2022

An Analysis on Network Flow-Based IoT Botnet Detection Using Weka

Cian Porteous
Technological University Dublin

An Analysis on Network Flow-Based IoT Botnet Detection Using Weka

Cian Porteous

A dissertation submitted in partial fulfilment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computer Science (Advanced Software Development)

05/01/2022

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Advanced Software Development), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: Cian Porteous

Date: **05 January 2022**

ABSTRACT

Botnets pose a significant and growing risk to modern networks. Detection of botnets remains an important area of open research in order to prevent the proliferation of botnets and to mitigate the damage that can be caused by botnets that have already been established. Botnet detection can be broadly categorised into two main categories: signature-based detection and anomaly-based detection. This paper sets out to measure the accuracy, false-positive rate, and false-negative rate of four algorithms that are available in Weka for anomaly-based detection of a dataset of HTTP and IRC botnet data. The algorithms that were selected to detect botnets in the Weka environment are J48, naïve Bayes, random forest, and UltraBoost. The dataset was generated using a realistic network environment by The University of New South Wales, Canberra. The findings showed that botnet behaviours from the selected dataset could be detected by Weka with a high degree of accuracy and low false-positive rate. With all features included, the random forest algorithm was found to achieve the highest accuracy with 96.70%, and the algorithm that attained the lowest false-positive rates was also random forest with 0.008. With a reduced feature set of IP addresses and ports, the random forest algorithm attained the highest accuracy and precision and lowest false-positive rate. With only information regarding packets per second being sent and received, J48 was this time the most accurate with its predictions and attained the highest precision.

Key words: *Security, Botnet Detection, Weka, Internet of Things, Data Mining*

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to everyone who has helped me in the course of my Master's studies. In particular, I would like to express my gratitude to my supervisor Dr. Aneel Rahim for making this research project possible. His advice and regular meetings guided me throughout all stages of designing and writing this project and he provided me with essential feedback to ensure that this project was moving in the right direction.

I would also like to thank my parents Maureen and Paul, my partner Susannah, and my friends for their continuous support and encouragement throughout the duration of my postgraduate studies.

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

1.1 Background

The term botnet refers to an overlay network composed of computer endpoints that are managed by a central botmaster who exercises control over other nodes (termed bots) on the network, usually for malicious purposes (Xing et al., 2021). Botnets pose an ongoing threat to the integrity of businesses, governments, and individuals alike. The types of attacks carried out by these networks include distributed-denial-of-service (DDoS), identity theft, phishing, spam, and covert cryptocurrency mining using stolen computing resources (Stone-Gross et al., 2009).

Botnets originally came about in the 1990's through the exploitation of communication links in Internet Relay Chat (IRC) and, over the decades since then, botnet architectures have diversified and become increasingly sophisticated with three chief categories of architectures having been identified (Tyagi & Aghila, 2011). Centralised botnet architectures consist of a singular command and control (C&C) server from which the botmaster can supervise and govern the various nodes under its control. More often these days, decentralised and hybrid architectures are being utilised by botmasters which result in botnets that spread more easily and are more difficult to detect.

Detection of botnets can be difficult due to their varied and impermanent nature, and several detection methods have been developed. Although several taxonomies of botnet detection systems have been devised, two distinct categories are generally given: honeynet based detection and Intrusion Detection System (IDS) based detection systems (Khattak et al., 2014). The former is a form of active detection whereby the host becomes part of the botnet and studies the botnet from within its network. Intrusion Detection systems are a passive approach and can perform the necessary analysis to detect the botnet without itself becoming compromised.

1.2 Project Description

This project sets out to investigate the effectiveness of Weka in being able to recognize botnets in two datasets. Several botnet families will be under investigation. The algorithms used in Weka to conduct this research are J48, Naïve Bayes, Random Forest, and UltraBoost.

Weka has been used in research before, but the combination of classification algorithms used in this research have not been tested before, nor is there any evidence of the dataset used in this research having been used by Weka before. Many of the datasets used in prior research involving Weka are several years older than the dataset used in this study. The dataset selected for this study was created by the University of New South Wales (UNSW) Canberra, and it contains data of both malicious and benign network traffic flows making it an advantageous option for the study of botnet detection in realistic situations.

IRC-based botnets tend to be of a centralised nature whereas P2P-based botnets are decentralised as each host on the platform is capable of operating as both bot and server (Zhao et al., 2013). This research sets out to assess the botnet detection capabilities of certain classifier algorithms to detect a wide variety of botnets, including centralised and P2P, and those that communicate via HTTP and IRC.

All of the botnet detection algorithms and the data mining environment, Weka, that are used in this study are free and open-source so that the findings may benefit the largest possible demographic. It is assumed that the botnets in the dataset compiled by UNSW Canberra contains accurate information and a balanced spread of botnets so that the findings of this research will be a reliable indication of the algorithms being tested.

1.3 Research Project

This study wants to contribute to the growing body of research that investigates the effectiveness of botnet detection so that users can be better informed as to the strengths

and weaknesses of botnet detection frameworks that are available to them. *The primary objective of this dissertation is to determine whether four classification algorithms used in Weka can achieve a high level of accuracy and a low false-positive rate in the detection of IoT bots in a recently created dataset.* The results of the experiment will then be discussed and will be compared with conclusions as to their feasibility and impact. The problem will be analysed and addressed with a literature review.

1.4 Research Question

"Can the Naïve Bayes, Random Forest, J48, and UltraBoost algorithms used in Weka achieve a high level of accuracy and a low false-positive rate in the detection of IoT bots in a recently created dataset?".

1.5 Research Aims and Objectives

The primary aim of this research project is to provide users with more information about the effectiveness of Weka at detecting a selection of IoT bots using captured packets of their network-flows.

1. To determine the accuracy, false-positive rate, and false-negative rate of Weka when using the J48, regression tree, binary forest, and naive Bayes algorithms for a dataset of IoT bots.
2. To test these algorithms in the Weka environment for their effectiveness at detecting botnets when various features have been selected.
3. To determine which algorithm is the most effective approach in correctly identifying IoT bots.
4. To identify topics for research that might improve and expand on the knowledge in this domain.

1.6 Research Methodologies

Null Hypothesis (H0)

Weka cannot be used to successfully detect Internet of Things botnets using at least one of four of its inbuilt classification algorithms, namely J48, Naïve Bayes, Random Forest, and UltraBoost, with 95% prediction accuracy.

Alternate Hypothesis (H1)

Weka can be used to successfully detect Internet of Things botnets using at least one of four of its available classification algorithms, namely J48, Naïve Bayes, Random Forest, and UltraBoost, with 95% prediction accuracy.

Objective (O1)

The objective of the research is to show that Weka can successfully detect botnets from a recent dataset containing simulated Internet of Things botnet traffic.

Objective (O2)

It is an objective to assess the four algorithms being tested and to compare their respective accuracy, false-positive rate, precision, and execution time with each other and other approaches in the literature to determine which is the most effective and efficient for predictions about the selected dataset.

Objective (O3)

The final research objective is to assess the various algorithms with regards to their accuracy, false-positive rate, precision, and execution time when different sets of features have been selected as attributes.

Objective 1 will be completed by investigating the accuracy of Weka at correctly classifying botnets in the selected dataset. The accuracy will be evaluated using four algorithms – J48, UltraBoost, Naïve Bayes, and Random Forest. The results will then

be analysed to give a clear conclusion as to whether using Weka is a feasible approach in the field of botnet detection.

Objective 2 will be completed by comparing the results of each algorithm to determine which can provide the greatest accuracy precision, the lowest false-positive rate and the shortest execution time for the given dataset. These values will be compared to the results of other research papers, particularly papers that focus on the same tool Weka or that use the same dataset as was used in this paper – the Bot-IoT dataset.

Objective 3 will be completed by conducting the experiments three times. The first experiment will include all of the features that are available in the dataset. The second experiment will only include information about IP addresses and port numbers. The third experiment will only take the rate of packet transfer from source to destination and the rate of packet transfer from destination to source into account.

1.7 Scope and Limitations

The scope of this project is an analysis on botnet detection frameworks that are open-source. While commercial frameworks may provide greater effectiveness with their accuracy and a lower false-positive rate overall, commercial frameworks by their nature tend to hide the source code and functionality of their detection mechanism in order to prevent attackers from using this knowledge when designing malware. However, the costs required to use them can be prohibitive which leads to a reduced reach with regards to their impact.

Only four algorithms are being tested, namely J48, Naïve Bayes, UltraBoost, and Random Forest. These algorithms were selected due to both their availability in the Weka environment and their popularity among researchers. The UltraBoost algorithm is probably the least well known of the four given that it has mostly been used in medical diagnostic research up until now, and its effectiveness at detecting botnets will be contrasted to the more well known algorithms that are under investigation.

However, there is a notable scarcity of literature surrounding it and its performance in this field.

Weka is a popular tool among researchers for small to medium datasets, but as datasets get larger and more complex, Weka begins to slow down due to it being a memory-intensive approach. The heap data structure may need to be manually increased beyond its default size in the Java Virtual Machine that is running Weka, and the dataset that has been selected, being 16.7 GB in size and containing over 72 million records will likely need to be truncated in order for Weka to be able to perform the necessary algorithms in a reasonable amount of time.

1.8 Organisation of Dissertation

The rest of this dissertation is organised into the following chapters:

Chapter 2 - Literature review and related work

The state of the art in this field will be critically evaluated. The literature review will provide a thorough background on the various types of botnet, their communication patterns, and their topologies. The state of the art will be discussed, along with the main approaches that are used in detecting botnets. The effectiveness of various botnet detection frameworks and the features that they utilise along with their mechanism of action will be analysed. Some applications of the software used in this research – Weka – will be outlined from approaches that other researchers have attempted previously. The gaps in the research will be explored at the end of the chapter which will elicit the research question.

Chapter 3 - Design / methodology

The focus of this chapter will be to explain in detail the structure of the experimentation, explaining the technologies used, the steps involved, and the specifications of the software and hardware used in the research. The datasets themselves will be explained in detail, as well as basic instructions about getting the

data into the correct format so that they can be used by Weka for the experiments. The aim of this chapter is to make each step of the experiment replicable so that the research is reproducible.

Chapter 4 - Implementation / results

The actual implementation of the experiment will be given, including how the data was prepared, how the experiment was conducted, with the rationale behind each step and any setbacks that may occur during the experiment. A brief overview of the results will be given at the end of the chapter.

Chapter 5 - Evaluation / analysis

The results from the experiment will be highlighted in the context of how they pertain to the research question. The findings will be analysed in detail and a discussion will be had in relation to previous research that used Weka or the dataset used in this paper and whether the results from this research project are in line with what was expected from prior investigations.

Chapter 6 - Conclusion

The final chapter will review the previous chapters and contain an overview of the work that was carried out. The contribution and impact of this research will be discussed and some suggestions for future work will be provided to conclude the paper.

2. LITERATURE REVIEW AND RELATED WORK

This literature review will provide essential prerequisite background information regarding botnets, including their methods of communication, topologies, and life cycles. It will then highlight the state of the art in botnet detection research and the commonest approaches in detecting them.

2.1 Types of Botnet

Botnets are a dynamic and continuously evolving threat, and can be categorised according to several metrics, including their functionality, communication method, architecture, the type of system they target (e.g., Windows or Linux), the vulnerability they exploit, and whether they are controlled manually or operate autonomously.

2.1.1 Forms of Botnet Communication

Communication is a central aspect of the botnet's life cycle. Botnets rely on communication so that they can scan networks, send data to the botmaster, and propagate to increase the size of the botnet. The two main forms of communication used by botnets are Internet Relay Chat (IRC) and HyperText Transfer Protocol (HTTP)

IRC

IRC botnets operate by setting up an IRC client on a compromised computer, which can be used to communicate between the bot and the botmaster in setting up a Command and Control (C&C) terminal to issue commands. An IRC bot is easy to set up and propagate as all it requires is an IRC script and server to function. A central feature of IRC botnets is the use of IRC in the C&C. The benefit and drawback of this feature come from the ease with which a central command terminal can issue commands, and the vulnerability of the botnet once this C&C terminal becomes incapacitated (Grizzard, 2007).

HTTP

There are several reasons why botnets that communicate through HTTP are advantageous for botmasters. HTTP botnets are less common than IRC botnets and therefore tend to be better at evading detection as less research and attention has been directed towards monitoring and detecting them. However, more botnets are beginning to use HTTP due to it being less common and also due to its communication being more difficult to detect.

Most organisations have firewalls that automatically block network traffic that is directed to ports that are not in use, including those that are frequently targeted by IRC botnets (Trend Micro, 2006). This effectively prevents communication between IRC botnets and botmaster, but firewall security parameters may be more easily bypassed by botnets that operate through HTTP communication.

2.1.2 Architecture

Botnet architectures are generally categorised according to their topology of which two main classes exist: a traditional centralised architecture or more recent decentralised architectures which communicate on a peer-to-peer basis:

Centralised

The centralised system was once the most popular way of organising a botnet, but it is becoming less common due to it being more vulnerable to being disrupted and the reduced anonymity of the botmaster whose identity may be compromised by following network traffic.

Centralised models consist of many nodes where one entity, which can be one or a small number of hosts, is in control of all of the other nodes on the network. The biggest threat to a centralised architecture is that it is susceptible to being taken down if the central node is eliminated. One way in which botmasters attempt to prevent the central node from being taken down is to use fast-flux or dynamic DNS techniques as a way of masking which node is the botmaster (Ramsbrock & Wang, 2013).

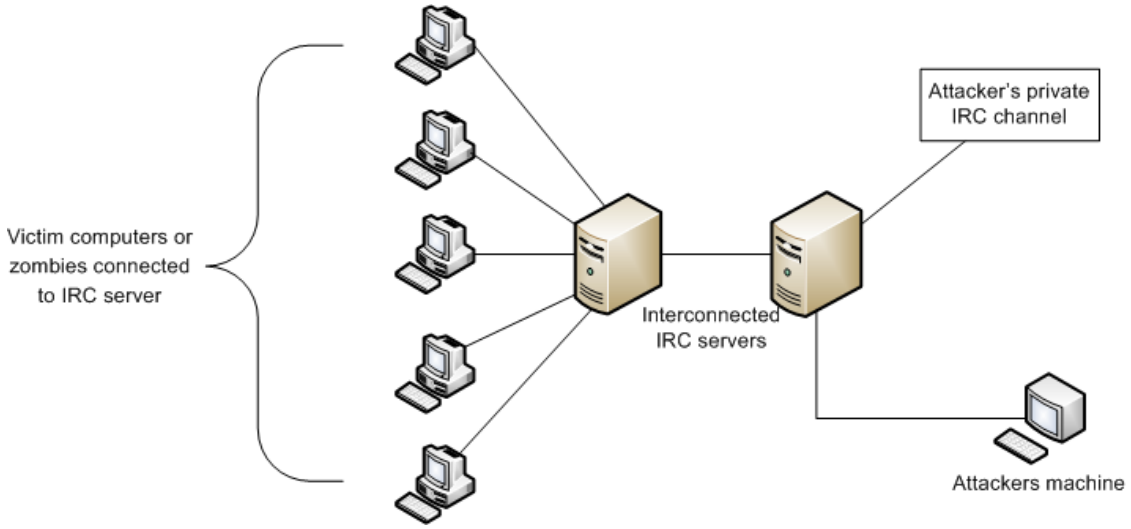

Figure 1. A typical centralised botnet with IRC servers as bots (Alparslan et al., 2012).

Figure 1 shows a typical decentralised botnet with infected computers all communicating with a central IRC server which communicates directly with the botmaster's private IRC channel. Under practical circumstances, the number of victim computers will be many orders of magnitude greater than depicted above.

Peer-to-Peer

Ogu et al. explain how the relatively newer decentralised architecture of botnets have peer-to-peer protocols as their foundation. Botnets employing peer-to-peer topologies do not use a central server or proxy to administer command and control commands to the other nodes on the network. Instead, every node is capable of acting as a command and control server or as a bot that receives commands. Barford & Yegneswaran were the first to suggest that botnets using peer-to-peer forms of architecture would begin to encrypt their communication, which has remained a challenge in botnet detection since it was first developed (2007). As stated by Grizzard et al, in a P2P architecture, there is no centralised point for C&C and bots communicate with other peer bots instead of a central server. Nodes in a P2P network act as both servers and clients. Therefore, there is no centralized coordination point that can be incapacitated. The authors analysed the case study of the "Trojan.Peacomm" bot and observed that the P2P protocol is essentially being used as a name resolution server to upgrade the bot.

Wang et al. pointed out some weaknesses of known P2P bots and proposed a new P2P bot architecture. According to them, botnets such as Sinit, Phatbot, Nugache and Slapper have implemented different kinds of P2P control architectures. A Sinit bot host finds other Sinit bot hosts by using random probing. The extensive probing traffic will make it easy to detect the botnet. Phatbot uses Gnutella cache servers for its bootstrap process. This also makes the botnet easy to be shut down. Nugache relies on a seed list of C&C IP addresses during its bootstrap process. This makes it weak. Slapper does not have encryption and its command authentication enables others to easily hijack it. Keeping in view these weaknesses of P2P botnet architecture Wang et al. propose the design of advanced hybrid P2P botnet architecture, which is much harder to be shut down or monitored. Their hybrid P2P botnet architecture provides robust network connectivity, individualised encryption, controlled traffic dispersion and easy monitoring and recovery by its botmaster. Furthermore, if a bot is captured, the botnet exposure will be limited.

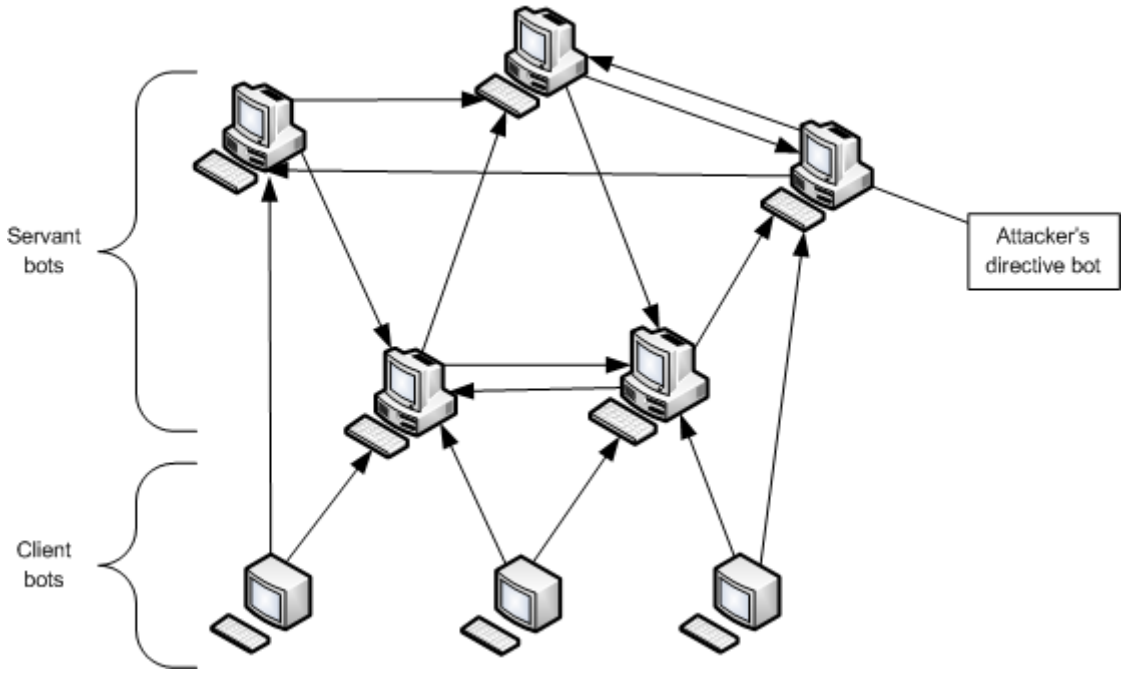

Figure 2. A typical P2P botnet with IRC servers as bots (Alparslan et al., 2012).

Figure 2 displays an architecture diagram for P2P botnets. Unlike IRC or HTTP botnets, any bot in a P2P botnet can publish a command. Therefore, if one is able to

identify a botmaster and bring it down, the P2P botnet will still be functional because any bot can issue botnet commands (i.e. be a botmaster).

2.1.3 Propagation

Propagation is a vital phase in the botnet lifecycle as the majority of botnets benefit from having more hosts on their network. For example, botnets that are involved in DDoS can generate more traffic with a greater number of bots, while botnets that are designed to steal sensitive data have access to a larger number of potential targets the larger the botnet gets. There are two distinct ways that botnets can propagate: active propagation and passive propagation. Active propagation requires the

2.1.4 Modern Botnets

According to this year's Imperva Bad Bot report, 40.8 percent of Internet traffic wasn't human, with 24.1 percent of that being bad bot traffic (2021). 20.1 percent of all bots employed sophisticated tactics to evade detection, such as cycling through IP addresses, use of proxies, and imitating the behaviour of human users. Perhaps surprisingly, the findings of the report showed that the majority of bad bots originate in the United States, which 45.9 percent of bad bot traffic was found to come from. This finding may be as a result of blanket bans on countries such as Russia and China, which were the two most commonly blocked countries in the report. Banning portions of the United States where the bots are found to operate from may be unfeasible for political, economic, or diplomatic reasons.

There is evidence that botnets are becoming more pervasive, with the estimated number of IoT devices infected with Mirai having increased from approximately 143,000 to 226,000 between 2018 and 2019 (*Enisa Threat Landscape*, 2020). IoT devices are increasingly becoming the vectors of passage and attack by botmasters, with a large range of demographics being targeted by people who create botnets, including those who are wanted for political criticism, regular users for their passwords and data, and bots that are designed for espionage being propagated by various organisations to gather data from foreign nations (Thanh Vu et al., 2021).

2.2 Types of Detection

Detection techniques can be divided into two broad categories – honeynet-based detection and intrusion detection (Sudhakar & Kumar, 2019). Honeynet detection systems exploit the mechanism by which the malware attacks in order to "trap" the attacker and then perform the necessary analysis on the malware. Honeypot techniques rely on the use of known software and network vulnerabilities to allow a node under surveillance to be taken in by a botnet, allowing for investigation of the so called "zombie" node. Recently, botmasters have begun to circumvent this method of investigation as some bot binary codes have been altered to detect when they are operating within a virtual machine or a sandbox environment, and bots that detect a suspicious environment can choose not to execute or to change its runtime behaviour to evade detection (Khattak et al., 2014). Intrusion detection systems rely on either anomaly-based detection or signature-based detection.

1. Anomaly-based detection evaluates network activity for unusual or suspicious behaviour such as higher than usual network latency or network volume or activity on certain ports that signal malicious activity (Binkley & Singh, 2006).

2. Signature-based detection works by comparing the bot signatures to those saved on the IDS in order to determine whether a bot might be present (Kugisaki et al., 2007). This approach has a very-low false positive rate but is limited in that it can only detect known bots whose signatures can be compared (Wurzinger et al., 2009).

The majority of modern botnets focus on anomaly-based detection for several good reasons. Firstly, signature-based approaches are completely ineffective against day-zero attacks as the binaries that are required as a comparison do not yet exist in any databases. Secondly, some anomaly-based approaches that detect unusual behaviour are able to overcome forms of botnet concealment such as dynamic IP generation or encrypted messaging. Anomaly-based detection can also identify

abnormal network behaviour such as high traffic, high latency, or uncommon ports being used in an unusual way (Miller & Busby-Earle, 2016).

Signature-based detection relies on distinctive binaries that correspond to specific, known botnets which can be compared to binaries within the network being monitored. Anomaly-based detection has the advantage of being able to discover botnets whose binaries are not yet stored in the framework's database and therefore cannot be readily compared. Signature-based detection has the advantage of requiring very little oversight as every detection correlates to a known set of binaries leading to a negligible false positive rate (Shinan et al., 2021).

Botnet detection remains difficult due to the similarity in appearance of botnets to regular network traffic, and also by the use of encryption algorithms by botnets to make detection based on packet inspection ineffective (Alauthaman et al., 2016)

2.3 The State of the Art

As botnets constitute one of the most threatening and widespread cyberthreats of our era, much research has come about in creating botnet detection frameworks. Botnet detection can be broadly categorised into three main approaches, but some novel concepts have also been developed in an effort to combat botnets in less traditional ways. The three primary categories of botnet detection are machine-learning, Khattak et al (2014) explore the various approaches and considerations surrounding botnet detection techniques. Active detection methods involve becoming part of the botnet itself by allowing a host node to become infected and then surveying the botnet behaviour. Passive botnet detection approaches instead focus on observing the botnet without the host becoming compromised. There are legal and ethical implications relating to active botnet detection as the need to continuously analyse flow information on a network may be in breach of individual or corporate rights.

2.3.1 Early Approaches

BotMiner is a detection framework that exploits the uniformity of behaviours displayed by bots in a botnet network (Zhao et al., 2013), and in so doing can detect a number of different operating botnets, including IRC-based, HTTP-based, and P2P botnets such as Nugache and Storm worm (Gu et al., 2008).

BotDigger takes a fuzzy-logic approach in detecting bots by creating a trust level of the bots in the network. BotDigger ascertains whether a host on a network is a bot in three ways: quantity evidence, temporal evidence, and linguistic evidence (Zhang et al., 2016). These three methods allow for large-scale groups of hosts to be analysed for whether they are a bot. Botdigger is used to detect bots that use domain generation algorithms (DGA). BotDigger is more successful at detecting larger-scale botnets such as university sized networks (Ardi & Heidemann, 2018).

BotHunter is a botnet detection framework that can detect bots based on a typical botnet's infection life cycle (Mahmoud et al., 2013). The detection framework analyses potential bots for distinguishing behaviours such as downloading bot binaries, execution of botnet code, sending or receiving network traffic that might represent C&C communication. There are three engines in BotHunter; Statistical Scan Anomaly Detection Engine (SCADE), Statistical Payload Anomaly Detection Engine (SLADE), and Signature engine. SCADE detects incoming and outbound network traffic for anomalous characteristics. SLADE looks for anomalous byte-distribution payloads.

SNORT is a network-based IDS and intrusion prevention system (IPS). Snort relies on signature-based detection with the obvious drawback being that it is ineffective against day-zero attacks as the malicious packets of data would not yet exist in the database that SNORT would use for comparison (Chakrabarti et al., 2010).

Rishi detects botnets by analysing IRC channel names in incoming network traffic. TCP packets containing headers that are related to the IRC headers are monitored, and

packets that contain the words NICK, JOIN, USER, QUIT, or MODE are marked as suspicious. A number of metrics are extracted from the packets including source and destination ports, source and destination IP addresses, the length of time that the connection lasted, and the names and nicknames of the IRC channel and user. These are then given a score based on whether they are similar to other connections that have been marked as suspicious, and when a sufficiently high score is reached an email is automatically sent to the admin (Göbel & Holz, 2007).

As with many signature-based detection approaches, detection based on likely malware names becomes a contest between attacker and security experts where attackers need to continuously change the names used to avoid detection and security professionals need to keep up with the names used to create a new signature for each instance that is found (Ramsbrock & Wang, 2013).

2.3.2 Machine Learning Approaches

Huancayo Ramos et al. developed a benchmark-based reference model which utilised 5 different machine learning classifiers – Decision Tree, Random Forest, Naive Bayes Gaussian, Support Vector Machine and K-Nearest Neighbours (2020). They used the CSE-CIC-IDS2018 dataset and the ISOT HTTP botnet dataset.

The findings of the experiments showed that the Random Forest and Decision Tree models exhibited the highest levels of accuracy of the methodologies that were tested. However, the datasets used had some limitations. The first dataset, CSE-CIC-IDS2018, contains only two families of botnets and many other types of malware resulting in a more general IDS study but not providing results that are useful to the research of botnets specifically. Although the second dataset, ISOT HTTP Botnet dataset, contains several families of common botnets, it does not contain any benign traffic which leads to a lack of credibility of the framework when dealing with realistic situations in actual networks.

Bhatt & Thakker developed a novel approach for detecting IoT botnets that works by forecasting botnets through anomaly detection (2021). The first step is to develop instance creation and then follow up by cataloguing the instances. They decided to use stream mining for this step instead of machine learning to reduce the time and memory requirements of the method. They ultimately determine whether an instance is a bot through graph structure based detection of anomaly (GSBDA) coupled with a K nearest neighbour algorithm.

2.3.3 Neural Network Approaches

There has been a lot of success in recent years by using neural networks to detect botnets. Alauthaman et al. used a joint classification and regression tree algorithm and neural network to detect peer-to-peer botnets with high accuracy and low false positive rates (2016). Their technique involved supervised learning and feature extraction from TCP control packet headers while the hosts are connected for a time period of 30 seconds. This approach cannot be implemented in real time, and the authors proposed that they would in the future attempt to detect botnets in an unsupervised way to extend the scope of their detection framework to devices that require immediate detection.

Li & Wang developed a neural network method that makes use of back propagation (BP) of traffic characteristics (2018). They worked on data that had been collected from the campus network of Sichuan University which provided a variety of benign traffic along with simulated botnet traffic which was injected into the database. They included six features in their experimentation including time interval, number of bytes, number of data packets, and duration. When the expected error was between 0.001 and 0.007 the false positive rate was zero, but with a higher expected error the false positive rate began to grow. The findings also showed that high accuracy and precision were attained by the proposed BP method.

Success at detecting botnet has also been achieved through the development of deep recurrent neural networks (DRNN) with synthetic minority oversampling technique

(SMOTE) by Popoola et al. which utilised hierarchical feature representation on network traffic flows resulting in very high precision, recall, and F1 scores on the dataset that was used (2021). Convolutional neural networks have also been successful at detecting botnets that run on Android devices, but experiments showed that on the ISCX botnet data sample, deep learning models outperformed CNN and the other traditional machine learning classifiers that were compared (Yerima et al., 2021).

Another deep learning approach which has seen recent success is the work conducted by McDermott et al. in developing a Bidirectional Long Short Term Memory based Recurrent Neural Network (2018). The results were promising with a significantly high accuracy yield of 99%, 98% and 98% when trying to detect Mirai, UDP, and DNS traffic respectively. The botnet dataset that was used was generated in a lab setting and is publicly available to researchers on request.

2.3.4 Other Approaches

Botnet detection strategies are numerous, with recent developments incorporating state-of-the-art technologies and a range of methods including deep learning, complex networks, swarm intelligence, software defined networking (SDN), moving target defence (MTD), and blockchain (Xing et al., 2021), with transfer learning emerging as a promising area of research (Alothman & Rattadilok, 2017). Wazzan et al. performed a meta-analysis over four years concluding that a combination of multiple levels of detection would be an impactful approach to conducting future botnet detection research (2021). It was also found that the majority of botnet detection methods are focused on late-phase botnets in the botnet lifecycle, and that early-phase detection would be useful.

Winter et al. devised a method to detect malicious activity through measuring abrupt changes in network entropy time series (2011). The idea behind this method is that a network attack will result in a measurable change in network attribute values which can be detected using network entropy time series. This framework is effective at identifying network anomalies that are otherwise undetectable solely through the

analysis of network flows, and is useful for those administering large computer networks in the face of common potential attacks such as DDoS and malware worms.

Artificial Immune Systems (AIS) are a method of detecting malicious activity that is based on the functioning of the human immune system – an exemplary intrusion detection system. Visconti and Tahayori used an interval type-2 fuzzy set paradigm - a fuzzy logic system that can tolerate a higher degree of certainty - to create an AIS which includes the use of computational white blood cells which can search for, recognize, and store or inhibit unwanted behaviours on host computers (2011).

Chaeikar et al. proposed a framework for use in P2P botnets with 4 main components: filtering, traffic monitoring, malicious activity detector and an analyzer (2010). Irrelevant traffic is first removed through filtering to reduce the workload for the more compute-intensive portions of the framework. The framework then identifies hosts that are exhibiting similar patterns of behaviour and communication in the traffic monitoring phase. The tool uses SNORT's scan detection engine, SCADE, to determine whether the suspicious hosts are engaged in typical botnet scanning and spam-related behaviours. The framework then analyses the hosts to determine if they have appeared in the results of earlier detection efforts.

Wuchner et al. used a compression-based graph mining approach (2019). This involved generating quantitative data flow graphs (QDFG's) from system call traces and extracting characteristic patterns in the form of subgraphs that are used to create a repository. The subgraphs in this repository can then be pattern matched to those found in unknown malware. The data retrieved from pattern matching to known malware is used to train classifiers.

2.4 IoT Botnets

Mirai was the first major botnet that targeted and propagated through IoT devices, many of which exist on the edge where fewer security measures are in place and where the hardware of edge devices itself is more susceptible to attack (Eustis, 2019). A

botnet detection method that was designed specifically for edge IoT devices was proposed by Tzagrkarakis et al. which focuses on detecting attacks in real-time while being able to function with only a limited amount of training and testing data which mimics situations in actual IoT attacks (2019). This is achieved through a sparse representation framework that can operate quickly with low computational requirements, followed by a greedy sparse recovery program which can act with only two features – threshold constant and sparsity level.

Another novel approach used to address IoT device botnets was developed by Abu Kharma et al. through the creation of a wrapper feature selection model by using a hybridization of the salp swarm algorithm and ant lion optimization – two examples of biomimicry applied to cybersecurity. The experiments were conducted on the BaIoT benchmark dataset and results found that their hybrid model was able to detect IoT botnets with a very high true positive rate of 99.9% (2021).

2.5 Weka

Weka is a free and open source machine learning environment written in Java that was developed at the University of Waikato. It provides a broad collection of machine learning tools and algorithms and it has become widely used in data mining research since the project's inception in 1992 (Hall et al., 2009). Some of the more commonly used algorithms in the Weka toolkit include bayesian logistic regression for text classification, best-first decision trees and functional trees, along with a suite of preprocessing filters. Two of the main advantages associated with Weka are its extensibility and its rich selection of extract, transform, and load (ETL) operations which make it compatible with a range of different types of dataset. Weka can be used through Java code or by using the graphical user interface (GUI) that is built into the application. The GUI also provides options for visualisation of the data.

Weka has been used in the field of botnet research before. Susanto et al. used Weka in order to classify several families of IoT botnets (2020). The experimentation was conducted using four machine learning algorithms – AdaBoost, decision tree, random forest, and naïve Bayes. The results showed differential levels of effectiveness: the

random forest method was the most accurate, achieving an accuracy of 100%, decision tree classification achieved 99.99% accuracy, AdaBoost achieved 98.53%, and naïve Bayes achieved the lowest accuracy, with 90.22%. The authors tested the classification accuracy of these algorithms using Scikit-learn tools in the same paper, and Scikit-learn was found to perform worse with every algorithm used except for the decision tree.

Khodamoradi et al. also used Weka, this time finding success in detecting malware of several categories (2015). The malwares detected included botnets, trojans, backdoors, rootkits, bootkits, viruses, and worms using an anomaly-based detection technique. 700 files were analysed: 500 containing healthy files, and 200 infected with malware. The classification algorithms used in evaluating the files were j48, j48grajt, ladtree, random forest, and reptree. The dataset used had five classes of executive files – the first contained 500 samples of healthy files taken from various Windows applications in the Program Files folder, and the remaining virus samples generated using the toolkits Next Generation Virus Creation Kit, Virus Creation Lab, Mass Code Generator, and Second Generation Virus Generator. The authors showed that Weka could be used to accurately detect different forms of malware, but not enough information is provided regarding the specific types of viruses that were generated with the toolkits that they used. This leads to difficulty in reproducing and comparing the results found here as the toolkits used are capable of generating many different types of malware files.

Hao et al. performed unsupervised detection of botnets with frequent pattern tree mining using Weka (2021). The proposed model is made up of three distinct steps: packet collection processing, rule mining, and statistical analysis of rules. They tested three algorithms – frequent pattern tree mining, random forest, and Bayesian network. The research was carried out using the PeerRush dataset from 2014 which contains three different botnet families – Zeus, Maledac, and Storm. When doing frequent pattern tree mining, a major determinant of success was to have the support level sufficiently high. With increasing support level, the accuracy grew to 100% and the false-negative rate became negligible.

Weka has also been used for less common forms of botnet detection, with a conversation-based botnet detection method that was developed by Chen et al. giving positive results (2017). A conversation-based feature selecting module was created which resulted in three conversation features being selected: duration time of flows in conversation, distribution of flows in conversation, and distribution of small packets in conversation. The researchers used the CTU-13 dataset which is a labelled dataset containing botnet, normal and background traffic (*The Ctu-13 Dataset, 2011*), and their findings showed that the conversion-based approach attained higher accuracy and a lower false-positive rate than the traditional flow-based detection approach that they attempted.

2.6 Botnet Datasets

The CSE-CIC-IDS2018 dataset was created through the collaboration of the Communications Security Exchange (CSE) and the Canadian Institute for Cybersecurity (CIC). The dataset was generated through virtualization software which was used to record the raw network packet flows of several simulated communication protocols: Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP3), Internet Message Access Protocol (IMAP), Secure Shell (SSH), and File Transfer Protocol (FTP). The dataset is predominantly made up of HTTP and HTTPS traffic. This dataset is limited to only two families of botnets present in the data, with simulations of the Ares botnet engaged in remote upload/download, remote shell and capturing behaviour. Traces of the Zeus botnet is also present which is a Trojan horse malware package that is generally spread by unintended downloads and phishing attacks. The rest of the files represent other forms of malicious attacks that are typically detected by intrusion detection systems (IDS), such as DDoS, Heartbleed, brute forcing, and infiltration attacks (*CSE-CIC-IDS, Canadian Institute for Cybersecurity,* 2018).

The Bot-IoT dataset was generated at UNSW Canberra by making a realistic network environment that combined normal network traffic with abnormal botnet traffic (Koroniotis et al, 2019). The dataset incorporates a wide range of botnet behaviours, including attacks such as DDoS and DoS, theft behaviours such as keylogging and data exfiltration, and simple service scans.

The ISOT HTTP botnet dataset is the result of research conducted by Alenazi et al. from monitoring network traffic in a virtual environment (2017). The first dataset is used in the present study and it consists of only malicious DNS traffic that was generated by several different families of HTTP botnets – Zeus, Zyklon, Blue, Liphyra, Gaudox, Blackout, Black Energy, and Citadel. Bots were deployed for several days on a virtual machine running Windows XP between 14/06/2017 and 21/06/2017, with all communication between the DNS server and the router being monitored and collected. The packet capture (PCAP) files were generated using Wireshark. Each bot had its own specific IPv4 address and Command and Control (C&C) server set up. For example, the Zyklon botnet was designated the IP address 192.168.50.14 with a DNS C&C server called zyklon.botnet.isot. The full details of each botnet's designated IPv4 address and C&C name is tabulated below:

192.168.50.14	zyklon.botnet.isot
192.168.50.15	blue.botnet.isot
192.168.50.16	liphyra.botnet.isot
192.168.50.17	gaudox.isot.gdox gdox.botnet.isot dox.botnet.isot
192.168.50.18	blackout.botnet.isot
192.168.50.30	citadel.botnet.isot
192.168.50.31	citadel.botnet.isot
192.168.50.32	be.botnet.isot black energy
192.168.50.34	zeus.botnet.isot

Table 1: List of botnets and their corresponding C&C servers

Citadel occurs twice in the list as the authors wanted to ensure the same behaviour on different machines. The Gaudox exploit kit requires three C&C servers to properly function which results in three separate names being given to each DNS servers.

The repeatability of experiments is a recurring challenge in the field of botnet detection research (Aviv & Haeberlen, 2011). This is caused primarily by privacy concerns related to benign traffic that is mixed in with botnet traffic during experimentation, and it is exacerbated by researchers not specifying which dataset they are using. Hence, selecting a valid dataset that can allow for experimental replication is essential.

2.7 Gaps in the Research

There is a growing body of research in the area of botnets, but there are notable gaps in the literature which should be addressed. Benign botnets have been around for almost as long as the internet has existed, but the innovative ways in which they are being used to coordinate attacks is less well documented (Johnson & Goetz, 2007).

Some recurrent issues within botnet detection research is the use of unbalanced botnet datasets for detection analyses, and uncomprehensive explanations as to how parameters and features were selected (Huancayo Ramos et al., 2020).

One of the most significant gaps in the research surrounding botnet detection is detection of botnets when they are in an early stage. The behaviours that botnets exhibit in the early phase of their-life cycle are generally scanning and simple communication. Research centred on the creation of realistic dataset with an emphasis on early-phase botnet behaviours, rather than botnets that are already engaged in attacks or malicious behaviour, constitutes a major gap in the research. Most importantly, researchers who are involved in botnet detection need to make it clear which dataset and what features of that dataset they are using when determining the effectiveness of botnet detection frameworks.

3. DESIGN/METHODOLOGY

This chapter sets out to give a detailed outline of the software used in the experimentation, how the experiment was designed and set up, why the selected dataset was chosen, and how the data was created and treated.

3.1 Aim of the Experiment

The aim of this experiment was to come up with a peer-to-peer botnet detection framework using the Weka environment. The variables under investigation are accuracy and false-positive rate. The experiments are to be carried out using a dataset which will be outlined in detail in chapter 3.2.

3.2 Botnet Dataset

In order to test the Weka framework, a botnet dataset created by UNSW Canberra containing a mixture of benign and malicious botnet traffic has been selected (Koroniotis et al, 2019). This dataset is relatively recent having been created in 2019, making it a favourable option as the findings of research based on it is likely to stay relevant for a longer period of time. The longevity of the dataset, along with the data already being available in labelled CSV (comma separated values) format as well as in raw PCAP format makes it an ideal choice. It contains several different botnet behaviours, including service scanning, DDoS, and Denial of Service (DoS).

The data is composed of three independent categorical features: category, subcategory, and attack. Category is an attribute containing string data type classes which describe the type of behaviour of a given instance in the dataset. There are five possible classes: DDoS, DoS, Reconnaissance, Normal, and Theft. The Normal class is the only one containing benign traffic in the dataset. There are eight unique string data type classes in the subcategory attribute: UDP, TCP, OS_Fingerprint, Service_Scan, HTTP, Normal, Keylogging, and Data_Exfiltration. Finally, the attack attribute is a binary value with 0 representing normal traffic and 1 representing malicious traffic. The five

Categories (i.e. DDoS, DoS, Reconnaissance, Normal, and Theft) are the target classes in this experiment.

Other botnet datasets were considered before settling on the dataset discussed above. The drawbacks of the datasets used by Ramos Huancayo et al. were mentioned under their approach in the literature review (2020). In brief, the CSE-CIC-IDS2018 contains too few botnet families with only two represented, and the ISOT HTTP botnet dataset contains only botnets without benign forms of traffic, which would probably be better suited to research surrounding classification of botnets families rather than detection of botnets for practical applications.

Newer datasets exist, such as the dataset generated by Garcia et al. containing some real botnet trace files using the honeypot capture method (2020). Moustafa created a recent botnet dataset containing simulated botnet traffic in the form of some common attacks and communication, including (2020). These datasets contain other forms of malware as well as botnets, while not containing benign traffic, which makes them suitable for malware classification research but not an ideal fit for botnet detection research.

The selected dataset was considered to be the most suitable option due to the heterogeneous nature of its contents. It consists of a variation of both benign and malignant traffic which is ideal for assessing the ability of a botnet detection framework to perform adequately under simulated traffic that reflects that found in real situations. However, the dataset was found to be extremely unbalanced, with the Theft class constituting only 1587 rows out of the entire dataset and the Normal class constituting only 9462 rows, both of which are greatly underrepresented, given that the entire dataset consists of over 73 million rows of data. The Reconnaissance class contains 1,821,639 rows, the DoS class contains 32,161,120, the DDoS class contains 37,231,379.

3.3 Experiment Design

3.3.1 The Waikato Environment for Knowledge Analysis (WEKA)

In order for Weka to read datasets, they need to be in a specific file format called attribute-relation file format (arff). There are several ways in which traffic data can be captured. In the instance of the datasets used in this, an application such as WinPcap, libpcap, or Wireshark were used to capture network flows in files called packet capture (PCAP) files. These files first need to be preprocessed and converted into arff files before the Weka environment is able to read or use their contents.

3.4 Setup Overview

Weka version 3.8.5 was installed on a Windows 11 host with 16 GB of RAM and an AMD Ryzen 5 3600 processor. The two datasets used in this research had to be converted from their original PCAP file format to the ARFF format which could be used by Weka.

3.4.1 Feature Selection

Seventeen total features were selected from the UNSW Canberra Bot IoT dataset, and reduced samples of these features were used for the subsequent experiments. The entire feature selection was: the network protocol used in the communication of the bots, the source IP address, the source port number, the destination IP address, the destination port number, the standard deviation of aggregated records, the record total duration, the minimum duration of aggregated records, the average duration of aggregated records, the destination-to-source packets per second, the source-to-destination packets per second, the maximum duration of aggregated records, a binary value corresponding to whether the packet is an attack or normal traffic, the category of attack, and the subcategory of attack. More information regarding these parameters and how they are abbreviated in their default state in the dataset is available in the Appendix A. The features are to be manually selected within

Weka, and each feature selection would then be tested by the four algorithms under evaluation.

3.4.2 Algorithm Selection

Four algorithms have been selected to test their accuracy and false-positive rates on the selected dataset. These are Naïve Bayes, J48, Random Forest, and UltraBoost. A brief explanation of these algorithms will now be given.

Naïve Bayes

Naïve Bayes is a probabilistic classifier algorithm that uses Bayes' Theorem with strong independence assumptions given the class (Webb, 2010). The term "naïve" refers to the predictors being independent of each other. It provides classification accuracy that is comparable to more complex classification algorithms. A major appeal of Naïve Bayes is the low computational overheads required for it to be effective. This feature has made Bayes a widely popular choice among classifier algorithms and it remains competitive with more sophisticated methods that have been developed. THe formula used to perform the classification is given below (Efron, 2013).

$$\mathrm{P(A|B)} = \frac{P(B|A)\cdot P(A)}{P(B)}$$

P(A|B) refers to the posterior probability of a class given a predictor. P(A) refers to the prior probability of a class. Prior refers to the probability of an event before the evidence has been considered, while posterior is the probability after having considered the evidence. P(B) refers to the evidence under evaluation, and B is the set of features that are selected from the dataset. P(B|A) is termed the likelihood which is the probability of the predictor given the class. In the case of this research, A signifies

the Category the botnet falls into, while B refers to the features of the instance that is being categorised.

J48

The J48 algorithm was developed specifically for the Weka environment. Researchers at the University of Waikato designed the algorithm as an implementation of the C4.5 decision tree algorithm. The decision tree generates decisions based on either univariate or multivariate predictors. After the creation of a tree, the algorithm proceeds to go back through each of the leaf nodes and prunes those leaves that are not useful to the algorithm (Salzberg, 1994).

Random Forest

Random Forest is a tree classifier that is constructed through the combination of multiple tree predictors, where each tree is created through randomly created values which are sampled independently (Breiman, 2001). The overall accuracy of the model is ultimately dependent on the constituent trees used in the model and their various correlations. The error rates of the random forest algorithm tend to have better prediction accuracy than other adaptive classifiers such as AdaBoost.

UltraBoost

UltraBoost is a classifier that adaptively boosts like AdaBoost but it differs in that it does not assume that base classifiers are weak. By default it begins with a naïve Bayes classification followed by a logistic regression sequence (*Ultraboost*, 2020). It was developed in an attempt to create a reliable classifier for medical data, with diagnoses using ultrasounds (Moustafa et al., 2020). However, its practical applications may go further than the field of medicine and its usefulness at predicting botnets will be examined in this paper.

3.4.3 How to Preprocess the Data

The capture files from the majority of botnet dataset are generated using Wireshark. These files are by default saved in PCAP format, which are not readily convertible into arff files that are usable by Weka (Fowler & Hammel, 2014). The most difficult complication involved in the conversion stems from PCAP files having a huge number of attributes, with over 271000 fields in 3000 protocols as of version 3.6.1. (*Wireshark · Display Filter Reference: Index*, 2021), which Fowler & Hammel compare to trying to find needles in a haystack (2014). They developed a method of converting PCAP files to arff files with a focus on retaining all of the information in the packet files as the tools that they found omitted some of the information.

The method involves first converting the pcap files to packet details markup language (PDML) format using the tshark tool on Wireshark, and proceeding to use a Python tool that the authors developed which converts the PDML file to arff format. The code for the Python script is publicly available on GitHub, but the file has not been updated since 2014 and was found to be deprecated when the script was attempted due to containing legacy Python 2 code which would require some attention in order to be usable. Moreover, if all of the attributes in the capture files were to be converted to arff format and used in the experiments, the dataset would need to be considerably smaller if Weka was to be capable to perform its computations in a reasonable period of time. So instead, the files should be converted to CSV format using Wireshark's export feature, before being converted to arff format using the Weka ArffViewer Tool.

3.5 Strengths and Limitations of Designed Solution

The strengths and limitations of the designed solution will now be evaluated:

3.5.1 Strengths

The dataset contained in this report is publicly available, recent, and the selected parameters have been explained in detail. These three factors are critical when dealing

with a comparison of botnet detection frameworks to enable replicability of the exact botnet, design, and features used in conducting research in this area. Only through similar feature selection can differences in the capabilities of the frameworks be verified rather than differences in findings resulting from variations in the design of the experiment. The recency of the dataset is also a strength as new botnets are developed regularly with adapting behaviours that need to be taken into account when assessing the proficiency of botnet detection frameworks.

The Weka environment is free and open-source so the impact of these research results can reach a wider use base than commercial solutions that have been developed. The researchers that developed Weka update the documentation regularly so those who are interested in attempting to

3.5.2 Limitations

The datasets used may be somewhat limited in that they do not contain a comprehensive list of all possible botnets and therefore cannot give a conclusive idea as to whether the botnet detection framework given here is accurate against all possible botnets that exist.

Weka is well suited for small to medium sets of data, but for very large datasets the time taken to process can become a constraint due to Weka being relatively memory intensive. Botnet datasets used in research can get quite large which can cause processing time to be quite inhibitory – up to a day in some cases. For the purposes of research and feature selection when designing models, time constraints are not as important, but if results are needed promptly a big data processor would be more suited.

There is a possible lack of longevity for the relevance of the findings of this research paper. The botnet landscape is constantly changing and new strains of botnet families are developed in an irregular and unpredictable manner. As a consequence of this, the results produced here for the botnets present in the datasets used in this study may

become less relevant as new botnets with different forms of concealment and communication are developed, and the older families that are studied here get superseded by more modern threats.

As with all network-flow analysis techniques, this approach requires training data for attacks that are not yet known. A greater number of features used in the feature selection can likewise cause unnecessary overheads which limits the computers that it can be used on and the speed with which the computations can take place (Than Vu et al., 2021).

The research conducted in this paper was limited to research and conference papers that were published in English. Open-access research papers were used where possible, but some book chapters were used to clarify certain concepts in the literature review.

3.6 Assumptions

There are several assumptions that have been made in the design of this experiment:

- Firstly, it must be remarked that the dataset under investigation was not created by the author of this paper, and as such it is assumed that the data contained within it is accurate, balanced, and complete. It is assumed that the training and test sets contained in the selected dataset were independently sampled and that the data in each set do not overlap in any meaningful way. A significant overlap in data between the training and testing sets would lead to unreliable results.

- It is assumed that the botnet detection framework selected are operating adequately on the desktop computer that is evaluating their performance.

- It is also assumed that the results that were found while conducting the literature review provide honest findings from the detection frameworks that were developed. Due to time constraints, it would not be possible to test each detection framework and dataset that has been mentioned in this research paper.

4. IMPLEMENTATION/EXPERIMENT

The experiment consists of several distinct steps: balancing the dataset through undersampling, configuring the Weka configuration file to increase the heap size, installing the desired classification algorithms into Weka, converting the data into arff format that is compatible with Weka, inputting the data file to Weka for preprocessing, and selecting and running the classification algorithms.

4.1 Balancing the dataset

There was a high imbalance in the dataset, so it was decided to undersample to overrepresented classes. The first step in rectifying the imbalance was to extract the classes into their own CSV files using Code Snippet 1 in Appendix B. This created five CSV files of varying sizes. The most underrepresented class was found to be the Theft class with just 1587 samples. The remaining classes were then reduced to the size of the most underrepresented class using Code Snippet 2 in Appendix B and all of the Theft sample was simply appended to this file as no reduction was required for it. This resulted in a CSV file which contained an equal number of samples from all five classes. The rows contained in this new file needed to be randomised so that the training and testing data would contain an approximately equal distribution of classes for their classification. The randomisation was completed using Code Sample 3 in Appendix B.

4.2 Increasing the Heap Size

The PC used in this experiment had 16GB of RAM, but the default heap size allowed in 4 GB which is considerably lower than the host machine was capable of tolerating. The original attempts to run the algorithms resulted in the application crashing with a log file generated that contained information about the error that had occurred, which turned out to have been caused by the heap size exceeding its limit. The size of the heap allowed in the Java Virtual Machine can be increased by entering the Weka root directory, which is located in Program Files by default on Windows or wherever the installation took place on Linux devices. The RunWeka.ini configuration settings file

was then opened as an administrator with a text editor, and the section with the following was edited so that the upper bound of the heap size was increased to 8GB.

```
# the heap size (or any other JVM option)
#javaOpts=%JAVA_OPTS%
javaOpts=%JAVA_OPTS% -Xmx8192m
```

The file was then saved before exiting.

4.3 Installing Packages to Weka

Weka comes with some algorithm packages during initial installation, but there are many packages available to download that contain classification algorithms that might be preferred for the research being undertaken. In the case of this thesis, it was necessary to ensure that J48, Random Forest, Naïve Bayes, and UltraBoost had been downloaded onto the host machine. Algorithms can be downloaded by opening the Package Manager, which is located in the Tools tab in the GUI. The available algorithms with their associated categories are listed here, and once navigating to the required package, they can be downloaded and installed by pressing Install. It should be noted that no other Weka application windows (Explorer, Experimenter, Knowledge Flow, Simple CLI,) should be open during their installation, and a notification box appears instructing the user to close any other Weka application windows before proceeding.

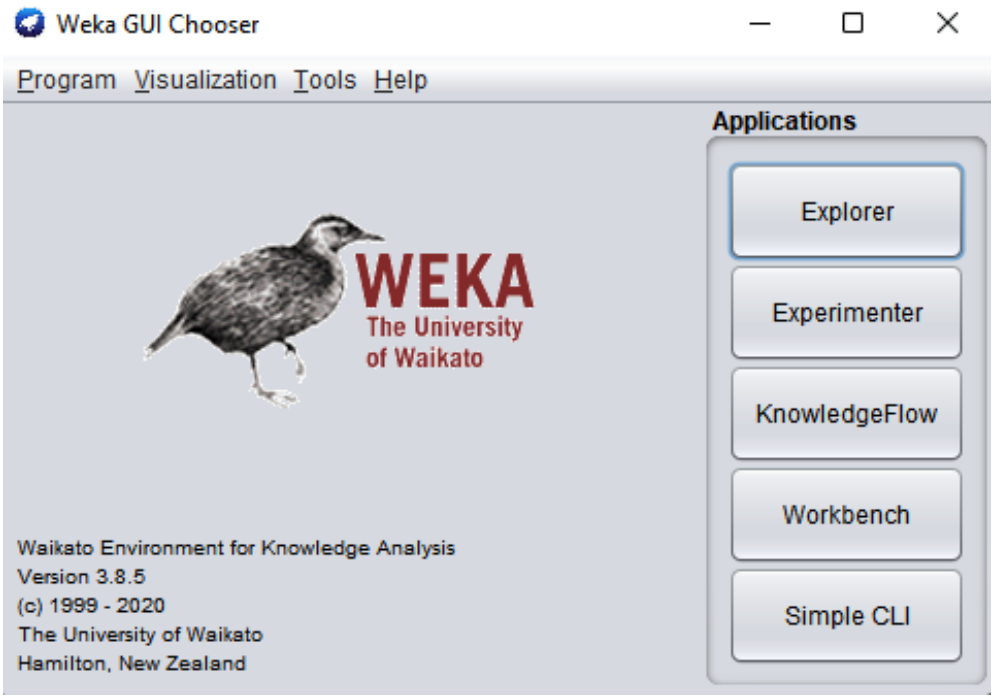

Figure 3: The Weka GUI on start-up

4.4 Preprocessing the Data

The "sport", "dport", and "max" attribute data entries needed to be cleaned. In order to be used by the algorithms in this research, string values need to be removed. The instances that contained non-numeric port values were removed from the dataset through a simple Python script which is found at Code Snippet 4 in Appendix B. Before using the script, it should be ensured that the column that needs to be cleaned should be put into the relevant index that is written in the script. This can be done by simply dragging the column in a spreadsheet such as Excel. This clearing step needs to be performed on both the training and testing datasets.

The dataset then needed to be converted to arff format before being uploaded to Weka. This conversion can be done via the Weka ArffViewer in the tools tab. The structure of the Arff file is shown below in a text editor which displays the various attributes and their associated type. When Weka starts up, a graphical user interface appears with five options to choose from in the Applications section: Explorer, Experimenter,

KnowledgeFlow, Workbench and Simple CLI. To access the algorithms necessary in this research, Explorer was selected.

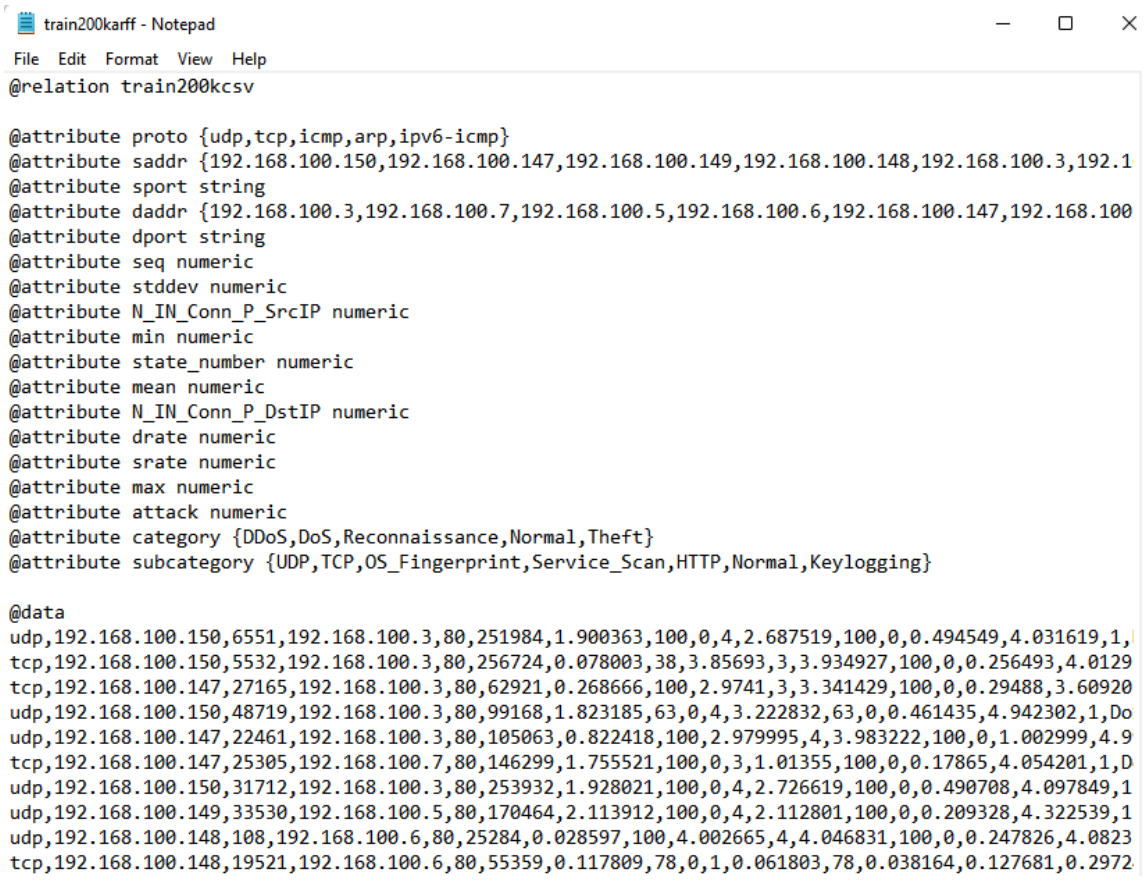

Figure 4. Structure of the Converted ARFF file in a text editor

In the Explorer application, the initial step was to input the training data. By clicking on the Open File button and navigating to the training dataset on the host's directory, the training set is inputted to Weka and the default attributes along with their counts and weights are visible in the GUI.

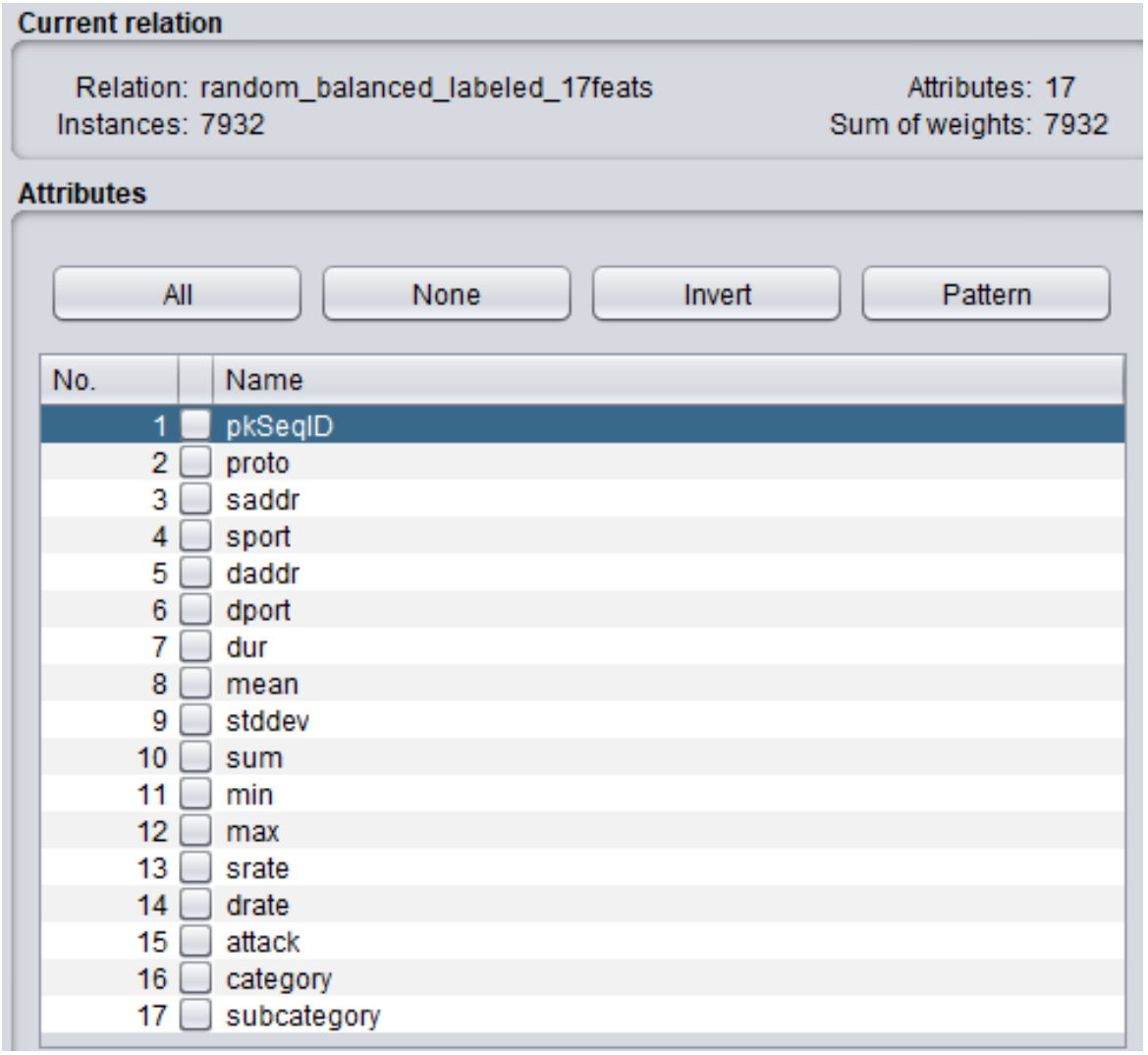

Figure 5: Attributes from training set displayed in Weka

There are 17 attributes and a total of 7932 instances contained in the overall dataset. A table displaying each attribute with its description can be found in the Appendix. At this point the relevant attributes can be selected, depending on which features are desired to be included for each experiment. For the first experiment, the subcategory, attack, and pkSeqID columns were omitted as pkSeqID is simply a row number and attack and subcategory are not realistic data that would be available to program that is evaluating a network. For each experiment that is run, the features can be selected in this section of Weka.

4.5 Running the Classifiers

After inputting the training data into Weka, the Classify tab at the top of the GUI was entered in order to select and run the algorithms to test their classification accuracy, precision, false-positive rate, and execution times. Under the Classifier section the Choose button was selected which opens a directory of the available algorithms that can be used on the data. The desired algorithm was selected and under Test Options the Supplied Test Set radio button was selected. It is also possible to input only the one dataset and divide it into test and training sets by the preferred ratio in the Percentage Split input box. The classifier was then started by pressing the Start button. Weka proceeds to build a model based on the training data that has been inputted and then subsequently tests the testing data according to the newly built model. This step was carried out four times, with a different classifier selected on each occasion.

4.6 Results Breakdown

When Weka finishes processing a classification algorithm, it generates a report that ends with a summary providing details of how well the algorithm performed. The algorithms that were tested had different levels of success. Four metrics were selected in order to evaluate the results of the approach used in the experiment: accuracy, false positive rate (FPR), precision, and execution time. An image of the generated report for the random forest algorithm is below

```
=== Summary ===

Correctly Classified Instances        2566               95.1428 %
Incorrectly Classified Instances       131                4.8572 %
Kappa statistic                          0.9393
Mean absolute error                      0.0244
Root mean squared error                  0.125
Relative absolute error                  7.6379 %
Root relative squared error             31.2537 %
Total Number of Instances             2697

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.969    0.012    0.951      0.969    0.960      0.951    0.987     0.969     Reconnaissance
                 0.984    0.004    0.984      0.984    0.984      0.979    0.996     0.978     Theft
                 0.909    0.021    0.916      0.909    0.912      0.891    0.961     0.907     DDoS
                 0.982    0.004    0.986      0.982    0.984      0.980    0.995     0.984     Normal
                 0.911    0.020    0.918      0.911    0.914      0.894    0.975     0.899     DoS
Weighted Avg.    0.951    0.012    0.951      0.951    0.951      0.939    0.983     0.948

=== Confusion Matrix ===

   a   b   c   d   e   <-- classified as
 508   8   7   0   1 |   a = Reconnaissance
   9 540   0   0   0 |   b = Theft
   8   0 489   5  36 |   c = DDoS
   1   1   2 549   6 |   d = Normal
   8   0  36   3 480 |   e = DoS
```

Figure 6: Weka's generated report for the J48 algorithm.

4.6.1 Accuracy

Accuracy is here defined as the percentage of results correctly judged. The formula used to calculate the accuracy requires the True Positive (TP), True Negative (TN), False Positive (FP), and True Negative (TN) values (Zhu et al., 2010). It can be calculated with the equation:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

In the case of this study, TP refers to the number of PCAP files in which botnets are present and were correctly identified as being present. FP refers to the number of PCAP files in which botnets are absent but were incorrectly identified as being present. TN refers to the number of PCAP files in which botnets are absent and were correctly

identified as being absent. FN refers to the number of PCAP files in which botnets are present but were incorrectly identified as being absent.

When all of the 17 features were included, the random forest algorithm attained the highest accuracy with a correct prediction result of 96.70%. The UltraBoost algorithm obtained the lowest accuracy having only correctly predicted 23.99% of the instances in the sample. The accuracy results for all of the tested algorithms are tabulated below:

Algorithm	**Accuracy**
Naive Bayes	44.16%
UltraBoost	23.99%
J48	95.14%
Random Forest	96.70%

Table 2: The tested algorithms with their respective accuracies

4.6.2 False Positive Rate

In statistical analysis, the FPR is a measure of accuracy for a test that is defined as the probability of erroneously rejecting the null hypothesis when it is true (Wang & Zheng, 2013). It can be formally denoted as:

$$false\ positive\ rate\ (\alpha) = \{reject\ H_0 \mid H_0\ true\}$$

The formula used to calculate the FPR requires the FP and TN values. It can be calculated with the equation:

$$FPR = \frac{FP}{FP + TN}$$

The lowest false positive rate was attained by random forest with 0.008, while the highest false-positive rate was obtained by UltraBoost with 0.135. All of the algorithms with their associated false-positive rates are tabulated below:

Algorithm	**False-Positive Rate**
Naive Bayes	0.075
UltraBoost	0.135
J48	0.012
Random Forest	0.008

Table 3: The tested algorithms with their respective false-positive rates

4.6.3 Precision

Precision is a measure of statistical variability. The formula used to calculate the Precision requires the TP and FP values. The equation below can be used to calculate its value (Ting, 2010).

$$Precision = \frac{TP}{TP + FP}$$

The highest precision was attained by the random forest algorithm with 0.967, while UltraBoost resulted in the lowest precision with 0.356. Each of the tested algorithms with their associated scores is tabulated below.

Algorithm	Precision
Naive Bayes	0.544
UltraBoost	0.356
J48	0.951
Random Forest	0.967

Table 4: The tested algorithms with their respective precisions

4.6.4 Execution Time

There was a large variation in the execution times of the four algorithms. The algorithm that took the longest duration of time to execute was random forest with 1.04 seconds. The algorithm that took the shortest duration of time to execute was naïve Bayes with 0.01 seconds. All four of the algorithms with their associated execution time are tabulated below.

Algorithm	Execution Time (seconds)
Naive Bayes	0.01
UltraBoost	0.77
J48	0.75
Random Forest	1.03

Table 5: The tested algorithms with their respective execution times

4.6.4 Results with Four Selected Features

When only four features are used in the classification – source and destination ports, and source and destination IP addresses – the accuracy and precision results were lower, and the false-positive rate was higher. The accuracy, precision, false-positive rate, and execution time results are tabulated below.

Algorithm	**Accuracy**	**False Positive Rate**	**Precision**	**Execution Time (seconds)**
Naive Bayes	43.38%	0.137	0.533	0.01
J48	92.77%	0.018	0.927	0.03
UltraBoost	20.43%	0.168	0.456	0.19
Random Forest	95.00%	0.012	0.951	0.81

Table 6: The tested algorithms and their associated scores when four features are selected

4.6.5 Results with Two Selected Features

The experiments were recreated a third time with an even further reduced feature set, this time with only information recording the rate of packet transfer from source to destination and the rate of packet transfer from destination to source. The accuracy, precision, false-positive rate, and execution times for the two selected features is tabulated below.

Algorithm	**Accuracy**	**False Positive Rate**	**Precision**	**Execution Time (seconds)**
Naive Bayes	24.10%	0.185	0.542	<0.01
J48	61.33%	0.084	0740	0.02
UltraBoost	24.88%	0.183	0.501	0.04
Random Forest	60.07%	0.097	0.713	0.85

Table 7: Results of algorithms when two features are selected

4.7 Findings Discussion

The results were readily available in the summary table given by Weka after a classification algorithm had been executed. Given all of the features in the dataset, random forest was the most accurate and also the most precise and had the lowest false-positive rate, and naïve Bayes was the fastest. There was a large variation between the longest and shortest execution times when all features were used, with a difference of a multiple of over 100 between the fastest and slowest algorithms.

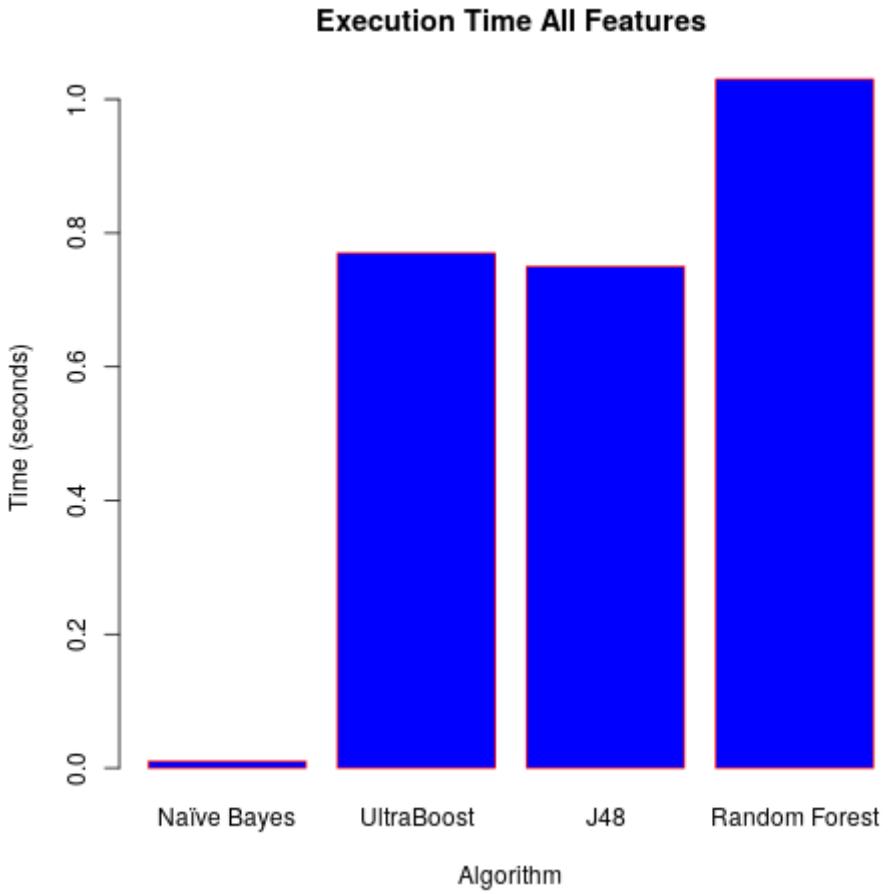

Figure 7: Bar chart showing the execution times with all features

This gap became significantly reduced when only source and destination ports and source and destination IP addresses were included in the list of features, but the execution times overall were shorter for all three algorithms except for Naïve Bayes, which had the same execution time when four features were used. The gap between the execution times of the four algorithms widened considerably:

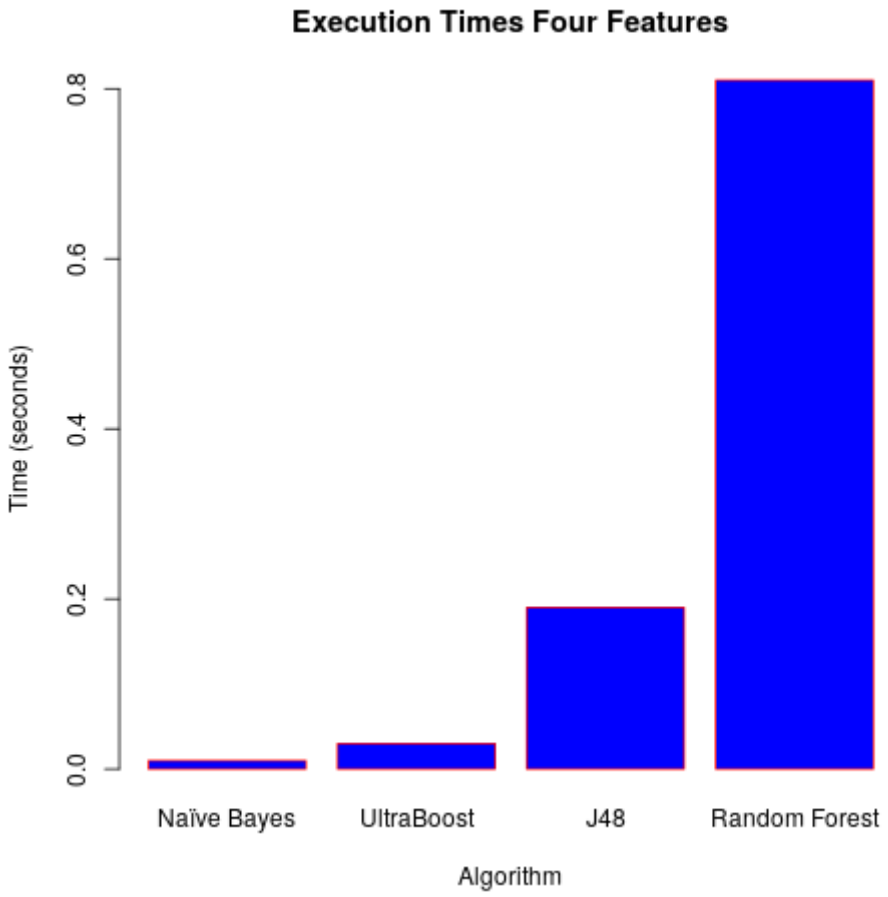

Figure 8: Bar chart showing the execution times with four features

When only two features are included in the selection – Source to destination packets per second and destination to source packets per seconds – the resulting accuracies changed significantly. J48 performed the best out of the algorithms attaining a resulting accuracy of 61.33%. Naïve Bayes predicted the lowest accuracy with just 24.10% and UltraBoost performed only marginally better with its classification accuracy with 24.88%. Random Forest performed slightly worse than J48 with a classification accuracy of 60.07%. The lowest false positive rate was attained by J48 with 0.084.

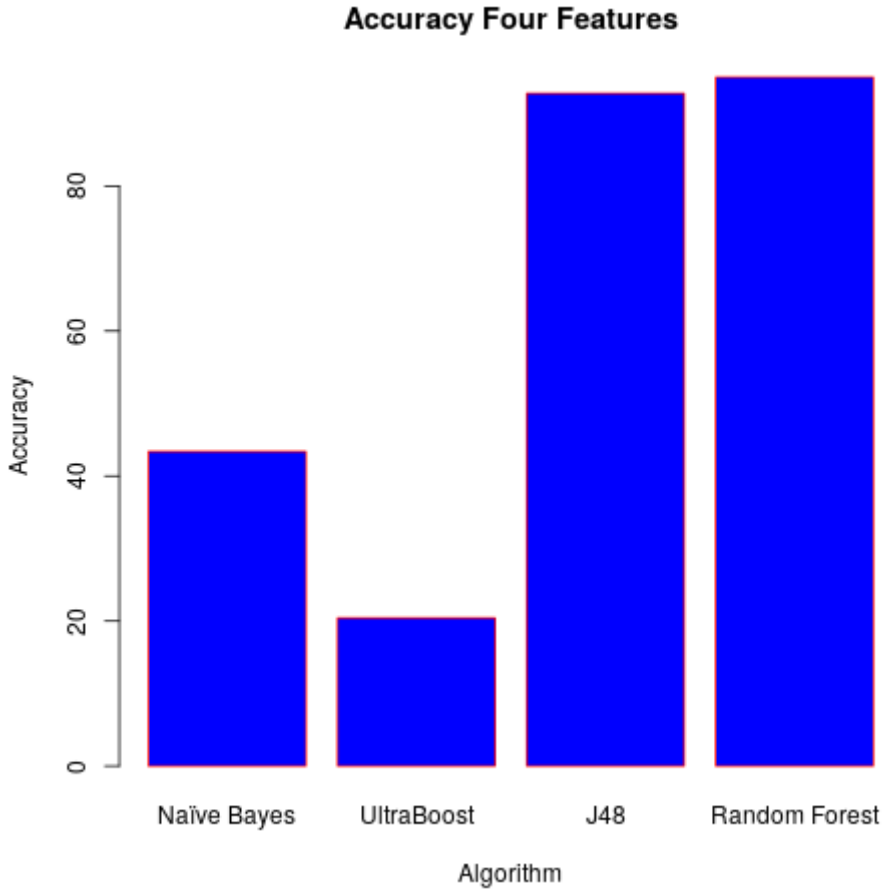

Figure 9: Bar chart showing the accuracies with four features

4.8 Strengths and Weaknesses

Strengths

Weka automatically generates a detailed report of the algorithms that it created, which can be immediately interpreted to determine whether the data and algorithm used have been accurately classified. The displayed results can be readily compared to the results of other algorithms used within Weka or by different statistical methods used in other research papers.

The abstraction of the statistical methods used by Weka results in fewer limitations as to who can use the tools provided by it. With little preparation, one can quickly and effectively carry out a number of statistical tests which can provide an idea of which algorithms are best suited to the selected datasets.

The dataset was sufficiently heterogeneous, and the imbalances in the dataset were overcome successfully through undersampling techniques to create a dataset where all of the classes are equally represented.

Weaknesses

Weka is limited by the size of the datasets that can be inputted into it. In the case of the present study, the testing and training subsets of the dataset needed to be truncated considerably so that Weka could perform the necessary calculations. However, on performing the tests several times with different truncated samples of the original data, the results were found to have a high level of precision which indicates that smaller datasets are sufficient in providing consistent and reliable results for the selected dataset.

The dataset that was selected did not contain a realistic spread of benign and malicious traffic. However, it was deemed to be the most suitable dataset available due to its recency and the diversity of significant attacks contained in it. None of the publicly available datasets were considered to have all of the desired qualities for botnet

detection research, so it was decided that the most important factors (recency and variety of attacks) would be selected for. However, if the dataset had been engineered using SMOTE technology to correct its imbalance rather than undersampling, the results may have been more favourable for the Naïve Bayes and UltraBoost algorithms.

The final balanced dataset was also rather small which resulted in execution times that may have been less precise than a dataset with larger quantities of data. This was due to the fact that the least represented class contained approximately 1600 features and the approach that was used required random samples of the other classes to be taken to create an equally distributed set of classes. This led to a much smaller dataset than had been anticipated, though some of the algorithms still performed well with the minimal training data available to them.

The decision to include UltraBoost in the algorithm selection is difficult to justify after seeing the results, given its poor performance across all feature selections. However, it was originally included due to it performing better than the other three algorithms when the original 5% dataset which was suggested by the authors of the dataset was used. Unfortunately, its promising performance was not retained after the undersampling technique had been applied to the dataset.

The decision to use the more traditional machine learning algorithms that are available in Weka could be considered weak, given that the vast majority of recent contributions to this field come from neural networks, and deep learning in particular. However, it was deemed possible that traditional machine learning may have use cases in certain environments where deep learning techniques would not be feasible (such as microprocessors that have been installed on edge networks). For this reason, it was determined to still be an acceptable approach in this field of research.

5. EVALUATION/ANALYSIS

5.1 Research Overview

The null hypothesis (H0) given in the introduction, namely that "*Weka cannot be used to successfully detect Internet of Things botnets using four of its inbuilt classification algorithms, namely J48, Naïve Bayes, UltraBoost, and Binary Forest, with 95% prediction accuracy*" can now be fully evaluated as to its veracity. As the Random Forest and J48s algorithm resulted in attaining above the 95% accuracy prediction threshold, we can reject the null hypothesis.

The first objective (O1) was *to show that Weka can successfully detect botnets from a recent dataset containing simulated Internet of Things botnet traffic.* This research objective was achieved, as Weka was successful in reliably detecting botnets within the designated 95% target that had been set when using the J48 and Random Forest algorithms. However, not all of the algorithms were equally successful so the results of these experiments have varying contributions to this field of research.

The second objective (O2) was to compare the differing predictive accuracies, false-positive rates, precisions, and executions times of the algorithms under investigation. This research objective was achieved as it was determined that there were dissimilarities in the selected metrics. The performances of the selected algorithms were visualised in the form of bar charts and tables to highlight the differences between them.

The third objective (O3) was to compare the results of the algorithms when different sets of features were available to them. This was achieved by firstly including all of the features in the dataset (which resulted in the best performances by all algorithms), and then reduced sets of features. The first set of reduced features contained information solely regarding source and destination IP addresses and source and destination ports. The findings showed that Random Forest provided the highest rate of accuracy with

this reduced feature availability and was the only algorithm to attain an accuracy of over 95%. With the reduced features, the time taken to both build and test the models was significantly longer for all algorithms. When the features were altered and reduced further to just the attributes of “srate” and “drate”, J48 provided the greatest level of accuracy with 61.33% and Naïve Bayes scored the lowest with 24.10%.

5.3 Discussion

The results were in line with expectations. The highest accuracy was attained with the Random Forest algorithm and the lowest accuracy resulted from UltraBoost. The algorithms provided a varied rate of success at classifying the network flow files for the features that were selected for in the dataset.

While the research objectives were achieved, albeit in an environment that might be difficult to make interoperable with a truly viable IDS environment, an important takeaway from this research paper is that the classification of botnets according to the type of traffic that they are exhibiting is feasible using free and open-source software that can be used with relatively few barriers of entry. This is a favourable outcome as it allows for a greater variety of researchers with various skill sets and backgrounds to contribute to the growing body of research in this field.

An interesting outcome was that when two selected features were used – source to destination packets per second and destination to port packets per second – J48 was the most accurate with its predictions attaining a score of 61.33%, and also took only 0.02 seconds to execute. Naïve Bayes became the least accurate predictor for the experiment with two features. This could be an important factor at points on the network where only low amounts of data are available for classification. An example of this could be a node on the network that is not directly receiving packets for inspection so it would be difficult to determine the protocols, IP addresses, and ports involved, but the rate at which the packets are moving would be available for assessment. With different sets of limited features being available in different

environments, it is worth researching which approaches are more accurate depending on the attributes that are available when classifying incoming data.

In the set of experiments when only information regarding the source and destination IP addresses and ports were used, random forest once again became the most accurate when classifying the botnets with an accuracy score of 95%.

5.3.1 Comparison to Previous Research

The results produced by the research in this paper will now be compared to the results found in other recent papers, taking into account studies that either used the same dataset as the one used in the present study, or that used similar algorithms in their predictions.

Bot-IoT Dataset

The algorithms tested during this research were less accurate than some of the successes that have been reached with more sophisticated methods such as the results attained by Ibitoye et al. on the same dataset as used in this paper – the Bot-IoT dataset (2019). They investigated the use of spike neural networks (SNN) and feedforward neural networks (FNN), and the FNN achieved 95.1% accuracy and 95% precision, though no mention was provided as to the execution times of the classification. The SNN performed worse than the FNN with only 91% accuracy and 92% precision. The results from the random forest algorithm and J48 in this study outperformed the SNN method in accuracy and precision. UltraBoost and Naïve Bayes were both less accurate and less precise. It is important to note that a higher number of features was used in the algorithms in this study than in the research conducted by Ibitoye et al., with more features generally producing more accurate results in classification algorithms.

Apostol et al. used a deep neural network (DNN) approach on the same dataset, achieving much higher accuracy and precision results with 99.7% and 0.99 respectively (2021). These scores were significantly higher than any those attained by the algorithms that were used in this research paper, again without providing the execution times required for their computation. Nevertheless, these scores were the

highest found in the literature for the chosen dataset, and in research conditions are evidently more accurate at classifying botnets. An NVIDIA Geforce graphics card and Core i7 CPU were used in the experiment, which are relatively high range processors which can perform the necessary computations for deep learning quickly. However, there is a trade-off between the use of high grade components and versatility of the approach. The more sophisticated components that are required for deep learning might not be as feasible in nodes on edge or fog networks (Santos & Almeida, 2020), so it is uncertain as to the practical implementation of approaches involving deep learning towards the edge.

Pokhrel et al. analysed several machine learning algorithms on the dataset, including K-Nearest Neighbour (KNN), naïive Bayes, and multi-layer perceptron neural network (MLP ANN) to test their performances (2021). They selected seven features in the dataset, namely pkts, rate, state, drate, dur, spkts, and dpkts. Before SMOTE was applied to the dataset, naïve Bayes achieved 99.4% accuracy on real-time data but after SMOTE this was reduced to 51.5%. The KNN and MLP ANN accuracy results also fell by lesser degrees. The split that was used in their experiments for a training to testing ratio was 5:1 which differs from the 2:1 that was selected in the experiments in this paper. This highlights the differences in outcomes that can occur with approaches that use SMOTE and those that do not. Naïve Bayes appeared to perform significantly worse with the set of features that were selected in this report.

Weka

Weka's classification in other datasets is similar to the results found in the dataset in this study. Susanto et al. came to different results regarding the accuracy and execution times when using random forest and naïve Bayes (2020). Their experiments, carried out on the N_BaIoT dataset, resulted in random forest classifying botnets with 99.99% accuracy while naïve Bayes predicted with 82.35% accuracy. This is in contrast to the results produced in the present paper, where the random forest and naïve Bayes accuracy results had a very different result – 96.7% and 44.16% respectively.

Jabar & Mohammed tested a large selection of Weka's algorithms on a 2017 dataset – CICIDS2017 – including performing an accuracy and precision analysis (2020). Two of the algorithms that they tested were also tested in this paper, random forest and naïve Bayes. For their dataset, random forest performed notably better at classification accuracy and precision, scoring 99.99% and 0.998 respectively when classifying with 10 features. Naïve Bayes also scored higher than the results found in this paper, with results of 81.32% accuracy and just 0.039 precision when classifying for 10 features. The differences in outcomes between Jabar & Mohammed's research and the research carried out here may be a result of differential feature selection, or indeed the data itself from the dataset that was selected for this research being less capable of being correctly classified. One reason why Naïve Bayes may have performed significantly worse in this research than Jabar & Mohammed's experiments may be as a result of the zero-frequency problem, whereby a categorical variable from test set didn't show up in the training set which could lead to less precision and accuracy in the classification of the data.

Hao et al. achieved between 91.9 and 95.3% precision when using three typical machine learning algorithms in Weka on the PeerRush dataset, namely random forest, decision trees, and Bayesian network (2021). While the random forest results were similar to those found in this research, the most significant finding from their study was 100% accuracy when the minimum support was increased to 0.0015. This was the case due to the Zeus network flow packets being very low compared to the other two botnets in the dataset so the optimal minimum support that was discovered was found to reduce the Zeus botnet related IP rules below the level. However, the PeerRush dataset from 2014 that was used in that study contained fewer real features that would be found in actual network traffic, such as data associated with ports and protocols used. As such, it is unclear whether their results would have been as successful with a newer dataset with a more complete range of network features.

5.4 Implications of Research

The research showed that Weka was capable of correctly classifying botnets with the use of the four algorithms under investigation. UltraBoost was the least accurate, with a correct prediction accuracy rate of 23.99% when all features were used but also with a relatively high execution time of 0.77 seconds, with the likely implication that this algorithm may not be ideally suited to the purposes of botnet classification research, or at least not for the dataset that was used in the present research. J48 was relatively fast when four features were selected for. A possible inference of this finding is that J48 could be well suited for classification that must be carried out in a short period of time. J48 could be implemented in an IDS where prompt identification of malicious data is a priority, in speeds that could approach real-time given the conversion of data to a suitable format and classification of the dataset are stream-lined. Conversely, for highly secure networks such as bank systems or government communication, highly accurate identification with very few false-negatives is the most crucial feature, and in these situations approaches such as the highly successful neural networks designed by Apostol et al. would be better suited (2021).

Weka's usefulness as an actual IDS in a practical sense is somewhat limited, due to the nature of having to input data into Weka, format it into the suitable ARFF files that Weka can read, preprocess the data and perform all of these necessary actions through a GUI which needs to be manually employed by the user. However, it could be possible to automate the conversion of incoming Wireshark PCAP files into ARFF format and to develop a script that would immediately preprocess and execute a set of preselected algorithms on a periodic basis. An API already exists for Weka, and a text to ARFF conversion script has already been developed in Java (*Using the Api - Weka Wiki*, 2021). This API offers an extensibility that could lead to several practical use cases when using Weka, including the development of a practical IDS by using the algorithms that Weka facilitates.

If Weka was to be used in the development of a true IDS, its merit at detecting DDoS, DoS, theft, and reconnaissance, and normal network behaviours has been supported by the findings of the experiments that were carried out. However, given that the selected dataset had significantly more attacks than passive communication behaviours, further research would need to be undertaken to assess its ability at the full range of behaviours that are typical of a botnet that is still in its early phase. Preventing bots from scanning and spreading in the first place is an important step in the overall health of societal networks and a prophylactic measure that might be more useful than simply detecting malicious behaviour.

Thanh Vu et al. performed a meta-analysis on botnet-detection techniques, and it was found that the majority of detection strategies proposed in the literature used neural networks and machine learning approaches (2021). Techniques such as the one used in this study with network flow analysis make up a sizable proportion of the ML network traffic research that has been conducted. It is important to differentiate between the results that are found with neural network approaches and those that are found with less computationally costly approaches such as Weka, as the two approaches will have different use cases when put into practice depending on what resources are feasible for their implementation.

UltraBoost performed poorly on all of the experiments. This algorithm has so far mostly been used in medical diagnostics which is what it was designed to do. It is based on a combination of naïve Bayes with a logistic regression sequence and this model proved to have little merit with the findings of this report as it underperformed compared to the other selected algorithms.

5.4 Overall Assessment of Research

Overall, the results were acceptably comparable to previous research. The precision and accuracy attained by random forest appeared to be the most effective, while the speed with which naïve Bayes could classify was the lowest, leading to a trade-off between accuracy/precision and execution time. Depending on the size of the dataset and the overall prediction accuracy required, J48 or Random Forest could be used to varying degrees of success and speed.

6. CONCLUSION AND FUTURE WORK

The final chapter will evaluate the findings of this research in light of the objectives and research question that were outlined in the introduction. Recommendations for future research will be provided along with details of the contribution and impact of the research that was undertaken.

6.1 Research Overview

The state of the art methods and approaches involved in the field of botnet detection were practically evaluated in the literature review chapter of this research paper. It was found that machine learning and artificial intelligence approaches to botnet detection are popular, and increasing in popularity. There are growing efforts to combine machine learning techniques with other areas of research, such as software defined networking, blockchain technologies, deep learning, and explainable artificial intelligence. Weka was also found to provide high levels of accuracy in its classification.

6.2 Experimentation, Evaluation, and Limitations

A conclusive discussion about the experimentation, the evaluation of the results, and an exploration of the limitations inherent to the methodology are summarised below.

6.2.1 Experimentation

The experimentation had varying levels of success. UltraBoost performed consistently poorly with accuracy scores between 20.43% and 24.88% depending on the number of features that were selected. Random forest performed consistently well, with a lowest accuracy of 60.07% attained when only two features were used, and 96.70% when all features were used. Naïve Bayes had results that were expectedly similar to UltraBoost, with its lowest accuracy attained being 24.10% with two features and its highest accuracy being 44.16% when all features were available. J48 performed the best when only two features were used with a score of 61.33%. Its highest accuracy was attained when all features were used with a score of 95.14%.

The execution times were dissimilar, with Naïve Bayes being considerably faster than the other algorithm when all features were used with an execution time of 0.01 seconds, while random forest was the algorithm with the longest execution time with 1.03 seconds.

6.2.2 Evaluation

The selected dataset contained a wide array of botnet behaviours, and it appears to be the most comprehensive botnet dataset created to date with its authors having combined many different types of botnet behaviour into one big set of CSV files. However, the proportion of normal traffic to malicious traffic was skewed with a lot more malicious traffic than would exist under realistic conditions. A dataset with a more balanced distribution of benign and malicious traffic would be a better test of the effectiveness of botnet detection frameworks. A more realistically designed dataset would contain a skew in the opposite direction, with most of the traffic being benign and only some of it containing traces of botnet behaviour. Given this constraint, a much smaller subset of the data was taken to correct the imbalance which may have led to lower amounts of training data than would be ideal for some of these algorithms (notably UltraBoost and Naïve Bayes).

Nevertheless, Weka was demonstrated to be a competent tool in detecting botnet behaviours such as DDoS, DoS, and Scanning, and the random forest algorithm was found to be the most accurate with its predictions. The null hypothesis (H0) was rejected and the research objectives (O1 and O2) were fulfilled to varying degrees.

When compared to previous Weka research, it was found that random forest was generally among the best performing algorithms and naïve Bayes was among the worst which was the same result as the experiments conducted for this paper. When results were compared to studies using the same dataset – Bot-IoT dataset from UNSW Canberra – the accuracy and precision was found to be significantly higher when using deep learning techniques such as DNN and slightly higher when using techniques like

FNN than when using the algorithms in this paper. However, the practical implementation of FNN and DNN in certain environments such as the edge was considered to be more limited than the use of less computationally-intensive algorithms such as those made available through Weka. If work could be carried out to test the two approaches on the same dataset in the future, it would be more feasible to make a direct comparison between the neural network approach and Weka and the findings would help to clarify the differences in classification accuracy, precision, false-positive rate, and execution time between the two approaches.

6.2.3 Limitations

The limitations of the experiment were outlined in chapter 4, but in light of the results they should be explored further. The application used was able to accurately predict botnets from a relatively new dataset, but it was limited as to the size of the dataset that was available for classification. A subset of the dataset created by the researchers at UNSW was available which was less than 5% of the original size of the original dataset, but this 5% subset was found to be highly unbalanced and therefore did not give reliable results. It was necessary to extract the different classes of data from the entire dataset in order to create a balanced dataset This constituted a considerably smaller proportion of the entire dataset – accuracy precision, false-positive rate, and execution time – but if a bigger dataset was required in practice then the algorithms may have attained different results. Instead, big data technology that is designed for analysing large data sets would likely be a more suitable choice.

Furthermore, Weka was only able to accurately detect botnets that had already been captured, labelled, and suitably preprocessed. This limits the usefulness of the results of this approach in its practical applications, as real-world traffic flows constantly in huge quantities, and unless a suitable application programming interface (API) was designed so as to use Weka in real-time, its practicality as an intrusion defence system to be used by actual people is severely limited. The design and development of such an API would likely be a complex task due to the vast and variegated amounts of data that would need to be prepared and converted to a suitable ARFF format before processing. While the results were positive in a lab setting where there was sufficient time to

adequately prepare and select the data, the suitability of this approach on its own appears to be best suited to researching algorithms in an experimental way. Algorithms that have first been demonstrated to be effective at classifying botnets in an experimental setting can then be adapted for use in systems that are capable of classifying real time network flows.

6.3 Contribution and Impact

This dissertation focused on the detection of botnets using Weka. Weka is a tool that can be installed onto any device running Windows or Linux and can be put to work readily with a wide selection of machine learning and data mining algorithms with users requiring only limited knowledge of the machine learning tools and algorithms that are being used. As was the objective of the research undertaken in this paper, further evidence has been provided to support the effectiveness and relevance of Weka within the field of botnet detection research. These findings can be interpreted positively, in that machine learning practitioners and laypeople alike have an additional tool that can be used in the ongoing competition between standard network users and malicious botmasters.

A significant benefit of choosing Weka for botnet detection is that it can be used on virtually any modern personal computer, with the latest stable release Weka 3.9 only requiring Java 8 or later to function correctly (*Requirements - Weka Wiki*, n.d.). This opens up the possibility of using Weka 3.9 to detect the presence of malware on a variety of systems. For example, microprocessors equipped with an API that can perform the algorithms available to Weka could exist in the places where typical host machines used in research would not be feasibly installed, such as edge and fog networks. However, the development of such an API would not be non-trivial for the reasons outlined in the previous section.

A further contribution is the analysis of feature selection for four different algorithms, with three different feature selections having been analysed which represented

different amounts of available data and also different categories of data in network flow packets. It was found that the algorithms had differing levels of success depending on the amounts and types of data that were available to them.

Finally, the usefulness of the UltraBoost algorithm in botnet detection has been put into question by the findings of this report. UltraBoost performed the worst at accurately classifying botnets under 2 of the 3 feature selections. Given the dearth of scientific literature on this algorithm, it was not yet possible to compare the results of UltraBoost to the findings of research of other datasets so its disappointing performance when certain features are selected may leave interested researchers to decide not to investigate it further for purposes other than medical diagnostics. The reason why this algorithm was included in the first place is that it was found to be effective in its classifications when the original dataset that was created by the authors was used for classification. However, when the dataset was altered to make it balanced its accuracy fell considerably.

This study adds to the growing body of knowledge in botnet detection research. It gives further support to the feasibility of using a free and open source application with very few overheads and serves as a benchmark that can be used in future comparisons and analyses between the various approaches to botnet detection. On a more general note, this research provides evidence of the effectiveness of Weka in classification research in general. The impact of this is significant as researchers who are not well versed in how to construct effective classification algorithms from first principles have the option to use Weka in the analysis of their data without a need for prerequisite knowledge in advanced programming or machine learning. Researchers who study machine learning tools or classification algorithms in general may also benefit from the knowledge that an additional contribution has been made by Weka.

6.4 Future Work and Recommendations

The present study focused on botnet detection frameworks that target botnets that are in the later stages of their life-cycle. By this point, the botnet is already propagating and attacking other nodes, by which point substantial damage has already occurred. It is important to focus on early-stage botnet behaviours and patterns of communication such as scanning so that botnets can be detected before they have infected enough machines to cause serious problems. A focus on prophylactic detection would help to reduce the spread of botnets before widespread dispersal has the chance to occur. In particular, further experimentation is needed to evaluate the possibility of detecting botnet communication. There are other novel avenues of investigation that require attention, such as the implementation of botnet detection frameworks in edge and fog networks, where the maximum compute and memory possibilities may be lower than those available in lab settings. To this end, experiments designed for measuring the effectiveness of botnet detection frameworks with a strong focus on computational overheads would be worthy of exploration, particularly with various delimiting categories for each type of device that is feasible in different portions of the network.

It is equally important for the datasets used in future botnet research to remain relevant and up to date with the constantly changing trends in families of common botnets. To achieve this end, further research efforts should be put into network virtualization with the aim to model both of the primary methods of botnet communication (HTTP, IRC) to provide variegated datasets which would better test the versatility of botnet detection frameworks. Research focused on the creation of well labelled and realistic network traffic flows in various file formats would likewise make research in this domain more streamlined.

Specifically, botnets that are communicating in the early stages of their life cycle by scanning the network environment should be targeted for research, as botnets that are still in the scanning phase are capable of causing fewer problems than botnets that have already spread. Botnet detection in the propagation phase also constitutes a major

gap in the research which needs to be addressed, and the creation of more datasets with general botnet scanning communication patterns would help to prevent day-zero botnet attacks where their signatures would not yet be present in the databases that are available to some botnet detection frameworks.

A final recommendation for future research is to model feature selection in botnet classification research around environments where limited data attributes are available for use by the detection framework. Examples of this would be nodes on the network that are simply transmitting data but may not necessarily have access to the contents of the data, or nodes that can determine where the data is coming from or heading towards but have no other information to do with the packets.

The landscape of the internet is continuously changing. The success of botnets relies on their concealment within legitimate and often essential systems of communication. As networks continue to expand, with an attack surface that is growing with the proliferation of devices that are connected to the internet, the importance of developing methods to detect and prevent the spread of botnets is difficult to overstate. Society is moving steadily into an era where the ubiquity of IoT devices will provide both opportunities and threats in equal measure. As botnets continue to diversify and become more complex through techniques such as encrypted communication and dynamically generated IP addresses, researchers in this field should continue their efforts in devising new strategies and tools to detect their presence.

7. BIBLIOGRAPHY

Abu Khurma, R., Almomani, I., & Aljarah, I. (2021). Iot botnet detection using salp swarm and ant lion hybrid optimization model. *Symmetry*, *13*(8), 1377. https://doi.org/10.3390/sym13081377

Alauthaman, M., Aslam, N., Zhang, L., Alasem, R., & Hossain, M. A. (2018). A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks. *Neural Computing and Applications*, *29*(11), 991–1004. https://doi.org/10.1007/s00521-016-2564-5

Al-Duwairi, B., & Al-Ebbini, L. (2010). Botdigger: A fuzzy inference system for botnet detection. *2010 Fifth International Conference on Internet Monitoring and Protection*, 16–21. https://doi.org/10.1109/ICIMP.2010.11

Alenazi, A., Traore, I., Ganame, K., & Woungang, I. (2017). Holistic model for HTTP botnet detection based on DNS traffic analysis. In I. Traore, I. Woungang, & A. Awad (Eds.), *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments* (pp. 1–18). Springer International Publishing. https://doi.org/10.1007/978-3-319-69155-8_1

Alothman, B., & Rattadilok, P. (2017). Towards using transfer learning for botnet detection. *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 281–282. https://doi.org/10.23919/ICITST.2017.8356400

Alparslan, E., Karahoca, A., & Karahoca, D. (2012). Botnet detection: Enhancing analysis by using data mining techniques. *IntechOpen*. https://doi.org/10.5772/48804

Apostol, I., Preda, M., Nila, C., & Bica, I. (2021). IoT Botnet Anomaly Detection Using Unsupervised Deep Learning. *Electronics*, *10*(16), 1876. https://doi.org/10.3390/electronics10161876

Aviv, A., & Haeberlen, A. (2011) Challenges in experimenting with botnet detection systems, *4th USENIX Workshop on Cyber Security Experimentation and Test (CSET'11)*, 1–8. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1646&context=cis_papers

Barford, P., & Yegneswaran, V. (2007). An inside look at botnets. In M. Christodorescu, S. Jha, D. Maughan, D. Song, & C. Wang (Eds.), *Malware Detection* (pp. 171–191). Springer US. https://doi.org/10.1007/978-0-387-44599-1_8

Bhatt, P., & Thakker, B. (2021). A novel forecastive anomaly based botnet revelation framework for competing concerns in internet of things. Journal of Applied Security Research, 16(2), 258–278. https://doi.org/10.1080/19361610.2020.1745594

Biglar Beigi, E., Hadian Jazi, H., Stakhanova, N., & Ghorbani, A. A. (2014). Towards effective feature selection in machine learning-based botnet detection approaches. *2014 IEEE Conference on Communications and Network Security*, 247–255. https://doi.org/10.1109/CNS.2014.6997492

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Campanile, L., Gribaudo, M., Iacono, M., Marulli, F., & Mastroianni, M. (2020). Computer network Simulation with ns-3: A Systematic Literature Review. *Electronics*, *9*(2), 2-4. https://doi.org/10.3390/electronics9020272

Chaeikar, S., Zeidanloo, H., Manaf, A., Ahmad, R., & zamani, mazdak. (2010). A proposed framework for P2P botnet detection. *International Journal of Engineering and Technology*, 2, 161–168. https://www.researchgate.net/publication/310793596_A_Proposed_Framework_for_P2P_Botnet_Detection

Chakrabarti, S., Chakraborty, M., & Mukhopadhyay, I. (2010). Study of snort-based IDS. *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, 43–47. https://doi.org/10.1145/1741906.1741914

Chen, R., Niu, W., Zhang, X., Zhuo, Z., & Lv, F. (2017). An effective conversation-based botnet detection method. *Mathematical Problems in Engineering*, *2017*, 1–9. https://doi.org/10.1155/2017/4934082

CSE-CIC-IDS, Canadian Institute for Cybersecurity (2018). Retrieved 29 December 2021, from https://www.unb.ca/cic/datasets/ids-2018.html

Efron, B. (2013). Bayes' theorem in the 21st century. *Science*. https://doi.org/10.1126/science.1236536

Enisa threat landscape 2020—Botnet (2020.). [Report]. ENISA. Retrieved 31 December 2021, p. 13. from https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-botnet

Eustis, A. (2019). The Mirai Botnet and the Importance of IoT Device Security. *16Th International Conference On Information Technology-New Generations (ITNG 2019)*, 85-89. https://doi.org/10.1007/978-3-030-14070-0_13

Fowler, C. A., & Hammel, R. J. (2014). Converting PCAPs into Weka mineable data. *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 1–6. https://doi.org/10.1109/SNPD.2014.6888681

Garcia, S., Parmisano, A., & Erquiaga, M. J. (2020). *IoT-23: A labeled dataset with malicious and benign IoT network traffic*. Zenodo. https://doi.org/10.5281/zenodo.4743746

Göbel, J., & Holz, T. (2007). Rishi: Identify bot contaminated hosts by irc nickname evaluation. *HotBots*. Retrieved 31 December 2021 from https://www.usenix.org/legacy/event/hotbots07/tech/full_papers/goebel/goebel.pdf

Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B., Dagon, D.: Peer-to-peer botnets: overview and case study. In: Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets, p. 1. USENIX Association (2007).

Gu, G., Gu, G., Yegneswaran, V., Porras, P., Stoll, J., & Lee, W. (2009). Active botnet probing to identify obscure command and control channels. *2009 Annual Computer Security Applications Conference*, 241–253. https://doi.org/10.1109/ACSAC.2009.30

Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *17th USENIX Security Symposium* (pp. 139-154). Atlanta, Georgia; College of Computing, Georgia Institute of Technology. Retrieved 29 April 2021, from https://www.usenix.org/legacy/events/sec08/tech/full_papers/gu/gu_html/index.html

Hao, S., Liu, D., Baldi, S., & Yu, W. (2021). Unsupervised detection of botnet activities using frequent pattern tree mining. *Complex & Intelligent Systems*. https://doi.org/10.1007/s40747-021-00281-5

Huancayo Ramos, K., Sotelo Monge, M., & Maestre Vidal, J. (2020). Benchmark-based reference model for evaluating botnet detection tools driven by traffic-flow analytics. *Sensors*, *20*(16), 3-25. https://doi.org/10.3390/s20164501

Holz, T., Holz, S., Moritz & Dahl, Frederic & Biersack, Ernst & Freiling, Felix. (2008). Measurements and mitigation of peer-to-peer-based botnets: a case study on Storm worm. Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats. https://www.usenix.org/legacy/event/leet08/tech/full_papers/holz/holz.pdf

Ibitoye, O., Shafiq, M. O., & Matrawy, A. (2019). Analyzing adversarial attacks against deep learning for intrusion detection in iot networks. *2019 IEEE Global Communications Conference (GLOBECOM)*. https://doi.org/10.1109/GLOBECOM38437.2019.9014337

Idrissi, I., Boukabous, M., Azizi, M., Moussaoui, O., & El Fadili, H. (2021). Toward a deep learning-based intrusion detection system for IoT against botnet attacks. *IAES International Journal of Artificial Intelligence (IJ-AI)*, *10*(1), 110. https://doi.org/10.11591/ijai.v10.i1.pp110-120

Imperva. (2021). *Bad Bot Report 2021* (pp. 3-29). Retrieved from https://www.imperva.com/resources/resource-library/reports/bad-bot-report/

Jabar, A., & Mohammed, I. (2020). Development of an optimized botnet detection framework based on filters of features and machine learning classifiers using CICIDS2017 dataset. *2nd International Scientific Conference of Al-Ayen University (ISCAU-2020)* Baghdad; IOP Publishing. http://doi:10.1088/1757-899X/928/3/032027.

James R. Binkley and Suresh Singh. 2006. An algorithm for anomaly-based botnet detection. In Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2 (SRUTI'06). USENIX Association, USA, 7.

Johnson, M., & Goetz, E. (2007). Embedding information security into the organization. *IEEE Security & Privacy Magazine*, *5*(3), 16-24. https://doi.org/10.1109/msp.2007.59

Khattak, S., Ramay, N., Khan, K., Syed, A., & Khayam, S. (2014). A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys & Tutorials*, *16*(2), 898-924. https://doi.org/10.1109/surv.2013.091213.00134

Khodamoradi, P., Fazlali, M., Mardukhi, F., & Nosrati, M. (2015). Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms. *2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS)*, 1–6. https://doi.org/10.1109/CADS.2015.7377792

Koroniotis, N., Moustafa, N., Sitnikova, E., & Slay, J. (2018). Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques. In J. Hu, I. Khalil, Z. Tari, & S. Wen (Eds.), *Mobile Networks and Management* (pp. 30–44). Springer International Publishing. https://doi.org/10.1007/978-3-319-90775-8_3

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, *100*, 779–796. https://doi.org/10.1016/j.future.2019.05.041

Kumar, K., Kumar, N., & Shah, R. (2020). Role of IoT to avoid spreading of COVID-19. *International Journal Of Intelligent Networks*, *1*, 32-35. https://doi.org/10.1016/j.ijin.2020.05.002

Liu, L., Chen, S., Yan, G., & Zhang, Z. (2008). BotTracer: Execution-based bot-like malware detection. *Lecture Notes In Computer Science*, 97-113. https://doi.org/10.1007/978-3-540-85886-7_7

McDermott, C. D., Majdani, F., & Petrovski, A. V. (2018). Botnet detection in the internet of things using deep learning approaches. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN.2018.8489489

Miller, S., & Busby-Earle, C. (2016). The impact of different botnet flow feature subsets on prediction accuracy using supervised and unsupervised learning methods. *Journal Of Internet Technology And Secured Transaction*, *5*(2), 474-485. https://doi.org/10.20533/jitst.2046.3723.2016.0060

Moustafa, A. F., Cary, T. W., Sultan, L. R., Schultz, S. M., Conant, E. F., Venkatesh, S. S., & Sehgal, C. M. (2020). Color doppler ultrasound improves machine learning diagnosis of breast cancer. Diagnostics, 10(9), 631. https://doi.org/10.3390/diagnostics10090631

ns-3. (n.d.). Retrieved 23 November 2021, from https://www.nsnam.org/about/.

Ogu, E. C., Ojesanmi, O. A., Awodele, O., & Kuyoro, 'Shade. (2019). A botnets circumspection: The current threat landscape, and what we know so far. *Information*, *10*(11), 337. https://doi.org/10.3390/info10110337

Pokhrel, S., Abbas, R., & Aryal, B. (2021). Iot security: Botnet detection in iot using machine learning. *ArXiv:2104.02231 [Cs]*. http://arxiv.org/abs/2104.02231

Popoola, S., Adebisi, B., Ande, R., Hammoudeh, M., Anoh, K., & Atayero, A. (2021). SMOTE-DRNN: A Deep Learning Algorithm for Botnet Detection in the Internet-of-Things Networks. *Sensors*, *21*(9), 2985. https://doi.org/10.3390/s21092985

Ramsbrock, D., & Wang, X. (2013). Chapter 12—The Botnet Problem. In J. R. Vacca (Ed.), *Computer and Information Security Handbook (Second Edition)* (pp. 223–238). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-394397-2.00012-X

Requirements—Weka wiki. (n.d.). Retrieved 29 December 2021, from https://waikato.github.io/weka-wiki/requirements/

Santos, S. F. dos, & Almeida, J. (2020). Deep learning towards edge computing: Neural networks straight from compressed data. *ArXiv:2012.14426 [Cs]*. http://arxiv.org/abs/2012.14426

Salzberg, S. L. (1994). C4. 5: Programs for machine learning by J. Ross Quinlan. *Machine Learning*, *16*(3), 235–240. Morgan Kaufmann Publishers, Inc. https://doi.org/10.1007/BF00993309

Shinan, K., Alsubhi, K., Alzahrani, A., & Ashraf, M. (2021). Machine Learning-Based Botnet Detection in Software-Defined Network: A Systematic Review. *Symmetry*, *13*(5), 11. https://doi.org/10.3390/sym13050866

Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., & Kemmerer, R. et al. (2009). Your botnet is my botnet. *Proceedings Of The 16Th ACM Conference On Computer And Communications Security - CCS '09*. 635-647 https://doi.org/10.1145/1653662.1653738

Sudhakar, & Kumar, S. (2019). Botnet detection techniques and research challenges. *2019 International Conference On Recent Advances In Energy-Efficient Computing And Communication (ICRAECC)*. https://doi.org/10.1109/icraecc43874.2019.8995028

Susanto, Stiawan, D., Arifin, M. A. S., Idris, Mohd. Y., & Budiarto, R. (2020). Iot botnet malware classification using weka tool and scikit-learn machine learning. *2020 7th International Conference on Electrical Engineering, Computer Sciences and Informatics (EECSI)*, 15–20. https://doi.org/10.23919/EECSI50503.2020.9251304

Thanh Vu, S., Stege, M., El-Habr, P., Bang, J., & Dragoni, N. (2021). A Survey on Botnets: Incentives, Evolution, Detection and Current Trends. *Future Internet*, *13*(8), 9. https://doi.org/10.3390/fi13080198

The CTU-13 dataset (2011). A labeled dataset with botnet, normal and background traffic. *Stratosphere IPS*. Retrieved 29 December 2021, from https://www.stratosphereips.org/datasets-ctu13

Ting, K. M. (2010). Precision. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 780–780). Springer US. https://doi.org/10.1007/978-0-387-30164-8_651

Trend Micro (2006). Taxonomy of botnet threats [White Paper]. 11. https://sites.cs.ucsb.edu/~kemm/courses/cs595G/TM06.pdf

Tyagi, A. K., & Aghila, G. (2011). A wide scale survey on botnet. *International Journal of Computer Applications*, *34*(9), 10-11.

Tzagkarakis, C., Petroulakis, N., & Ioannidis, S. (2019). Botnet attack detection at the iot edge based on sparse representation. *2019 Global IoT Summit (GIoTS)*, 1–6. https://doi.org/10.1109/GIOTS.2019.8766388

UltraBoost. GitHub. (2020). Retrieved 27 December 2021, from https://github.com/tedcary/UltraBoost.

Using the api—Weka wiki. (2021). Retrieved 31 December 2021, from https://waikato.github.io/weka-wiki/using_the_api/

Vignau, B., Khoury, R., Hallé, S., & Hamou-Lladj, A. (2021). The botnet simulator: A simulation tool for understanding the interaction between botnets. *Software Impacts*, *10*, 1-2. https://doi.org/10.1016/j.simpa.2021.100173

Visconti, A., & Tahayori, H. (2011). Artificial immune system based on interval type-2 fuzzy set paradigm. *Applied Soft Computing*, *11*(6), 4055-4063. https://doi.org/10.1016/j.asoc.2010.12.011

Wang, H., & Zheng, H. (2013). False positive rate. In W. Dubitzky, O. Wolkenhauer, K.-H. Cho, & H. Yokota (Eds.), *Encyclopedia of Systems Biology* (pp. 732–732). Springer. https://doi.org/10.1007/978-1-4419-9863-7_224

Wazzan, M., Algazzawi, D., Bamasaq, O., Albeshri, A., & Cheng, L. (2021). Internet of things botnet detection approaches: analysis and recommendations for future research. *Applied Sciences*, *11*(12), 27. https://doi.org/10.3390/app11125713

Webb, G. I. (2010). Naïve Bayes. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 713–714). Springer US. https://doi.org/10.1007/978-0-387-30164-8_576

Winter, P., Lampesberger, H., Zeilinger, M., & Hermann, E. (2011). On detecting abrupt changes in network entropy time series. *Communications And Multimedia Security*, 194-205. https://doi.org/10.1007/978-3-642-24712-5_18

Wireshark · display filter reference: Index. (2021). Retrieved 23 December 2021, from https://www.wireshark.org/docs/dfref/

Wuchner, T., Cislak, A., Ochoa, M., & Pretschner, A. (2019). Leveraging compression-based graph mining for behavior-based malware detection. *IEEE Transactions On Dependable And Secure Computing*, *16*(1), 99-112. https://doi.org/10.1109/tdsc.2017.2675881

Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C., & Kirda, E. (2009). Automatically generating models for botnet detection. *Computer Security – ESORICS 2009*, 232-249. https://doi.org/10.1007/978-3-642-04444-1_15

Xing, Y., Shu, H., Zhao, H., Li, D., & Guo, L. (2021). Survey on botnet detection techniques: classification, methods, and evaluation. *Mathematical Problems In Engineering*, *2021*, 1-24. https://doi.org/10.1155/2021/6640499

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., & Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, *39*, 2-16. https://doi.org/10.1016/j.cose.2013.04.007

Zhu, W., Zeng, N. and Wang, N. (2010) Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations.
NESUG Proceedings: Health Care and Life Sciences, Baltimore, Maryland, 1-9.
https://www.lexjansen.com/nesug/nesug10/hl/hl07.pdf

8. APPENDIX A

Abbreviation	Description
pkSeqID	Row Identifier
proto	Protocol used for communication
saddr	Source IP address
sport	Source port number
daddr	Destination IP address
dport	Destination port number
stddev	Standard deviation of aggregated records
sum	Total duration of aggregated records
min	Minimum duration of aggregated records
mean	Average duration of aggregated records
dur	Record total duration
drate	Destination-to-source packets per second
srate	Source-to-destination packets per second
max	Maximum duration of aggregated records
attack	Whether it is an attack (binary value)
category	Category of attack
subcategory	Subcategory of attack

Table 3: Dataset attributes with their descriptions

9. APPENDIX B

```
import csv

file_list = []
for i in range(1, 75):
   file_list.append("UNSW_2018_IoT_Botnet_Dataset_" + str(i) + ".csv")

for i in file_list:
   with open(i) as inp:
       lines = csv.reader(inp)
       with open('normal_data.csv', 'a') as out:
           for row in lines:
               if row[-2] == "Normal":
                   out.write(",".join(row) + "\n")
```

Code Snippet 1: Python code that extracts the normal data from all 74 of the source files and writes them to a CSV file. This can be amended and repeated for all five classes by changing the string value of the targeted second last row where the class value is located.

```
import pandas
import random
import os

n = 1821639 # total rows
s = 1588 # required rows
filename = "reconnaissance_data.csv"
skip = sorted(random.sample(range(n), n-s))
df = pandas.read_csv(filename, skiprows=skip)
print(df)
output_path='undersampled_data.csv'
df.to_csv(output,mode='a',header=not os.path.exists(output))
```

Code Snippet 2: Python code that reduces the size of a CSV file using the Pandas library. It randomly takes 1588 rows (the same number as the least represented sample)

and writes them to a new CSV file. This could be repeated for all four of the overrepresented classes.

```
import pandas as pd

df = pd.read_csv('unrandom_balanced.csv', header=None)
ds = df.sample(frac=1)
ds.to_csv('random_balanced.csv')
```

Code Snippet 3: Python code that takes the balanced CSV file and uses the Pandas library to randomise the rows contained in it, writing the output to a new CSV file.

```
with open('random_balanced_labeled_17feats.csv', 'r') as inp:
    rows = inp.readlines()
    with open('cleaned_all_feats.csv', 'w') as out:
        out.write(rows[0])
        for row in rows[1:]:
            li = list(row.split(",")) # turn the row into a list
            if li[3].isnumeric():
                out.write(row)
```

Code Snippet 4: Python code that cleans the first column. The code writes the first line (which is the header containing the attribute names) to a new file and then proceeds to write every line where the fourth column (in the case of this dataset the “dport” column) is numeric. Its purpose is to remove rows where port values contain letters or other non-numeric data. This can be performed on other columns whose values need to be numeric.