is a private Tor network by using a tool called chutney provided by Tor to set up a private Tor network for our data collection. An advantage of this approach is we will get to capture encrypted Tor traffic at every node within the OP-HS communication. This will also have no impact on Tor users, as we are setting up our own private Tor network.

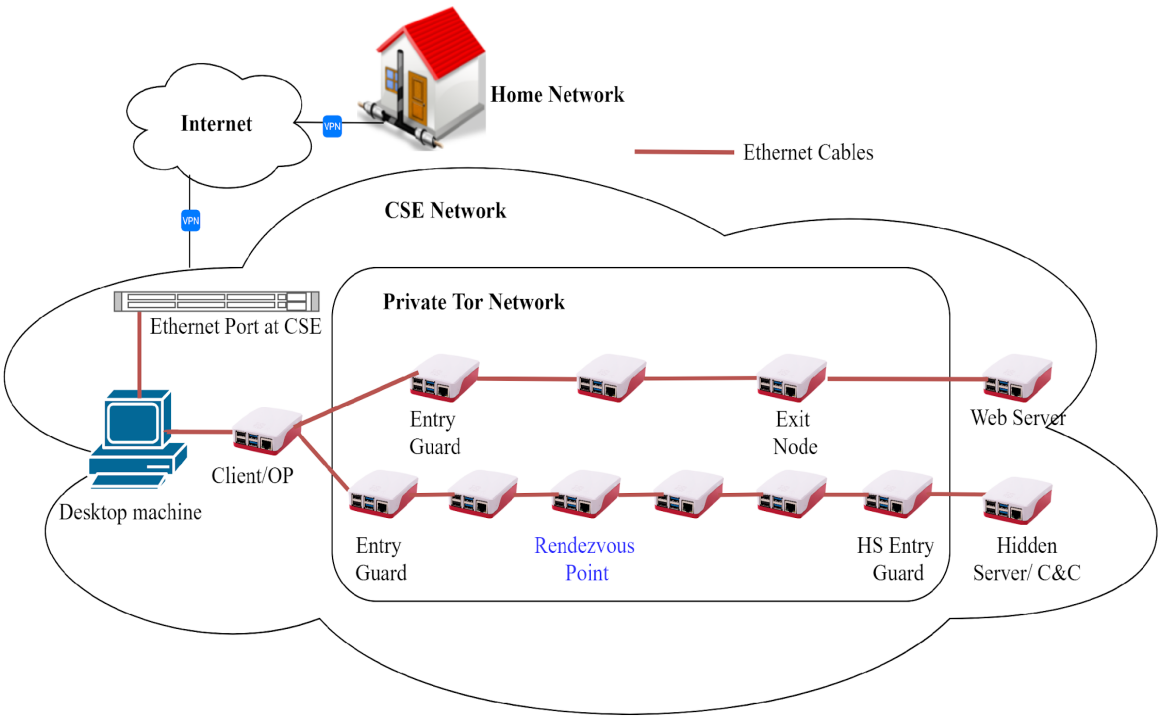

Figure 7: Testbed set up using private Tor network [53]

We have acquired 12 Raspberry Pi 4s and a Cisco LinkSys router as well as multi-port USB hubs and some ethernet cables in order to attempt to set up a private Tor network as seen on Figure 7. While such a set up would be immensely useful in order to use ML classifiers across any given node within the private Tor network, it would require a strong understanding of computer networking in order to capture and label data across all of these nodes.

This would become a complex structure to maintain and debug. More equipment including ethernet cables and USB Type-C cables to charge the Raspberry Pis were required to work on this set up. Furthermore, we were primarily looking at capturing network traffic leaving the botnet client to the guard node and decided that running the C&C server as a Tor HS would not have much benefit for our research. It can also be added that documentations and resources were quite limited on this approach, which made it less viable.

3.2.3 Simulator based approach

Shadow Tor simulator is widely used by researchers to simulate the Tor network which is a discrete event network simulator. One benefit of using this approach is that we no longer rely on the live Tor network, unlike the architecture we suggested in *Section 3.2.1* which is usually discouraged as it might pose a threat to the security and privacy of other legitimate Tor users.

However using the Shadow simulator we are unable to capture network traffic using tools tcpdump or t-shark so it would not be useful for our data collection.

3.2.4 Summary

We have explored different datasets that are publicly available and could not find a suitable dataset to apply supervised ML on for our research, so a big component of Thesis Part B was testbed set-up and data collection. Due to the Tor Shadow simulator not being fit for our purpose and the private Tor network being too complex, we decided to utilise the live Tor network making use of Whonix Gateway.

According to the Tor Research Safety board [55] even though it is encouraged to use a private Tor network wherever possible, we can take steps to mitigate the risk by only attacking ourselves and our own traffic and not infecting other devices through secondary infection, which is satisfied by our proposed architecture as seen on section 3.3.1 and 3.3.2.

Furthermore, we foresee our preliminary dataset to be quite small for the thesis generated by only 2-3 bots, which is highly unlikely to overload the Tor network in any capacity like the Mevade botnet over Tor in 2013 [38]. The C&C, bots and the report server will be controlled by us. Due to time constraints of the Honours Thesis, to reduce complexities of the testbed we decided to use the live Tor network using Whonix as mentioned in Section 3.2.1.

3.3 Botnet Binary Selection

When selecting the botnet binary (client) and C&C server for our testbed we were faced with two options as discussed below -

First we considered botnet C&C server and client source code or binaries that had been previously used maliciously in a production environment in the past, but has been leaked or released since. It is important to ensure these malware are safe to set-up and operate in our testbed.

These have the benefit of closely reflecting real world botnet infection behaviour. Furthermore many newer botnets have been created using and improving up on these leaked source code. For instance Moobot, Satori and Masuta are based on the Mirai source code [55]. One big disadvantage of this approach is that it proved to be hard to find leaked malware source code with the exception of Mirai. The source code had to be closely studied

before being considered a good candidate for generating our dataset, to ensure that we can set-up and run them safely.

The second approach we had considered was using a botnet designed for testing and research purposes. We had considered creating our own botnets using Remote Access Trojans (RAT) or using C&C server and botnet client source code created for non-malicious research purposes. This includes post-exploitation tools such as Build Your Own Botnet (BYOB) [56].

3.3.1 Mirai

Mirai source code was released by authors on Hacker Forums in 2016 [57], and has since been widely researched such as Antonakakis *et. al.* [59]. The loader and bot is written in C while the C&C server and the scan listen server is written in Go. The C&C server uses MySQL for storage. It mainly targeted IoT devices such as routers and IP cameras.

Mirai is capable of contaminating other vulnerable devices using telnet flooding, looking for open telnet slots and trying to log in using commonly used username and password combinations. It has been used to coordinate DDoS attacks, and was used in October 2016 to take down DNS provider Dyn.

Figure 8 shows the architecture of mirai, in addition to the items we talked about in Section 2.2.1 for botnet lifecycle we see that mirai contains a report server and a loader. When infected bots are successful at brute forcing a vulnerable host with the predefined username and password combinations, the details are sent by the bot to a report server, which then contacts the loader server which loads the botnet binaries in this vulnerable host that had just been discovered by the botnet. The vulnerable host then becomes a part of the mirai botnet.

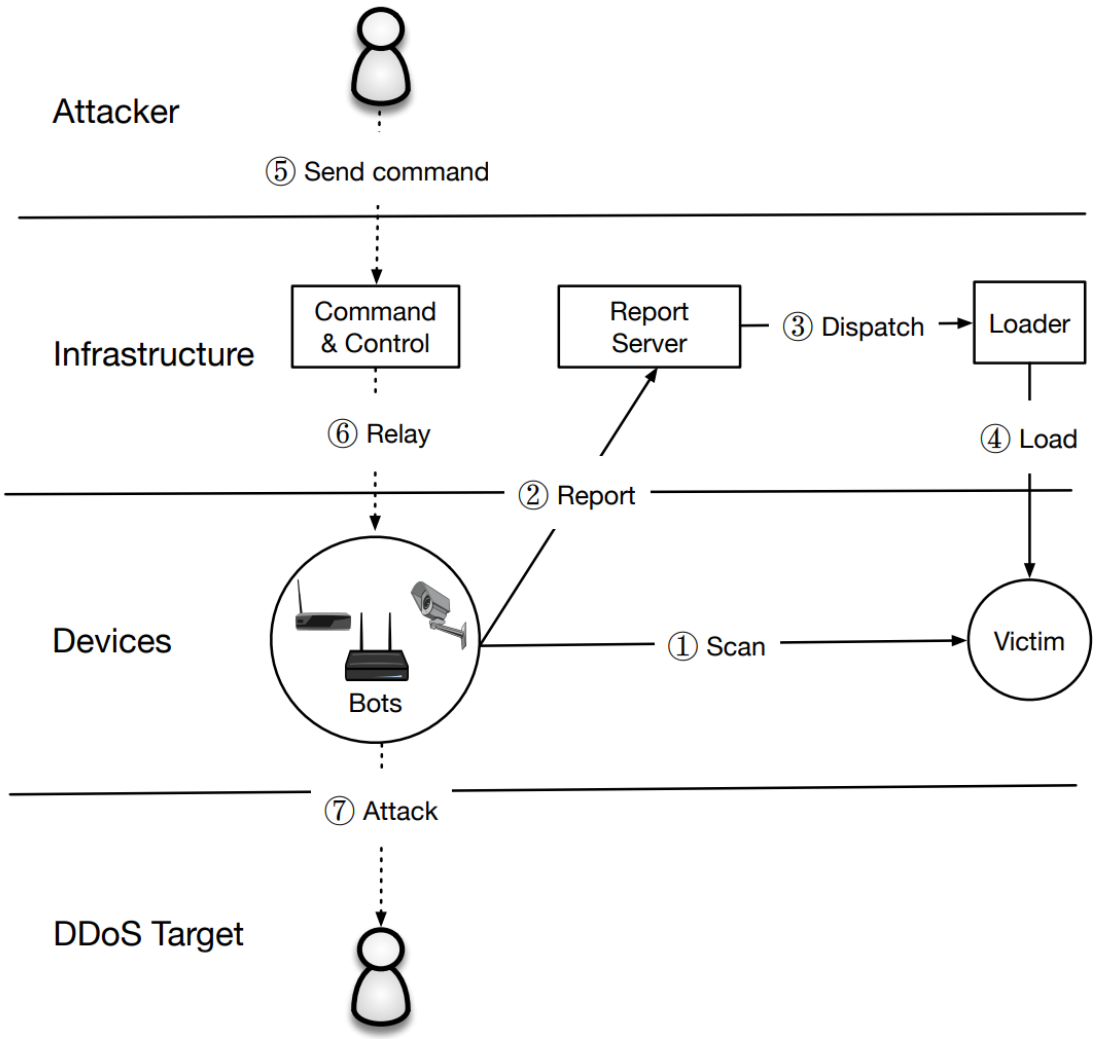

Figure 8: Mirai Lifecycle [59]

In Figure 9 we can see our proposed set-up for mirai within our testbed defined in Section 3.2.1. We had set up our botnet without a loader server, as we were looking to only capture traffic from the botnet client.

In the process of setting up mirai, we were faced with a few blockers, mainly since the source code has been untouched since 2016 some of the dependencies were no longer compatible with current versions, we had to downgrade go to 1.16 and make code changes in order to compile the C code for the botnet client [60]. We also had to provide a domain name, and the source code didn't support raw IP addresses.

Despite addressing all the issues mentioned above and being able to get Telnet bruteforce to work on our testbed set up, we were unable to communicate with the C&C server using Tor.

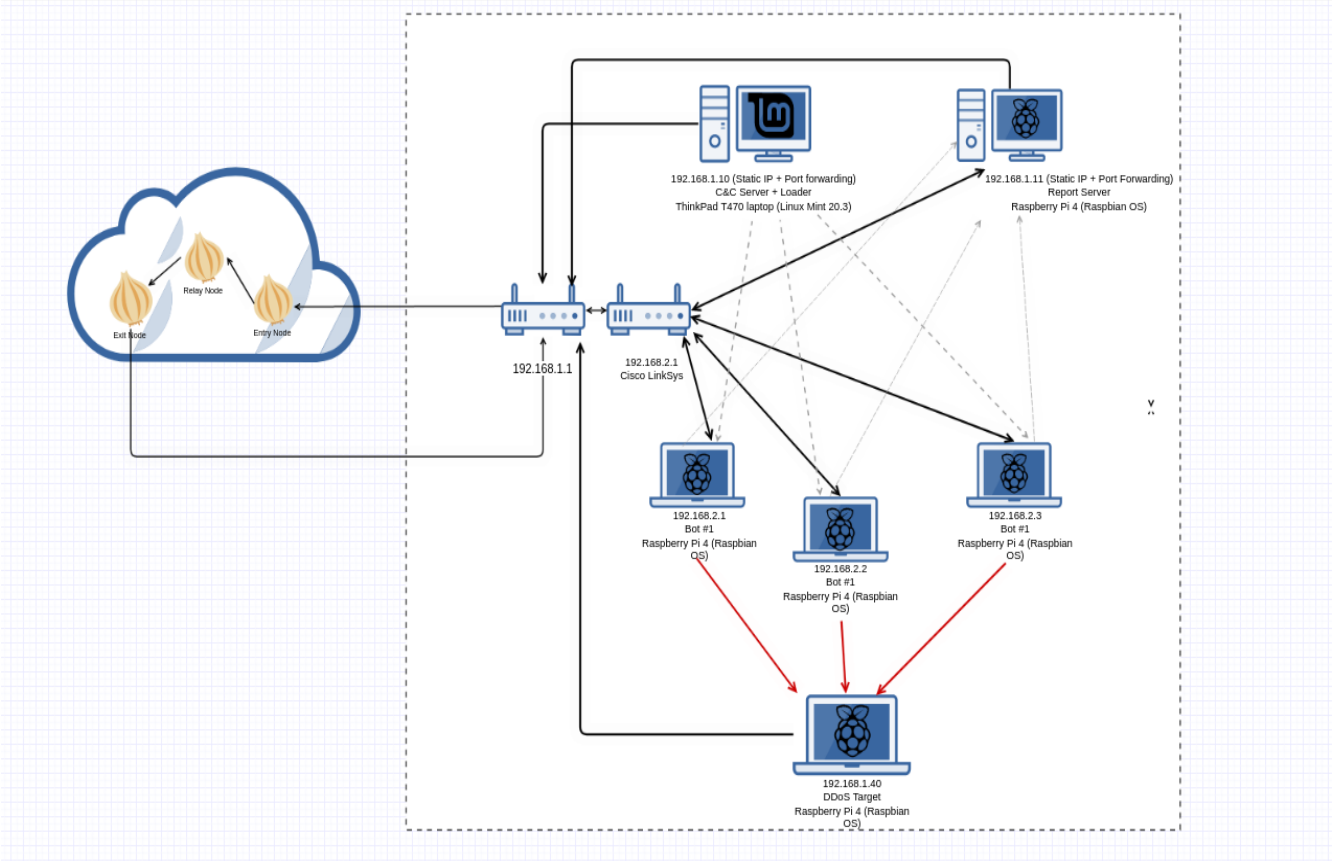

Figure 9: Mirai Data Collection Testbed

3.3.2 Build Your Own Botnet (BYOB)

BYOB (Build Your Own Botnet) is a post-exploitation tool and we have selected it for now as it can be used to create a C&C server with an easy to use UI. [56]

BYOB C&C has built-in exploitation modules to initiate malicious activity such as keylogging, cryptocurrency mining, packet sniffing and port scanning on the infected bots. Furthermore it supports writing our own modules and letting us execute post-exploitation modules from the C&C server to the infected botnet.

BYOB establishes an encrypted TCP connection between the botnet client and C&C server. Using this approach we can still collect data for the botnet lifecycle mentioned on the earlier slides. BYOB infection steps are as described -

- Dropper - first file to be executed, tiny file whose job is to fetch the next stage of code (the initial download).

- Stager - Runs environment checks, downloads main payload.

- Payload - Remotely import python packages and modules from C&C - operates as a reverse shell.

Even though this is a test botnet rather than a botnet used in a production environment - it works very similar to botnets such as GitPaste-12 or roboto which had similar reverse shell capabilities [61][62].

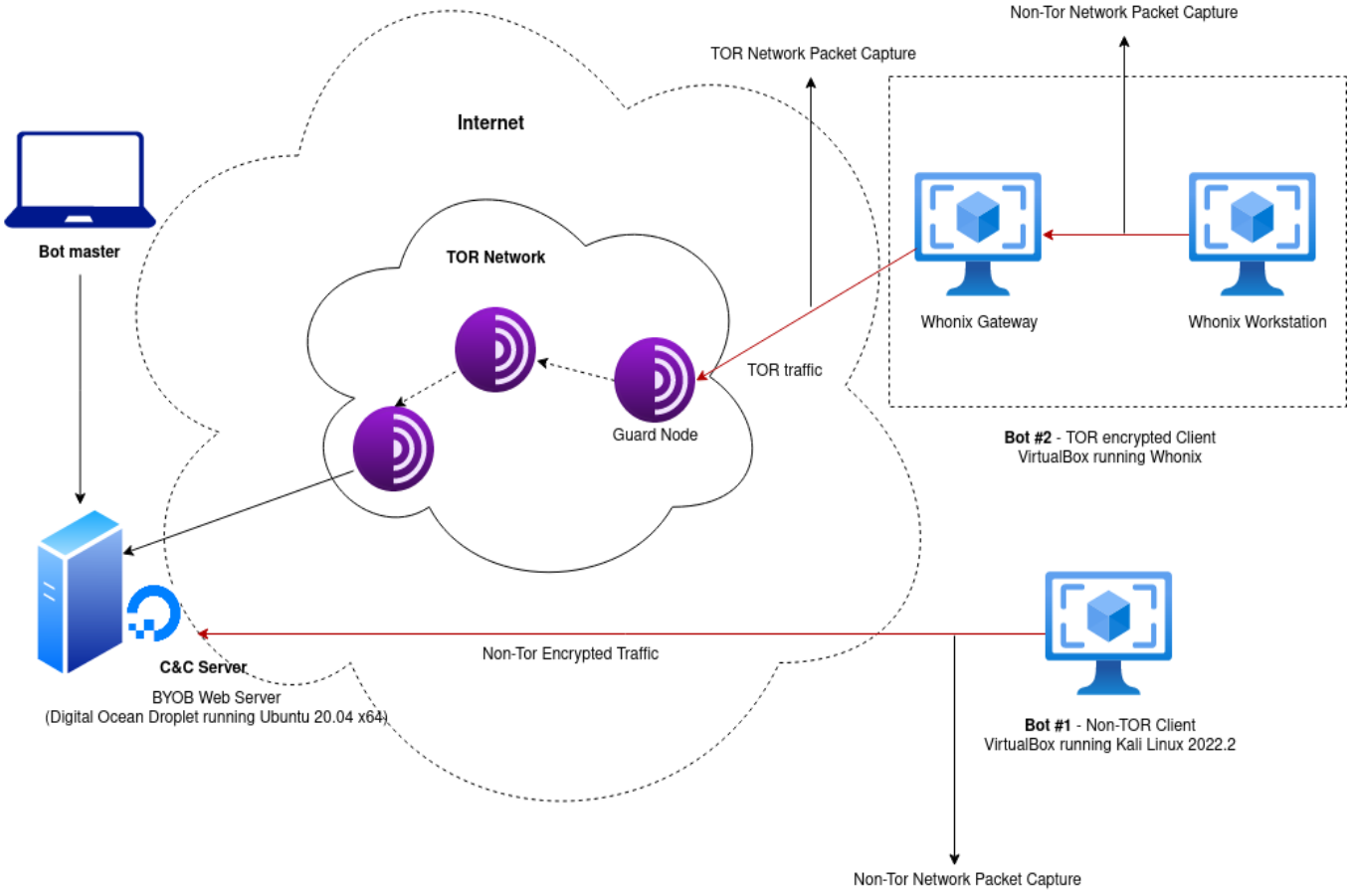

Figure 10: BYOB Data Collection Testbed

Figure 10 shows BYOB botnet on our testbed as described on Section 3.2.1. We are mainly interested in the network traffic capture to and from the Whonix Gateway interface that connects to Tor for our dataset.

3.3.3 Summary

We have explored Mirai and BYOB as candidates for setting up a botnet to collect network traffic from. Due to the botnet client to C&C connection issues we had faced with mirai, we decided to focus on using BYOB for our data collection, which is detailed in Section 4.

We have provided documentation on our set-up procedure for the testbed as well as running BYOB and mirai botnets in it, which will enable us to generate more data in order to enrich our dataset or for other future improvements.

4 Dataset Creation

We used the testbed set-up with BYOB described in Section 3.3.2 to collect botnet traffic over Tor as well as separate traces containing benign network traffic. We collected network traffic for botnet communication and benign traffic in 1 minute traces, collecting 180 samples of each for Tor-encrypted botnet communication and Tor-encrypted benign network traffic, with a total of 360 traces. Our dataset entirely consists of network packets captured from the Whonix Gateway network interface that connects to Tor.

For benign traffic collection, from our Whonix Workstation we accessed different web pages of popular news, social media, shopping websites as well as using popular search engines to replicate real world user activity as closely as possible. We also accessed several popular Tor Hidden Services (.onion links), all of which was done using the in-built Tor Browser in Whonix Workstation. Furthermore we streamed video on YouTube and Vimeo and audio on SoundCloud and Wikimedia Commons. We downloaded several files and software using the Tor browser, and updated Debian package sources and packages on Whonix Workstation using apt. We also collected traffic from non-malicious ssh sessions. We ensured the BYOB client was terminated and we were not communicating with the C&C server when collecting benign traffic.

For botnet communication traffic, we captured traffic in the 3 following lifecycle stages -

1) Infection stage - first execution of botnet binary by a vulnerable host, fetching additional dependencies from C&C server.

2) C&C Communication - bot to C&C communication, executing commands on bot using reverse shell.

3) Attack traffic - network traffic captured from executing post-exploitation modules. For this dataset we only used our botnet to perform port scanning attacks.

We ensured that all other processes apart from BYOB and critical system processes that are

using the network are terminated before we start our capture. We are able to check for processes to terminate using the netstat and lsof commands.

We are able to capture network traffic using tcpdump, and we use the timeout command to ensure we terminate our capture within our defined time intervals. The raw PCAP files we captured for both botnet and benign network traffic using tcpdump were mostly taken in 5 minute intervals. We have collected 2.4 GB of network traffic over a duration of 360 minutes and 1,309,939 network packets [65].

4.1 Data Preprocessing

The data we had captured were all mostly kept to 5 minute traces, to ensure we were not mislabelling our captures, or some other processes were using the network. However, to increase the number of traces we decided to 1 minute traces as mentioned before. This was done using the editcap tool to produce 360 traces in total, 180 each of benign and botnet traces.

The second problem we encountered was that our data was in pcap format and we had produced 360 pcap files from our previous step. In order for us to import this dataset to a numpy array on python, we would prefer csv or txt files instead. Even though it is possible to use pcap libraries in python it would be time consuming to load the dataset everytime we were running our ML models.

While it is possible to use Wireshark and manually export the packet captures as CSV, this process is not feasible and would be time consuming to repeat. We used a python script that would export the columns we required from our pcap files and write to a plain text file.

We extracted several statistical features as mentioned in Section 5.1 using the features used by Hayes *et. al.* for k-fingerprinting for fingerprinting Tor encrypted web traffic using Random Forest [63]. For this reason we were required to export our packet capture data as raw text with 2 columns. The first column represents the time in seconds since the first packet was captured and the second value being -1 or +1 to indicate the direction of the packet, in or out.

5 Experimental Evaluation

We are now interested in evaluating how effectively we can use Machine Learning models to perform a binary classification on whether a given network trace contains botnet communication based on the dataset we had created as described on Section 4.

We are primarily working on a binary classification problem, the 2 classes being - botnet tor traffic and benign tor traffic, labelled in our dataset as 1 and 2 respectively.

5.1 Feature Selection

In order to make effective predictions we would be required to add some features to our existing dataset. We utilise the feature extraction code that was used by Hayes *et. al.* for their implementation of the k-fingerprinting model for Tor website fingerprinting [63]. We made use of their feature selection script [64], and were able to extract 175 statistical features, Some of the important features in k-fingerprinting were number of incoming and outgoing packets, total number of packets, incoming and outgoing packets as a fraction of total packets, packet ordering statistics and concentration of incoming and outgoing packets.

5.2 Machine Learning Model Building

Looking at the classification models used by Fajana *et. al*. for TorBot Stalker [10] and Robles *et. al.* for TorMal2019 [11] and Hayes *et. al.* for k-fingerprinting we can notice that the best performing Machine Learning classifiers were C4.5 Decision Trees and Random Forest Classifiers for similar fingerprinting problems to us.

From our 180 instances each for benign and botnet traffic, we first shuffle the data traces and then create a training dataset consisting of 144 traces of botnet traffic and 144 traces of benign traffic. We use the remaining 36 traces of botnet traffic and 36 traces of benign traffic for creating our validation set to test our trained ML model against, which represents a 80-20 split.

We make use of several ML Models including Random Forest, Gaussian Naive Bayes, CART (Classification and Regression Tree) Decision Tree, Adaptive Boosting (AdaBoost), eXtreme Gradient Boost (XGBoost) Classifier and Light Gradient Boosting Machine (Light GBM) Classifier.

We have used average accuracy scores from 10-fold cross validation on our ML models using our training dataset, to estimate performance of our ML models on unseen data. Cross-validation provides strong models, in comparison to train-test-split we are losing a section of our data for testing purposes while with 10-fold cross validation we can make predictions using all of our data. Another key reason for implementing cross validation is that it allows us to do hyper-parameter tuning which allows us to make models with higher accuracy given our initial dataset.

For the Random Forest Classifier we have tweaked the number of trees to be used in the forest by applying grid search and cross-validation. The benefits of a grid search is rigorous

collection of parameter values across the entire training dataset, the problems that might arise from this is overfitting on the dataset. In order to avoid this we use Crossfold Validation in order to maximise the values that are applicable regardless of the dataset. The final hyper-parameter values are based on highest average performance across the 10 folds.

5.3 Machine Learning Result Analysis

After completing training on the ML models mentioned above, we use them to predict the labels of our validation set. We track the validation accuracy, test precision, test recall and test F1 Score for all the classifiers. The results when sorted by test accuracy which we are using as our evaluation metric are as follows -

	Random Forest	XGBoost	AdaBoost	LightGBM	Decision Tree	Gaussian Naive Bayes
Test Accuracy	87.5%	87.5%	86.1%	83.3%	81.9%	73.6%
Test Precision	88.5%	90.1%	84.2%	83.3%	79.5%	84%
Test Recall	86.1%	83.3%	88.9%	83.3%	86.1%	58.3%
Test F1 Score	87.3%	87.0%	86.5%	83.3%	82.7%	68.9%

Table 2: ML Model Performance

Furthermore we can visualise the training time and test accuracies using matplotlib -

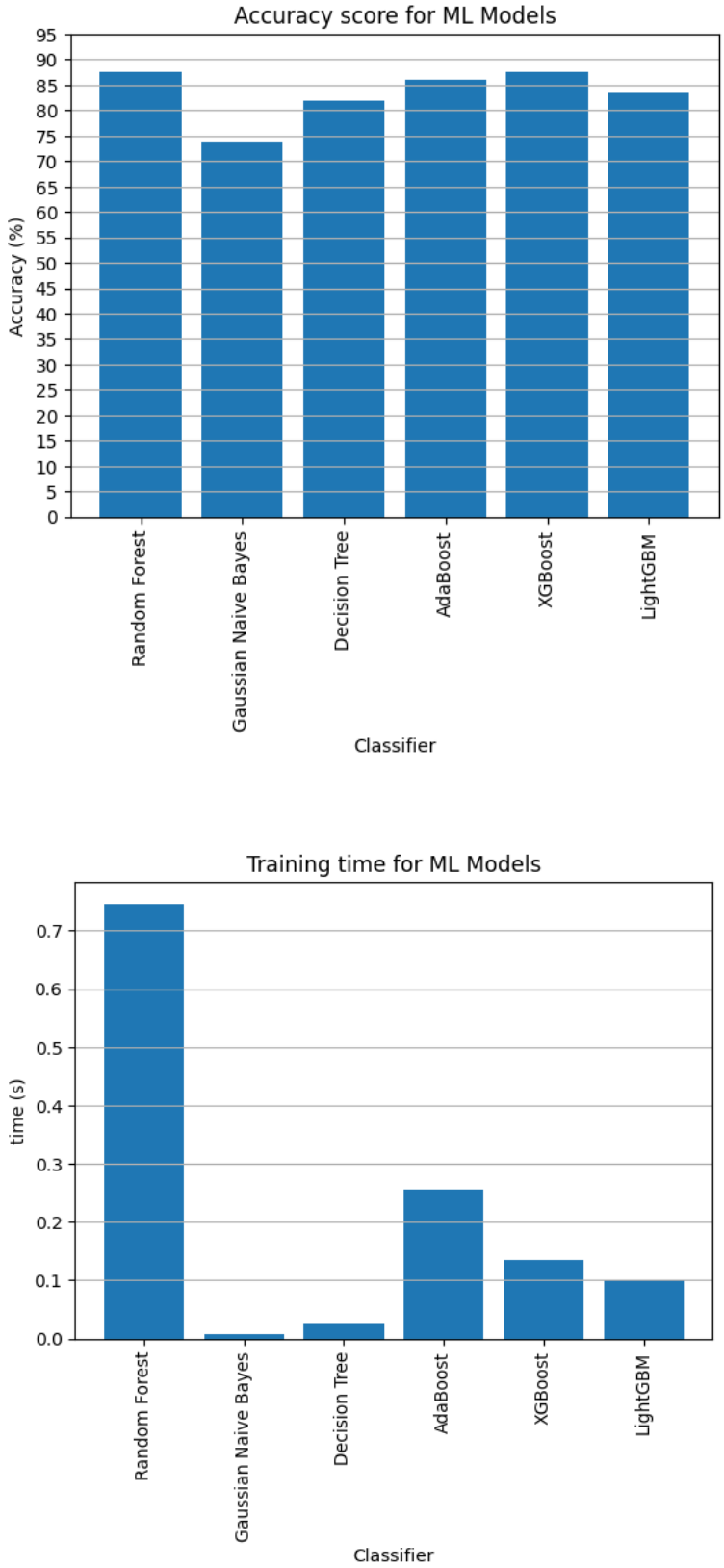

Figure 11: Machine Learning Model Performance

We have also added plots for confusion matrices based on our ML models in Appendix A.

We noticed that after training and validating over several randomised training and validation sets XGBoost, Random Forest, Decision Tree and LGBM give us the best results for test accuracy, with Gaussian Naive Bayes constantly performing poorly for our dataset. We can also see that the performance of XGBoost and LGBM is on-par with our Random Forest model despite taking a fraction of the time to train.

One limitation of our experiment was that we had a single dataset of 360 network traces, and our validation set consisted of only 72 randomly selected traces. We can expect to see an improvement in performance with additional labelled network traces added to our dataset.

6 Conclusion and Future Work

6.1 Conclusion

Preliminary research as described in Section 2 showed that Machine Learning had been effectively used to fingerprint Tor encrypted website traffic and malicious traffic in the past.The main focus for our thesis is to explore whether we can create a dataset for botnet traffic over Tor network, and following that if we can use advanced Machine Learning techniques to identify Tor encrypted botnet communication from benign Tor traffic.

Work done for this thesis provides a review of different testbed architectures that can be used to collect Tor encrypted botnet traffic, as well as detailed set-up instructions for mirai and BYOB botnets in our testbed. We have also contributed by creating a Tor botnet dataset in the absence of any publicly available datasets as outlined in Section 3.1 [65]. The dataset we have created is also labelled by the botnet lifecycle phase which makes it unique.

FInally we have been able to create advanced Machine Learning models to detect Tor-encrypted botnet communication with high accuracy, fulfilling the objectives we set out in Section 2.4. Previous research in detecting Tor encrypted malware traffic have mostly used Random Forest and Decision Tree, as a part of our research we were able to observe how we can use Gradient Boosting Machines such as XGBoost and LightGBM on such datasets and compare their performance against traditional Decision Tree Classifiers.

6.2 Future Work

Given how the scope for this thesis was quite broad involving Network Security and Machine Learning there is a lot of scope for future work based on it. There are several directions we can focus on expanding on our research, such as -

1) **Enriching the dataset -** We can enrich our dataset to make it more robust, by adding network traces for more botnet attack vectors over Tor such as DDoS, key logging, cryptocurrency mining and click fraud, and add network captures from other botnet binaries that use different protocols to communicate with the C&C server such as IRC, Telnet, P2P and HTTP. We can also consider creating a dedicated test dataset, which would enable us to consistently check accuracies of several ML models. This dataset can then be made open source and would be a great contribution in the field of network fingerprinting.

2) **Classifying stages of botnet lifecycle -** As a part of the dataset created by us for this thesis we have also labelled the botnet communication network captures with the lifecycle phase it belongs to. We can use advanced ML techniques to identify the stage of the botnet lifecycle the network trace belongs to.

3) **Creation of a tool that detects Tor encrypted botnet communication** - this can analyse Tor network traffic traces and predict if botnet communication is present. We can potentially consider applying advanced Deep Learning algorithms such as Convolutional Neural Networks and Long Short-Term Memory Neural Networks and feature selection techniques as well as better preprocessing and hyperparameter tuning to improve our test accuracy.

References

[1] I. Karunanayake, N. Ahmed, R. Malaney, R. Islam and S. K. Jha, "De-Anonymisation Attacks on Tor: A Survey," in IEEE Communications Surveys & Tutorials, vol. 23, no. 4, pp. 2324-2350, Fourthquarter 2021, doi: 10.1109/COMST.2021.3093615.

[2] W. Ahmad, “Why Botnets Persist: Designing Effective Technical and Policy Interventions,” *International Policy Research Initiative at MIT*, Feb-2019. [Online]. Available: https://internetpolicy.mit.edu/wp-content/uploads/2019/09/publications-ipri-2019-02.pdf. [Accessed: 27-Apr-2022].

[3] J. Demarest, “Taking down botnets,” *FBI*, 15-Jul-2014. [Online]. Available: https://www.fbi.gov/news/testimony/taking-down-botnets. [Accessed: 27-Apr-2022].

[4] Y. Namestnikov, “The economics of botnets,” *Kaspersky Lab*, Jul-2009. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2009/07/01121538/ynam_botnets_0907_en.pdf. [Accessed: 27-Apr-2022].

[5] “What is a botnet?,” *Palo Alto Networks*. [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-botnet. [Accessed: 27-Apr-2022].

[6] O. Yoachimik, “Cloudflare thwarts 17.2m RPS ddos attack - the largest ever reported,” *The Cloudflare Blog*, 19-Aug-2021. [Online]. Available: https://blog.cloudflare.com/cloudflare-thwarts-17-2m-rps-ddos-attack-the-largest-ever-reported/. [Accessed: 27-Apr-2022].

[7] “What is a DDoS botnet?,” *Cloudflare*. [Online]. Available: https://www.cloudflare.com/en-au/learning/ddos/what-is-a-ddos-botnet/. [Accessed: 27-Apr-2022].

[8] “Tor,” *Tor Project: Overview*. [Online]. Available: https://2019.www.torproject.org/about/overview.html.en. [Accessed: 27-Apr-2022].

[9] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The Second-Generation Onion Router”, in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, San Diego, CA, 2004, p. 21.

[10] O. Fajana, G. Owenson, and M. Cocea, “TorBot stalker: Detecting tor botnets through Intelligent Circuit Data Analysis,” *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018.

[11] M. B. B. D. Robles, J. M. Samaniego, and J. A. C. Hermocilla, “Characterization and Classification of Malware Traffic over the Tor Network”, in *Philippine Computing Science Congress (PCSC2020), Baguio City, Philippines*, 2020, 10 Pages.

[12] Z. Ling, J. Luo, K. Wu, W. Yu and X. Fu, "TorWard: Discovery of malicious traffic over Tor," IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 1402-1410, doi: 10.1109/INFOCOM.2014.6848074.

[13] “History,” *Tor Project*. [Online]. Available: https://www.torproject.org/about/history/. [Accessed: 27-Apr-2022].

[14] A. Tiwari, “Tor explained: What is tor? how does it work? is it illegal?,” *Fossbytes*, 23-Feb-2021. [Online]. Available: https://fossbytes.com/everything-tor-tor-tor-works/. [Accessed: 27-Apr-2022].

[16] D. L. Huete Trujillo and A. Ruiz-Martínez, “Tor Hidden Services: A systematic literature review,” *Journal of Cybersecurity and Privacy*, vol. 1, no. 3, pp. 496–518, 2021.

[17] “How Do Onion Services Work?,” *Tor Project*. [Online]. Available: https://community.torproject.org/onion-services/overview/. [Accessed: 27-Apr-2022].

[18] B. Evers, J. Hols, E. Kula, J. Schouten, M. den Toom, R.M. van der Laan and J.A. Pouwelse, “Thirteen Years of Tor Attacks”, *Delft University of Technology, The Netherlands*. [Online]. Available: https://github.com/Attacks-on-Tor/Attacks-on-Tor. [Accessed: 27-Apr-2022].

[19] P. Sirinam, M. Imani, M. Juarez, and M. Wright, “Deep fingerprinting,” *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[20] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas, “Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services”, In *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 287–302.

[21] I. Ghafir, V. Prenosil, and M. Hammoudeh, “Botnet Command and Control Traffic Detection Challenges: A Correlation-based Solution”, *International Journal of Advances in Computer Networks and Its Security– IJCNS*, v. 7, pp. 2250–3757, 04 2017.

[22] “IOT botnet,” *Trend Micro*. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/definition/iot-botnet. [Accessed: 27-Apr-2022].

[23] “What is the Mirai botnet?,” *Cloudflare*. [Online]. Available: https://www.cloudflare.com/en-au/learning/ddos/glossary/mirai-botnet/. [Accessed: 27-Apr-2022].

[24] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, “Botnets: A survey”, *Computer Networks*, pp. 378–403, 2013.

[25] A. Tyagi and A. Gnanasekaran, “A Wide Scale Survey on Botnet”, *International Journal of Computer Applications*, V. 34, 01 2011.

[26] “What is a botnet?: CrowdStrike,” *crowdstrike.com*, 12-Jan-2022. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/botnets/. [Accessed: 27-Apr-2022].

[27] J. Grizzard, V. Sharma, C. Nunnery, B. Kang, and D. Dagon, “Peer-to-Peer Botnets: Overview and case study”, pp. 1–1, 01 2007.

[28] “What is DNS fast flux?,” *Cloudflare*. [Online]. Available: https://www.cloudflare.com/learning/dns/dns-fast-flux/. [Accessed: 27-Apr-2022].

[29] R. Borgaonkar, “An Analysis of the Asprox Botnet”, 07 2010.

[30] P. Arntz and ABOUT THE AUTHOR Pieter Arntz Malware Intelligence Researcher , “Explained: Domain generating algorithm,” Malwarebytes Labs, 06-Dec-2016. [Online]. Available: https://blog.malwarebytes.com/security-world/2016/12/explained-domain-generating-algorithm/. [Accessed: 27-Apr-2022].

[31] H. R. Zeidanloo, Mohammad Jorjor Zadeh Shooshtari, Payam Vahdani Amoli, M. Safari, and M. Zamani, “A taxonomy of botnet detection techniques,” 2010 3rd International Conference on Computer Science and Information Technology, 2010.

[32] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, “A taxonomy of botnet behavior, detection, and Defense,” IEEE Communications Surveys & Tutorials, vol. 16, no. 2, pp. 898–924, 2014.

[33] “What is a honeypot? how it increases security,” *Rapid7*. [Online]. Available: https://www.rapid7.com/fundamentals/honeypots/. [Accessed: 27-Apr-2022].

[34] “Intrusion detection system (IDS): Signature vs. anomaly-based,” *N-able*, 05-Mar-2021. [Online]. Available: https://www.n-able.com/blog/intrusion-detection-system. [Accessed: 27-Apr-2022].

[36] D. Brown, “Resilient Botnet Command and Control with Tor,” *YouTube*, 08-Nov-2013. [Online]. Available: https://www.youtube.com/watch?v=X7kkwOjsFQA. [Accessed: 27-Apr-2022].

[37] Nex, “Skynet, a tor-powered botnet straight from reddit: Rapid7 blog,” *Rapid7*, 06-Dec-2012. [Online]. Available: https://www.rapid7.com/blog/post/2012/12/06/skynet-a-tor-powered-botnet-straight-from-reddit/. [Accessed: 27-Apr-2022].

[38] O. C. A. Abendan II, “MEVADE serves adware, hides using SSH and tor,” *MEVADE Serves Adware, Hides Using SSH and Tor - Threat Encyclopedia*, 14-Oct-2013. [Online]. Available: https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/130/mevade-serves-adware-hides-using-ssh-and-tor. [Accessed: 27-Apr-2022].

[39] “BrickerBot malware emerges, permanently bricks IOT devices,” *Trend Micro*, 19-Apr-2017. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/brickerbot-malware-permanently-bricks-iot-devices. [Accessed: 27-Apr-2022].

[40] S. Makoto, “Keeping a hidden identity: Mirai c&cs in Tor Network,” *Trend Micro*, 31-Jul-2019. [Online]. Available: https://www.trendmicro.com/en_us/research/19/g/keeping-a-hidden-identity-mirai-ccs-in-tor-network.html. [Accessed: 27-Apr-2022].

[41] M. Casenove and A. Miraglia, “Botnet over tor: The illusion of hiding,” *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, 2014.

[42] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun and A. A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features", In the *Proceeding of the 3rd International Conference on Information System Security and Privacy, SCITEPRESS, Porto, Portugal*, 2017.

[43] Mila, “Skynet tor botnet / trojan.Tbot samples,” *contagio malware dump*, 27-Dec-2012. [Online]. Available: https://contagiodump.blogspot.com/2012/12/dec-2012-skynet-tor-botnet-trojantbot.html. [Accessed: 27-Apr-2022].

[44] “Public PCAP files for download,” *Netresec*. [Online]. Available: https://www.netresec.com/?page=PcapFiles. [Accessed: 27-Apr-2022].

[45] N. Moustafa, “The bot-IOT dataset,” *The Bot-IoT Dataset*. [Online]. Available: https://research.unsw.edu.au/projects/bot-iot-dataset. [Accessed: 27-Apr-2022].

[51] S. García, M. Grill, J. Stiborek, and A. Zunino, “An empirical comparison of botnet detection methods,” *Computers & Security*, vol. 45, pp. 100–123, 2014.

[52] Whonix, “Design and goals,” *Whonix*, 15-Nov-2022. [Online]. Available: https://www.whonix.org/wiki/About. [Accessed: 23-Nov-2022].

[53] I. Karunanayake, “Data collection for Tor Botnets”, 2020.

[54] “Research Safety Board: Tor project,” *Tor Project*. [Online]. Available: https://research.torproject.org/safetyboard/. [Accessed: 23-Nov-2022].

[55] O. Caspi, “Botenago Strikes again - malware source code uploaded to github,” *AT&T Cybersecurity*. [Online]. Available:

https://cybersecurity.att.com/blogs/labs-research/botenago-strike-again-malware-source-code-uploaded-to-github. [Accessed: 23-Nov-2022].

[56] “How it Works,” *How it works*. [Online]. Available: https://byob.dev/docs. [Accessed: 23-Nov-2022].

[57] Jgamblin, “Mirai-Source-Code,” *GitHub*. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code. [Accessed: 23-Nov-2022].

[59] M. Antonakakis et al., “Understanding the Mirai Botnet,” in 26th USENIX Security Symposium (USENIX Security 17), Aug. 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[60] Glowman554, “Can't compile bot binary · issue #1 · GLOWMAN554/mirai,” *GitHub*, 12-Aug-2022. [Online]. Available: https://github.com/Glowman554/mirai/issues/1. [Accessed: 23-Nov-2022].

[61] A. Burt, “Gitpaste-12: A New Worming botnet with reverse shell capability spreading via github and Pastebin,” *Official Juniper Networks Blogs*, 03-Mar-2021. [Online]. Available: https://blogs.juniper.net/en-us/threat-research/gitpaste-12. [Accessed: 23-Nov-2022].

[62] G. Ye and Alex.Turing, “The awaiting Roboto botnet,” *360 Netlab Blog - Network Security Research Lab at 360*, 20-Nov-2019. [Online]. Available: https://blog.netlab.360.com/the-awaiting-roboto-botnet-en/. [Accessed: 23-Nov-2022].

[63] J. Hayes and G. Danezis, “Better open-world website fingerprinting,” CoRR, vol. abs/1509.00789, 2015. [Online]. Available: http://arxiv.org/abs/1509.00789.

[64] J. Hayes, “K-FP: Benchmarks for the K-FP WF attack,” *GitHub*, 25-Aug-2015. [Online]. Available: https://github.com/jhayes14/k-FP. [Accessed: 23-Nov-2022].

[65] A. Amin, “PCAP,” *OneDrive*, 21-Nov-2022. [Online]. Available: https://unsw-my.sharepoint.com/personal/z5018626_ad_unsw_edu_au/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fz5018626%5Fad%5Funsw%5Fedu%5Fau%2FDocuments%2FPCAP. [Accessed: 23-Nov-2022].

Appendix A

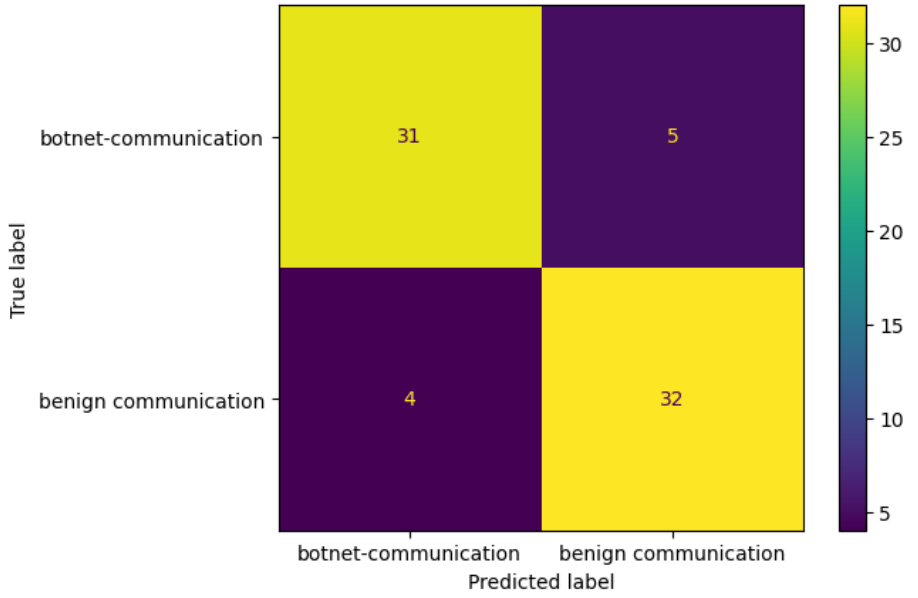

Figure A1: Random Forest Classifier Confusion Matrix

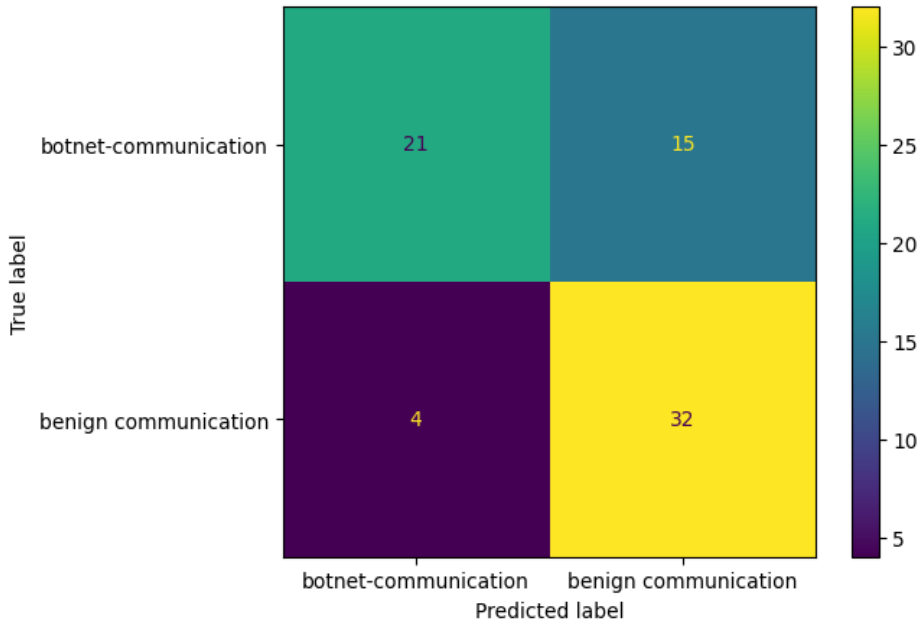

Figure A2: Gaussian Naive Bayes Confusion Matrix

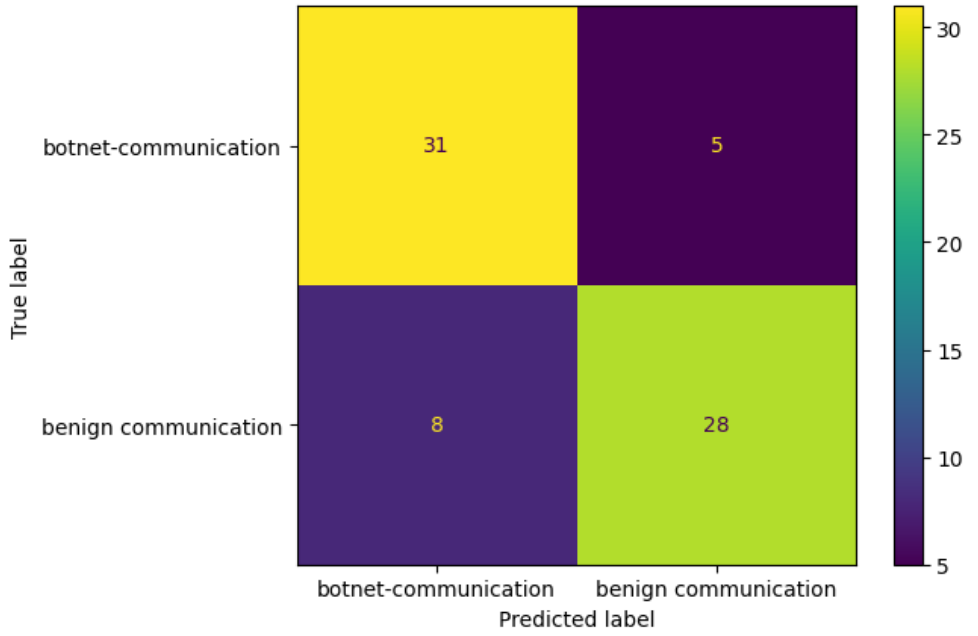

Figure A3: Decision Tree Confusion Matrix

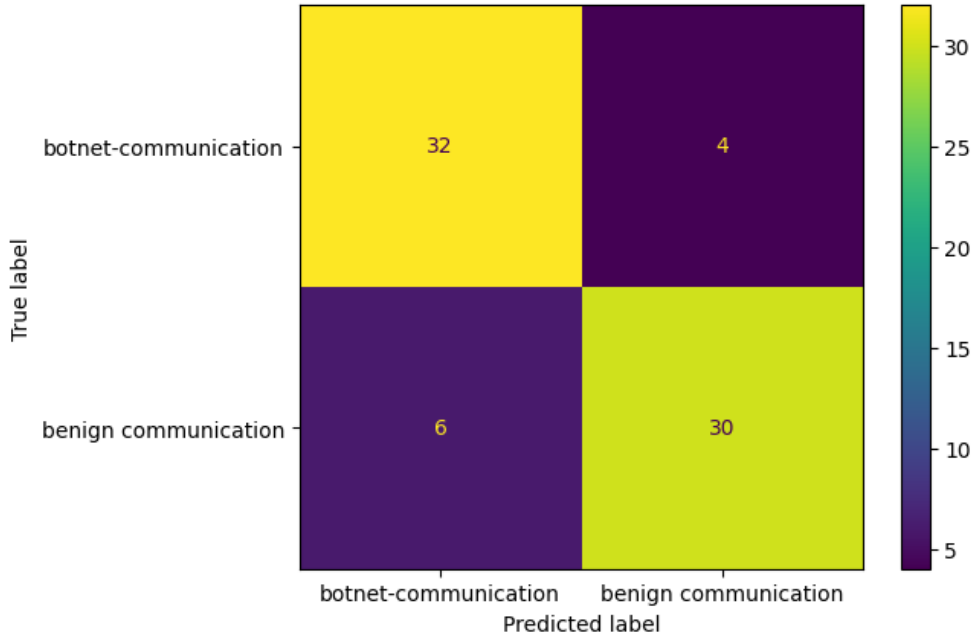

Figure A4: AdaBoost Confusion Matrix

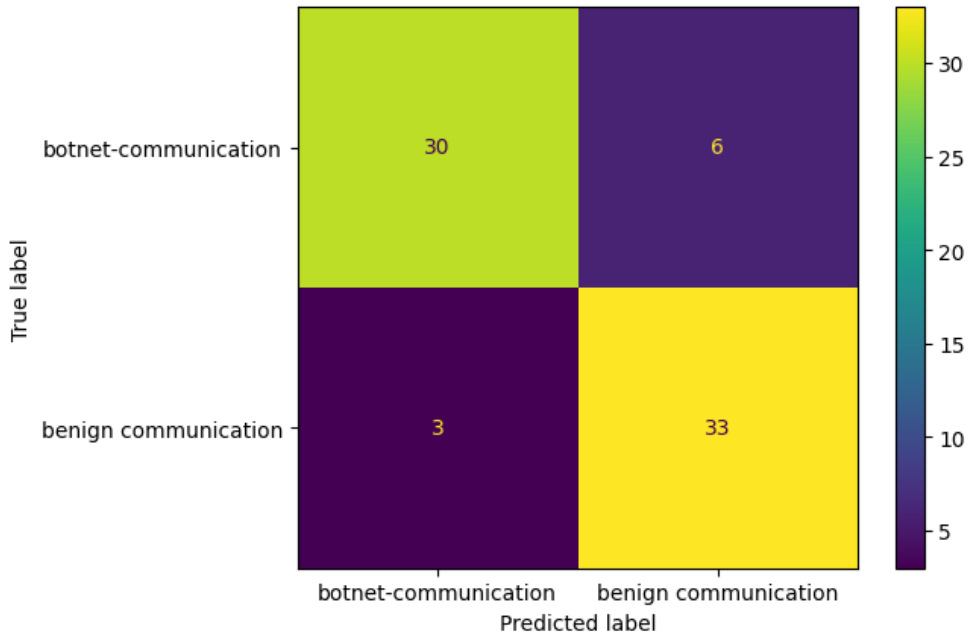

Figure A5: XGBoost Confusion Matrix

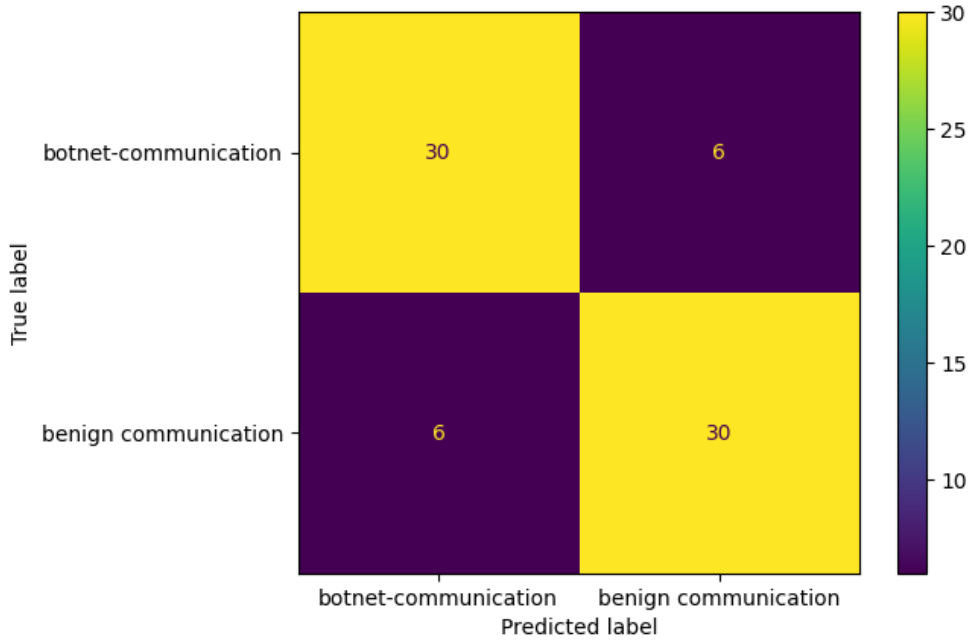

Figure A6: LightGBM Confusion Matrix