

Systematic Analysis on Peer-to-Peer Botnet Attack Detection

by

Faiza Binte istiaq

18301227

Rubaiyat E Mohammad

18301103

Moriom Tasnia

18301058

Kazi Moinul Hassan

18301290

Tanjim Tabassum

20101629

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
September 2022

© 2022. Brac University
All rights reserved.

Declaration

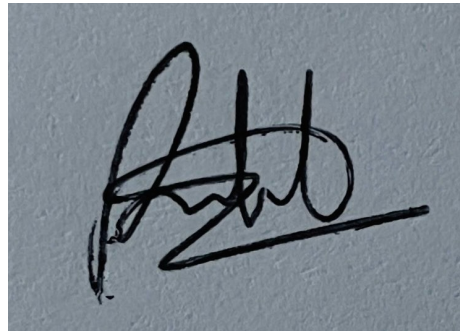
It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I/We have acknowledged all main sources of help.

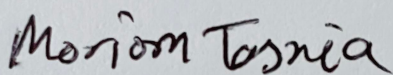
Student's Full Name & Signature:



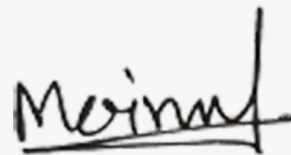
Faiza Binte Istiaq
18301227



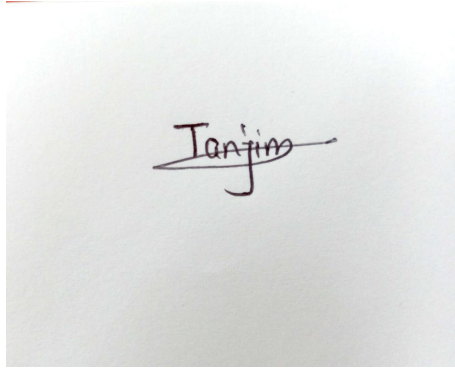
Rubaiyat E Mohammad
18301103



Moriom Tasnia
18301058



Kazi Moinul Hassan
18301290



Tanjim Tabassum
20101629

Approval

The thesis/project titled “Systematic Analysis on Peer-to-Peer Botnet Attack Detection ” submitted by

1. Faiza Binte istiaq(18301227)
2. Rubaiyat E Mohammad(18301103)
3. Moriom Tasnia(18301058)
4. Kazi Moinul Hassan(18301290)
5. Tanjim Tabassum(20101629)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 16, 2022.

Examining Committee:



Supervisor:
(Member)

Dr. Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

”Botnet” refers to a network of compromised machines that the bot master remotely controls to prosecute innumerable malicious activities through a CC server and miscellaneous slave machines. It is possible to categorize botnets as centralized (CC) or decentralized (P2P). According to their distributed functionality, recently P2P botnets is the most significant risks to network security . In this paper, we systematically analyze and compare some very recent peer-to-peer botnet algorithms and methods such as Honeypots, AutoBotCatcher, SDN, and PeerGrep to ascertain the most appropriate one for real-world applications. To perform this comparison, we examine AutuBotCatcher, an algorithm that utilizes the community detection method, Honeypot system, where we focus on the Nepethesis honeypot method. Additionally, the PeerGrep system integrates the PeerGrep algorithm, CART algorithm, and P2P traffic in SDN to automate and flexibly manage flow entries through machine learning.

Keywords: Botnet; Peer to Peer; Honeypots; AutoBotCatcher; SDN; PeerGrep

Dedication

To our honorable supervisor for his support and devotion.

Acknowledgement

First and foremost, I am grateful to Allah for allowing this thesis to be completed without significant interruptions.

As a final note, we want to express our deepest gratitude to our supervisor, Dr. Amitabha Chakrabarty, as his guideline and expertise were crucial to the success of this thesis.. His insightful feedback, advice, and assistance prompted us to refine our conceptualization to push our work to a more advanced level.

Finally, our families' support and prayers are deeply appreciated. Throughout this research work, they were constant supporters and supportive, and this would not have been possible without them.

Table of Contents

Declaration	i
Approval	iii
Abstract	iv
Dedication	v
Acknowledgment	vi
Table of Contents	vii
List of Figures	ix
List of Tables	x
Nomenclature	xi
1 Introduction	1
1.1 Research Objective	3
1.2 Research Challenges	3
1.3 Motivation	4
2 Literature Review	6
2.1 Peer to Peer Botnet	6
2.2 Functioning of peer to peer botnet	7
2.3 History	7
2.4 Current Scenario	9
2.5 Detection and prevention of peer to peer botnet	10
2.5.1 Honeypot and network traffic analysis	10
2.5.2 PeerGrep	11
2.5.3 Software defined network	11
2.5.4 AutoBotCatcher and mutual contact graph	12
2.5.5 Community detection	13
2.5.6 Decision tree and multilayer neural network	13
2.5.7 Focused View of Detection methods	14
3 Related work	15

4	P2P Botnet Detection Systematic	18
4.1	AutoBotCatcher	19
4.1.1	Network Data Transaction	19
4.1.2	System Architecture	21
4.1.3	Identifying Botnet	22
4.2	PeerGrep	22
4.2.1	System Design	23
4.2.2	Evaluation	25
4.3	Honeypot	27
4.3.1	Proposed Method	27
4.3.2	Analytic Results	28
4.4	Software Defined Network	30
4.4.1	SDN Based Detection	30
4.4.2	Network Traffic Analysis	33
4.4.3	Evaluation	34
4.5	Resilient Neural Network	35
4.5.1	Proposed Approach	36
4.5.2	Performance evaluation	37
5	Analysis and Result	40
5.1	Analysis	40
5.1.1	Survey comparison:	40
5.2	The Comparison Approach	41
5.3	Result	44
6	Conclusion and Future Work	45
6.1	Conclusion	45
6.2	Future Work	45
	Bibliography	50

List of Figures

1.1	Centralized and decentralized architecture	1
2.1	Peer to Peer botnet operation	10
2.2	Summary of detection methods	14
4.1	System Flow	21
4.2	System Design	23
4.3	P FR of P2P host detection for different value of θ_{bgp}	26
4.4	TPR and FPR of θ_{act}	26
4.5	PR of θ_S and θ_{CV}	27
4.6	ATSRW	29
4.7	Number of Compromised Peer	29
4.8	Bypass STARW Delay.	30
4.9	Y Values after Minimization	30
4.10	System overview	32
4.11	Flow diagram.	32
4.12	Sequence diagram of bot detection.	34
4.13	Block diagram of the proposed system	36
4.14	FPR, precision rate and F-measure comparison	38
5.1	Categories of comparing detection techniques	41

List of Tables

2.1	Timeline of Peer-to-Peer Protocols and Bots	9
4.1	Network traffic evaluation and detection accuracy.	35
4.2	Real world evaluation result.	35
4.3	Features selection algorithm computation time	38
4.4	Neural network training time with CART subset, ReliefF subset and PCA	38
5.1	Comparison with other approaches	43

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

BFT Byzantine Fault Tolerant

C&C Command and control

CART Classification and Regression Tree

CNN Convolutional Neural Network

DDos Distributed Denial-of-Service

DNS Domain Name System

DPI Deep Packet Inspection

FPR False Positive Rate

HTTP Hypertext Transfer Protocol

IRC Internet Relay Chat

P2P Peer to Peer

RNN Recurrent Neural Network

SDN Software Defined Network

TPR True Positive Rate

Chapter 1

Introduction

In recent years, Internet virus attacks have grown increasingly organized and profit-driven. Due to this trend, botnets play a crucial role in causing click fraud, unsolicited mail, and DDoS attacks. In computer science, a cluster of maliciously controlled computers (or "bots") owned by malicious software is called a botnet. Generally, malware is installed through viruses, worms, and Trojan horses. An attacker (the botmaster) remotely controlled the zombie computers. The combined bandwidth and computational power of massive botnets of computers are immense. DDoS attacks, keylogging, cracking of passwords, and email spam are some harmful activities that botmasters engage in. A significant drawback of the Internet today is botnets [6].

In a globalized world, centralized botnets are still frequently handed down. The centralized botnet connects its bots to several computers (called "CC") to receive orders. The command and control servers (CC) in this configuration have a severe flaw, despite being easy to build and capable of meeting a botmaster's demands. The elementary control communication architecture of a typical command and control server (CC) and decentralized botnet is depicted in Fig. 1.1 (Many CC servers are frequently involved in this). All of the bots in a botnet would stop communicating with their botmaster if those servers were shut off. Furthermore, by having a fake join a specific CC channel, defenses can simply keep an eye on the botnet.

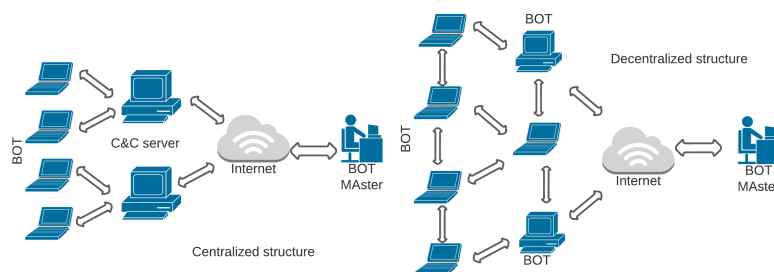


Figure 1.1: Centralized and decentralized architecture

Attackers have begun to progressively comprehend the drawbacks of Trojans, Storm botnets [16], or other commonly centralized botnets over the last few years. Pea-comm botnets [59] and Waledac botnets [58], a recently upgraded variant, have grown concerning peer-to-peer networks, which are resistant to dynamic stir. When many bots are lost, P2P botnet communication will not be disrupted. P2P botnets do not require a centralized server; bots act as both command and control servers and clients. P2P botnets are superior to conventional centralized botnets. They are more resilient and challenging for the security sector to protect than the subsequent generation of botnets. Security experts have investigated ways to recognize, track, and protect against botnet-based assaults as they grow in popularity and hazard. Most recent studies have been on CC botnets that have previously emerged, particularly those that are IRC-based. Such research must be done in order to combat the threat we currently face. The backbone of modern botnet architecture is the CC servers, according to a botmaster. Defenders will destroy the few CC servers, initially resulting in the botmaster's botnet losing control. Second, backers can quickly identify each command and control server (CC) by its IP address, which contains the list of CC servers, contingent on whether or not a few bots have been taken hostage. Thirdly, if backers seize ownership of or seize control of a botnet's CC, the entire botnet could be publicized. New botnets with a new control architecture will be developed and distributed by hackers. Network security experts will devote more time and resources to preventing botnet attacks. As a result, security experts must create new techniques to lessen the threat caused by peer-to-peer botnets. Presently, the only advice given by security professionals for reducing the spread of malware, including botnet infections, is to practice "basic computer hygiene" [48]. There are many ways to recognize, track, and protect against botnet-based assaults as they grow in popularity and hazard. Most recent studies have focused on command and control botnets that have emerged in the previous era, notably ones that consume IRC [47]. However, it can be excessively restrictive, and other mitigating strategies might not work. More research is required to develop affordable and efficient anti-malware solutions to combat botnets' proliferation and the current threat we face. Consequently, the purpose of this research is to contrast and compare the traits of various P2P botnet categories, in addition to their differences and similarities, while looking at their vulnerabilities and potential defenses. We explored P2P botnets to shed more light on them and assist security researchers and experts in creating efficient defenses against them.

This paper presents various models of peer-to-peer botnets and systems for detecting P2P botnets with a high detection rate and the ability to recognize unknown P2P signatures. The final components are arranged as follows. In Section 2, a framework for identifying P2P botnets is proposed. Section 3 demonstrates a correlation strategy. Section 4 offers judgments and recommendations for the future. The paper ends with a review of potential applications of the peer-to-peer botnet model and defenses against them.

1.1 Research Objective

The purpose of this research is to analyze and contrast the features of the various P2P botnet varieties, and examine their commonalities and dissimilarities, and to take a glance at their vulnerabilities and potential mitigation techniques. We expect to educate the masses about P2P botnets and guide security firms to develop strengthened defense mechanisms against them.

Our contributions are as follows:

- To find out different strategies for the protection of individuals and corporations.
- To set a multi-layered defense strategy for organizations that hold a large amount of sensitive data.
- To know how we can save ourselves from being infected by this botnet attack.
- To know how to prevent the vulnerability of devices so that cyber criminals cannot exploit them.
- To save devices from malware infection.

1.2 Research Challenges

People, companies, and governments are becoming extremely apprehensive about cyber security. Among the most challenging obstacles in cyber security is keeping our system confidential in a world while all of it is online, particularly our credit card details and charming kitten videos. Ransomware, phishing, malware attacks, and other cyber security threats can take various forms. India is ranked 11th internationally for local cyber attacks, and there have been 2,299,682 of them so far in 2020's first quarter. [47]. Cyber security issues may range from little ones like out-of-date software to major ones like a lack of support from top personnel. The following is a list of the most typical issues that information security professionals and organizations must handle.

1. **Recognizing that you are a target:** Small firms are often ignorant of the attractiveness of their assets and data to hackers. The bulk of enterprises in our modern economy possess things that attackers covet: information and cash. Every size of business is susceptible to cyber threats. According to Kevin Raske, a marketing specialist for cyber security, a fundamental grasp of cyber security best practices would significantly help many firms [17]. It requires constant awareness of your vulnerability. Human error is the leading cause of security breaches. The first step in developing a defense is acknowledging that attackers may target your organization.
2. **Data breaches due to remote work:** Compared to last year's \$3.86 million, IBM's 17th Annual Data Breach Report estimates that a data breach costs an average of \$4.24 million. The most recent edition of this research examines the consequences of a remote workforce [50]. As indicated in the reporting remote:

- Employees were involved in 5% of data breaches.
 - Remote employment contributed to a \$1.07 million rise in the average cost of a data breach.
 - It took firms with more than 50 percent of employees working remotely 58 days longer than usual to discover and manage breaches (287 days).
3. **Missing security patches:** Essential for fixing security vulnerabilities are patch updates. Regular system updates minimize potential cyber threats. A successful cyberattack can ultimately be linked to the absence of security controls, an unsafe configuration, or a lack of security awareness.
4. **Losing sight of the backup plan:** Any data loss may jeopardize your personal identity, obliterate your family's history, and possibly destroy your whole business. Whether you keep years of extremely sensitive client data or a big number of images of your puppy, you never want to discover that a significant portion or all of your data has been lost.
- Here is some way to prevent it:

- Utilize patch management best practices and maintain software updates.
- Utilize antivirus software and maintain its current version.
- To avoid accumulating a substantial amount of technical debt, you should include funding to replace or update IT in your purchasing plan.
- Segment the network to prevent the transmission of malware throughout the whole enterprise.
- Reduce the number of access points for hackers into a network by closing unused ports.
- Block incoming traffic from Tor exit nodes to prevent darknet hackers from launching attacks.

These are some scientific difficulties that must be addressed in the near future if cyber security is to become a reality. Also essential is research into cutting-edge botnet architectures that adversaries can rapidly create. Otherwise, we will continue to be susceptible to Internet virus assaults.

1.3 Motivation

Cyber security is the process of protecting and recovering from cyber attacks against computer systems, networks, devices, and applications. In addition, cyber attacks offer a complex and ever-evolving threat to your sensitive data, as fraudsters employ social engineering and artificial intelligence to develop new techniques to circumvent standard data protection mechanisms (AI). Cybercrime will become increasingly prevalent in 2022. Based on a study, cybercrime appears to affect the security of much more than 80% of global businesses, and even the rate of these offenses is expected to continue to increase in the future year. Attributed to the justification that this sort of criminal activity occurs in the virtual world, authorities' ability to monitor cyberattacks like botnet attacks are limited[8]. Botnets evolve rapidly in

order to exploit vulnerabilities and security shortcomings. The proliferation of IoT devices with IP addresses has made it easier than ever for botnets to propagate. IoT devices are generally more susceptible than personal PCs and have lower security safeguards. Thus, The prevention of cybercrime is crucial and must be addressed immediately. The main issue in detecting peer-to-peer botnets is mainly when botnets exploit the infrastructure of existing P2P networks . In this paper, we demonstrate a detection method that relies on modeling the time-varying emergence in the amount of substance of peers sharing a resource in a P2P network. This enables the detection of aberrant activities associated with parasitic P2P botnet resources in systems of this type. After all, We conduct extensive trials on the Mainline network, from which we acquire encouraging detection findings and tentatively identify patterns of parasite botnets.

Chapter 2

Literature Review

The review of the relevant literature provides a synopsis of the discussion on the most common problems associated with peer-to-peer botnet assaults across various platforms. Later, we will talk about related works done to detect P2P botnets. Index poisoning, Sybil assault, periodicity behavior analysis, mutual contact graph, community detection, blockchain, Software Defined Networks, and many other methods have all been explored by researchers in the quest to detect botnets, both centralized and decentralized, also known as peer-to-peer botnets. The application of supervised learning algorithms and the monitoring of net flow records over an extended time has been shown in many of these investigations to improve accuracy. In order to better comprehend botnets, we must define specific fundamental terminology. After that, we will walk you through a historical recap of the most significant moments related to bots and P2P protocols. Our research indicates that P2P botnets will soon emerge as a significant security risk in the Internet environment [10].

2.1 Peer to Peer Botnet

Because of their ease of use, centralized Botnets have been adopted by many different types of Botnets. IRC Botnets and HTTP Botnets are two of the most typical forms of cyberattacks. The security community, however, can benefit from botnets due to the centralization that they bring about, which in turn provides a single point of failure. [10]. Neither the bots nor the botmaster would be able to communicate with one another if the command and control server went down.

The terms "peer-to-peer," "bot," "botnet," and "botmaster" are defined below:

1. In a P2P system, each node performs the functions of both client and server.
2. The term "bot" refers to computer software that can carry out user-centric tasks automatically and without human intervention.
3. The term "botnet" refers to an online group of malicious robots that work together to illegally take over a computer system.
4. A botmaster is a person in charge of directing botnets to carry out remote processes.

2.2 Functioning of peer to peer botnet

P2P botnets are decentralized since they do not rely on a CC server to coordinate their activities. Bot software lists trusted computers and drop-off places where infected devices can update their virus. This list may include more computers that have been compromised. More sophisticated botnets utilize encryption in order to conceal communications between individual bots. Compared to a traditional botnet architecture, decentralization makes it significantly more challenging for security experts to acquire insight into botnet communication patterns. This is done to facilitate botnet evasion of discovery [38] Since there is no CC server, it is less likely that discovering one bot will lead investigators to take down the entire network.

The bootstrapping process used to construct a P2P botnet is an index-based botnet [5]. The bots in the botnet need to be ready to take further orders from the botmaster, like starting an attack or updating to a revamped version of the bot after the botnet has been set up. That will take place during the CC phase of the botnet, which is the most crucial step since it establishes the network topology and the resistance of the botnet to security measures. When sending commands, P2P botnets also leverage P2P traffic indexes. Two types of mechanisms are at play during the CC phase: pull and push. Pull orders from the botmaster is a possibility for robots. P2P only requires a peer to send a request message for the necessary file, as opposed to centralized botnets that handle this; at that point, the system will decide how to fulfill the request best. If the query message is still valid when the peer receives it and returns it with the command encoded, the search will continue until the file is found. With a push mechanism, robots wait for orders to be sent and then passively relay them to other robots.

The bots will follow the instructions to carry out their evil deeds during the attack. The botmaster will probably have plans to build a new botnet later if the botnet is compromised during the attack.

P2P file-sharing platforms have made it harder to detect these bots since their activity can blend in with real users and spread quickly over the network[35].

2.3 History

This first table summarizes some of the most well-known P2P protocols and bots. The first bot in the timeline is EggDrop, and the last is the Trojan.

The peer-to-peer bot Peacomm has just been released. There have been tremendous advancements in harmful bot technology in recent years. In particular, the first P2P bots like the Trojan have begun development. There is no timeline for the Peacomm bot worms in Table 2.1. Bots may be disseminated via worms.

While worms do help set up botnets, their main impact is on the initial spread of infected machines rather than on the bots' later ability to communicate with one another. We then focus on the communication mechanism when the botnet has al-

ready infected a large number of users. Worms are introduced in [5] by Kienzle et al.

The EggDrop bot was created in the very early days of the World Wide Web. There were, without a doubt, far more bots in the world before EggDrop.

Nevertheless, In the history of IRC (Internet Relay Chat) bots, EggDrop is considered to be a pioneer. Some legit applications of EggDrop encompass playing games (like the Turing test), correlating the transmission of files (legally shared files), automating the orders of the channel administrators, etc. Thus, it appears that the earliest bot inventions were motivated by a desire to improve Internet automation.

GTBot variants are among the first widely publicized harmful bots. There have undoubtedly been countless harmful bots in the past. It is important to note that mIRC.exe, an IRC client, was included in several GTBot versions. [7]. This bot demonstrates some of the earliest uses of IRC for botnet coordination.

We think that the launch of Napster contributed to the rise in popularity of P2P protocols, rather than botnet activity. Napster's client was designed so that users could browse for and download music from other users on the same network. Napster is not genuinely peer-to-peer since central servers were used to index files. Connecting to the central server would allow users to both indexes their own files and do searches across many users' computers. If a person located a file they needed, they might download it by establishing a direct connection with another peer. Since a significant number of music files distributed between users on Napster were unlawfully traded, a judge concluded that the service was illegal and ordered the site to be shut down.

Newer iterations of P2P file sharing attempted to elude authorities by eliminating the need for a central server.

After Napster was shut down, totally decentralized peer-to-peer systems began to emerge, albeit not solely as a result of this event. Since the Gnutella protocol is completely decentralized, it represents the start of peer-to-peer services. As seen in Table 2.1, various peer-to-peer protocols have been created to be as resilient, efficient, and dependable as feasible since the debut of Gnutella. Chord [1] and Kademlia [3] are two modern P2P protocols that propose using a distributed hash table to efficiently find data in P2P systems.

The architecture of P2P networks is ripe with potential entry points for cybercriminals.

The sophistication of malicious bots has grown significantly in recent years. Due to its well-designed and flexible code, Agobot variations are likely among the most widespread bots [32]. According to us, Agobot represents a tipping moment in which botnets have become a greater concern. Table 2.1 shows the extensive research and development being done on peer-to-peer bots right now. Some P2P bots have been built on top of preexisting P2P protocols, while others have attempted to construct their own. Peer-to-peer botnets will likely grow more prevalent than

typical decentralized CC designs, according to our forecast.

Date	Name	Type	Distinguishing Description
12/1993	EggDrop	Non-Malicious Bot	Recognized as early popular non-malicious IRC bot
04/1998	GTbot Variants	Malicious Bot	IRC bot based on mIRC executable and scripts
05/1999	Napster	Peer-to-Peer	First widely used hybrid central and peer-to-peer service
11/1999	Direct Connect	Peer-to-Peer	Variation of Napster hybrid model
03/2000	Gnutella	Peer-to-Peer	First decentralized peer-to-peer protocol
09/2000	eDonkey	Peer-to-Peer	Used checksum directory lookup for file architecture
03/2001	Fast Track	Peer-to-Peer	Use of supernodes within the peer-to-peer architecture
05/2001	WinMX	Peer-to-Peer	Proprietary protocol similar to FastTrack
06/2001	Ares	Peer-to-Peer	Has ability to penetrate NATs with UDP punching
07/2001	BitTorrent	Peer-to-Peer	Uses bandwidth currency to foster quick downloads
04/2002	SDbot Variants	Malicious Bot	Provided own IRC client for the better efficiency
10/2002	Agobot Variants	Malicious Bot	Incredibly robust, flexible and modular design
04/2003	Spybot Variants	Malicious Bot	Extensive feature set based on Agobot
05/2003	WASTE	Peer-to-Peer	Small VPN style network with RSA public keys
09/2003	Sinit	Malicious Bot	Peer-to-peer bot using random scanning to find peers
11/2003	Kademlia	Peer-to-Peer	Uses distributed hash tables for decentralized architecture
03/2004	Phatbot	Malicious Bot	Peer-to-peer bot based on WASTE
03/2006	SpamThru	Malicious Bot	Peer-to-peer bot using custom protocol for backup
04/2006	Nugache	Malicious Bot	Peer-to-peer bot connecting to predefined peers
01/2007	Peacomm	Malicious Bot	Peer-to-peer bot based on Kademlia

Table 2.1: Timeline of Peer-to-Peer Protocols and Bots

2.4 Current Scenario

The evolution of a botnet may be broken down into three distinct phases. Finding hosts is the initial order of business, followed by creating the botnet and keeping an eye out for further directives. There are some similarities between site botnets, leeching botnets, and P2P botnets, but there are also some key characteristics that distinguish them differently. By knowing where botnets are most likely to launch their attacks, security personnel may be able to stop them in their tracks. Both information security experts and would-be botmasters are fascinated by the prospect of a P2P botnet. The acronym "P2P" is commonly used to shorten the phrase "peer-to-peer." The next stage is to train workers to put these three safety measures into action. The effectiveness of Sybil's rebellion and potential defensive methods against index poisoning is also factored into this study. Attackers can use botnets, which are networks of infected computers numbering in the hundreds of thousands or even millions, to carry out their malevolent plans. Distributed denial of service attacks may be launched with the help of a botnet. Botnets are frequently used by attackers. Denial of service (DDoS) assaults, phishing, click fraud, bitcoin mining, brute force password attempts, and social media site hacking are all facilitated by the prevalence of distributed networks of computers and networks. There has been a recent trend among botmasters to include P2P communication protocols like Kademlia, Bittorrent, and Overnet into the frameworks upon which their botnets are

built. This is done so that there is no single point of failure in the botnet. Threats posed by botnets to the Internet’s monetary and social infrastructure are universal and should be taken extremely seriously. P2P-based, decentralized botnets have only been operational for a short while, but their popularity has skyrocketed in that time. These botnets function on a peer-to-peer paradigm. P2P botnets are evolving from a wide range of command and control protocols and hybrid architectures, growing increasingly sophisticated and covert. Many of the existing methods for detecting botnets rely on clustering, which hinders their real-time performance. Clustering is the reason why this is the case.

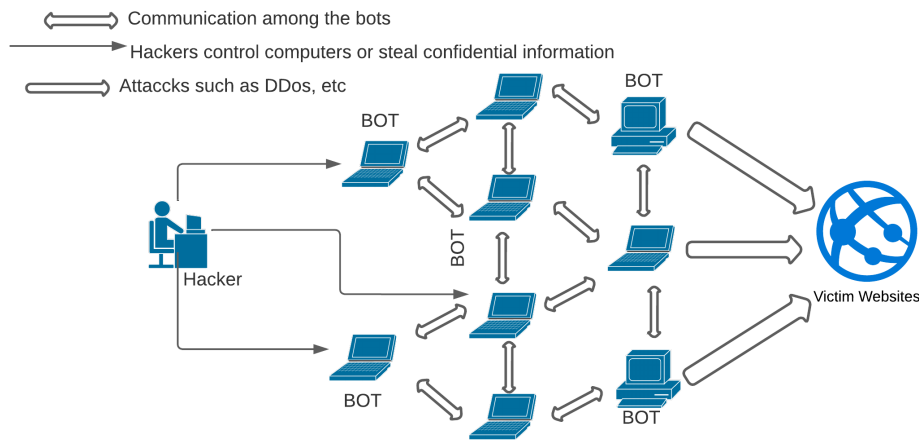


Figure 2.1: Peer to Peer botnet operation

2.5 Detection and prevention of peer to peer botnet

2.5.1 Honeypot and network traffic analysis

It is necessary to put a stop to the propagation of bots in order to defend networks against their effects. P2P botnets, on the other hand, make this procedure more difficult because there is no centralized location that can detect and halt them. To prevent further botnet proliferation and future assaults, scientists are researching technologies to identify botnet communication. This is done with the goal of preventing botnets from being used in cybercrime. Honeypots and network traffic analysis are the two primary methods utilized in botnet defense [36]. Honeypots are also utilized as one of the key ways to botnet defense.

It may be possible to detect active botnets in networks by monitoring network traffic, gathering data on the characteristics and activities of botnets, and creating a standardized model for them. Defenders may utilize this approach to identify and discover botnets. The botnet model in this instance is predicated on the occurrence of several network aberrations, including unusually high network volume of traffic, unexpectedly high system latency, and anomalous network traffic through

ports. Sites that behave maliciously and communicate directly in a way that is similar to how other hosts do can be rapidly and easily identified by using a common model. This method may be useful for locating known botnets, but it is not nearly as effective when it comes to locating newly created botnets. On the other hand, honeypots are helpful for evaluating the characteristics of new botnets, but they are not efficient at finding infected applications. Honeypots can be found online. Consequently, defenders typically combine the two strategies in order to detect botnets and identify the CC techniques used by the botnets [27].

2.5.2 PeerGrep

PeerGrep is a cutting-edge system that can find P2P bots. It gathers flow logs at a monitored network's perimeter and pinpoints interior hosts that have been infected by P2P bots. Surprisingly, their method attempts to find P2P bots on networks with as little as one bot or when traffic from P2P botnets coexists with traffic from legitimate P2P apps (like Skype). To do so, PeerGrep uses a two-step procedure. All hosts that participate in P2P communications, whether they are good or bad, are detected in the first step. By examining their network behavior patterns, PeerGrep separates P2P bots from other harmless P2P hosts in the second stage. PeerGrep can be installed on the border router of the monitored network in the real world to gather network data and spot suspected P2P bots.

This paper makes the following contributions:

- By using persistence and periodicity characteristics, they present a technique called PeerGrep that can detect long-lasting P2P activity while having a low false positive rate.
- For P2P bots that exhibit persistent and cyclic network activity, they suggest an effective technique for detecting them. This method will still be able to identify it even if there is just one bot on the monitored network or if both a malicious and a trustworthy P2P application are running on the same server.

Due to the rarity of seed bot data, the suggested approach, PeerGrep, does not necessitate its use [55]

2.5.3 Software defined network

Therefore, in SDN, network components do not need to be pre-programmed with the required forwarding rules, as the access point and data plane are kept away. They either forward the packets or discard them based on the rules set by their controller. Process

- **Fields to packets:** to do a matching operation matching precedence of the flow item, or "priority".
- **Packet-matching counters:** must be incremented on a regular basis.
- **Directives:** to alter the processing of the action set or pathway.

- **Timeouts:** to define the maximum amount of time before the switch’s flow termination.
- **Cookie:** The controller will employ this in order to restrict flow monitoring, flow editing, and flow deletion.

Almost any OpenFlow controller may be used to create software-defined networking since it is associated with SDN modules (SDN). Modules are responsible for determining the best routes for packets to travel from one end host to another, and these routes are based on the forwarding rules. OpenFlow switches allow for regional flow table and forwarding rule customization via an API that the modules use to communicate with the controller (API). Because of this, OpenFlow controllers supply OpenFlow switches with an intuitive and easy-to-use infrastructure for managing flow entries.

First, the Detection Agent classifies the incoming data packets into several categories before attempting to identify flows. Assuming we obtain flow-level data, we can next probe the network’s actions by extracting characteristics from the flows themselves. To put it another way, this will help us figure out how the network works. Based on machine learning models built with real-world samples of both P2P botnet traffic and safe P2P application traffic, the Detection Agent labels incoming data flows as either a P2P botnet or a relatively safe P2P application. The models were educated on information gleaned from both malicious P2P botnet traffic and benign P2P application traffic, thus the separation. The Detection Agent will notify the Rule Arbitrator with an alert message whenever a flow’s classification is complete. The alert will include both the flow categorization and the 5-tuple data. Following receipt of a report from the Detection Agent, the Rule Arbitrator makes the necessary changes to the flow tables inside the Data-link Bridges. If a Data-link Bridge receives a packet flagged as containing hazardous data, it will immediately reject the packet. Even though it’s the last, this is a crucial element[56].

2.5.4 AutoBotCatcher and mutual contact graph

AutoBotCatcher’s design is built on the idea that bots in the same botnet have substantial sociability with one another, leading to the formation of communities. The program known as AutoBotCatcher is, thus, to identify botnets by examining the traffic patterns of communities of IoT devices that have developed over time. With P2P botnet detection technology using blockchain to IoT, authorized agents may track the flow of IoT network traffic through their own individual subnets and share the data they collect as blockchain transactions. Multiple-computer botnets can be uncovered with this method. Obtaining this approval is a crucial step in the planned action. Conversely, block generators leverage existing network traffic to simulate the interconnections between IoT nodes mechanically. The block generators create a virtual version of these connection webs. Information on the locations of the various neighborhoods is recorded using a graph. As an example, AutoBotCatcher applies the Louvain technique to interaction graphs to make community identification. With the help of the states that a BFT blockchain keeps track of, AutoBotCatcher can take snapshots of the graph of mutual connections. AutoBotCatcher developed an authorized blockchain based on the BFT protocol. Limiting

the ability to add new blocks to the chain to just the nodes that have been verified as trustworthy, ensures that only the nodes that have been confirmed to be trustworthy are involved in the process by which the network reaches a consensus. This ensures that only block producers can access the blockchain and its recorded information about the network.

2.5.5 Community detection

Mutual connections between bots in the same botnet are more common than between hosts in a random P2P network. The bots in the botnet all follow the same instructions and spread the same propaganda for this reason. P2P bots display community behaviors and erect structures for the community that are perhaps utilized to detect and counter botnets. Such structures and behaviors can be used to detect the existence of botnets. AutoBotCatcher's ability to analyze communities by tapping into preexisting networks is a major selling point[54].

2.5.6 Decision tree and multilayer neural network

This study's major objective is to create a system for detecting P2P Bots that use the network's throughput statistics. As time goes on, the neural network becomes more and more contentious. It is a powerful tool for recognizing and controlling nonlinear systems due to its innate capacity to provide an estimate for any nonlinear problem. As a result, it is a powerful tool for monitoring and controlling complex systems. Such as Input, hidden layer(s), and output layers are common components of networks. Furthermore, the number of neurons in each layer remains constant throughout the whole structure. Neurons in the same brain layer may share an input and output connection if both neurons are active at the same time. A person's mental link to another person or thing may be compared to the gravitational pull of an item. A neural network is trained by repeatedly presenting it with fresh data sets and the corresponding outputs. This is done at various points throughout the training process. The network is provided with data collection and tasked with drawing an inference from that data. It is possible to assess the accuracy of a neural network's calculations by analyzing the results it produces.

The following are some key aspects of the approach proposed here. It catches Bots in the early stages of their spread before they can cause any damage. The system does not require deep packet inspection to do signature matching and can do it without reviewing the entire network's data (DPI). It detects Bots on any host system, port, or IP address. Finally, here are the things we presented below:

- A proposed framework's performance can be improved with the help of a defined method for decreasing network traffic.
- The information needed to do a connection-based detection comes just from the TCP control packet's header, making it payload-agnostic. Therefore, it is distinct from payload encryption methods and does not rely on deep packet inspection.
- We may reduce the dataset's size and complexity by using classification and regression trees to prioritize and pick out the most useful connection information.

- Recognizing and separating malicious P2P Bot activity from normal network traffic [51]

2.5.7 Focused View of Detection methods

Selected detection methods are widely used and these are the most recent methods that we found. Also, their execution methods are distinguishable according to their detection procedure. Because of this, we are summarizing those method in following-

Name	Founder	Year	Main Detection Method
AutoBotCatcher	Gokhan Sagirlar, Barbara Carminati, Elena Ferrari	2018	Block Chain
PeerGrep	Pengfei Wang, Fengyu Wang, Fengbo Lin	2018	Behavior Analysis
Honeypots	Meerah M. Al-Hakbani , Mostafa H. Dahshan	2015	Traffic Rerouting
SDN	Shang-Chiuan Su, Yi-Ren Chen, Shi-Chun Tsai , Yi-Bing Lin	2017	Dropping Malicious Packets
Resilient Neural Network	Mohammad Alauthaman1, Nauman Aslaml, Li Zhang1, Rafe Alasem2, M. A. Hossain	2018	Decision Tree

Figure 2.2: Summary of detection methods

By comparing the systems across a number of criteria, we will establish which detection method offers the most precise identification of the P2P botnet.

Chapter 3

Related work

Techniques for P2P Botnet detection and prevention have become more popular in recent years. Finding the compromised computer before it is used for malicious purposes is more important than understanding how a bot infects a computer. P2P botnets are becoming more widespread, which has sparked a ton of research into how to find and eliminate them.

Multiple methods have been developed to identify botnets. Some of these strategies are based on signatures, differences, networks, the Domain Name System (DNS), and data mining [18]. P2P Botnet detection systems may be broken down into three main areas according to the work of other specialists, such as Han et al. [29]. These are data mining, machine learning, and network behavior and traffic analysis. The techniques for detection may be broken down into 2 section: host based methods [19],[43] network based methods [46],[45],[41],[35],[28][14]. Since host-based solutions function similarly to antivirus software in that they identify bot activity on the host system, individually monitoring each host in a real network might be challenging. contrasted with network-based approaches, which use information gleaned through traffic analysis, which is based on a few key parameters. Network-Based tactics are the most often used because of the associated deployment brevity There are several network-based tactics. involving two groups: methods that make use of network traffic signatures and methods based on collective behavior. These methods are based on network traffic [41], [35], [46], and [48].

Peer Rush was recently introduced by Babak et al. [42]; it uses just one strategy to differentiate between healthy and malicious P2P traffic. For simple classification problems with just two possible outcomes, we use Parzen , KNN, and Gaussian data description classifiers [2]. When developing a profile for an application, the first step is to analyze traffic samples taken from other well-known P2P applications. In order to uncover P2P Botnets, Zhang et al. [30] developed a stealthy technique. The proposed method uses CC traffic to track down Bots. They pull out bits and packets transmitted and received for each communication path. BIRCH and hierarchical clustering are used to categorize network traffic. The method has an FPR of 0.2, a TPR of 100%, and doesn't need payload signatures to identify both malicious and benign hosts. This approach can detect P2P Botnets but not IRC or HTTP Bots, despite the fact that it can identify Botnets regardless of how they do illegal behaviors. The suggested solution employs DGA and fast-flux algorithms to make sure CC interactions are very secure, however, they may be bypassed via

flow interruption packets and other techniques. By measuring packet size, Liao et al. [19] were able to differentiate between Botnet activity and typical P2P traffic. The findings of their investigation are shown below.

Rather than idly collecting dust, P2P bots are actively involved in maintaining the relevance of information shared between other bots. The bot's data transmission is very slow. Traffic on a network may be sorted using a variety of methods, including Bayesian networks, Naive Bayes, and J48. Accuracy levels for the three algorithms ranged from 87% to 89%, and finally to 98%. $P - 2 - P$ Botnet data are smaller than those of other P2P software.

A host-based technique for detecting Bots was developed by Fedynyshyn et al. [25] by keeping an eye on how long they stayed around. In their C C model, they classified Botnet infections using a J48 classifier and a random forest approach (IRC, HTTP P-2-P). Bot command and control networks were also found to have many design elements with standard network traffic. Zhang et al.[37] developed a method in 2014 [19] for increasing the scalability and efficiency of systems. There are two aspects to this method: [18] detecting computers that may be involved in Isolating hosts as hosts or bots requires two things: [18] P2P connections, and [29] the gathering of statistical thumbprint, resulting from profiling P2P traffic. Website visits were boosted by P2P file-sharing programs including eMule, LimeWire, Skype, and Bit Torrent.

While under supervision, Waledac and Storm generated malicious network activity. The method uses a hierarchical clustering of P2P flows to identify Botnet P2P traffic with a 100% detection rate and a 0.2% false positive rate. Bot traffic is successfully identified with this method, which is useful since Bot activity often overlaps with normal host traffic and can be readily identified. For detecting malicious P2P Botnets, Zhao and Traore [32] suggested a harmful fast-flux network model. Characteristics of Botnet operations are evaluated using network data, allowing for distinctions to be made. The use of a decision tree method leads to more precise results.

To identify P2P botnets, Saad et al. [26] used machine learning and flow. There were 17 distinct features of the botnet's means of communication and command that were identified. The detection of IRC Botnets was evaluated using a number of different machine-learning strategies [4]. A 95% detection rate, 3% false (+)rate, and a 5% false (-) rate are all shown by the classifiers. To sum it all up, boosted decision trees are the most effective method. Community behavior-based approaches [45], [36] examine bots in the same P2P botnet community. Li et al. [25] suggested discovering highly linked subgraphs to detect P2P communities. This strategy centered on a backbone network, which required a vast communication graph. Yan et al. [45] presented a P2P botnet detection approach using group-level behavior analysis. P2P sites, which are susceptible to botnets with dynamic or randomized traffic patterns, were only investigated using statistical traffic aspects to cluster them. Their technique can't handle P2P botnets. Deep packet inspection (DPI) is a technique used by some proposed methods, such as [13], to assess the contents of network packets. Encrypting CC channels bypasses these tests. Network traffic signature-based techniques aren't ideal for dynamic IoT scenarios.

Sybil's attacks on botnets were studied by Davis et al. [12] and Luo [30] using a variety of settings. Both decided how to render the botnet susceptible to Sybil assaults. A method for locating honeypots in botnets using sensor technologies was presented by Costarella et al. [33]. These sensors detect a node that doesn't transfer malicious traffic, unlike honeypots. Zhao ' Traore [32] developed a P2P Botnet detection method based on fast-flux network behavior. They employ network traffic analytics to detect Botnet traffic. The decision tree algorithm yields great accuracy. Rahbarinia et al. [41] presented a botnet detecting method. They found and categorized P2P botnet activity. The distinctive network traffic patterns that various P2P apps display served as the foundation for their detection system. Each P2P application has a categorization profile that is created in order to measure network traffic. Samples of network traffic from the Storm, Zeus, and Waledac botnets, as well as normal P2P traffic, are taken for testing. Using machine learning, Tey was trained and made predictions. The results of their tests indicate that their solution successfully classifies network traffic. Considering this study, our implementation of SDN for network administration. For P2P botnet detection, Narang et al.'s [40] only relied on information from the TCP/ UDP /IP header. As,they are successfully classified traffic using machine learning techniques and had a high success rate. In contrast to SDN, their employment requires them to examine a wide variety of traffic data. Attack traffic characteristics are used by network-based algorithms[46], [28], [41],[42], [35] to identify hosts exhibiting the same abnormal behavior on the network. An example of a graph-based technique that focuses on the social behavior of botnets is described in Entelecheia [35], which analyzes long-lasting, infrequent flows. BotMiner [14] can tell that a group of hosts are all part of the same botnet if their ways of communicating are similar and they all do the same harmful things at the same time. But some bad things can be done in secret and without common communication patterns, making it hard for BotMiner to figure out what's going on. PeerRush [41] uses signatures to identify the source and category of P2P traffic.

This traffic can come from many different P2P apps, including some that are harmful, like P2P botnets. Techniques that emphasized how people behaved in communities and groups. Several studies [5][1][3] use analysis of group and community behavior to find botnets. In a P2P network, for example, [36], like us, uses the network traffic flow of hosts to find mutual contacts to find bots. While analyzes network data flow at the group level using Support Vector Machines (SVM). However, these techniques can only be used to find known types of bots.

Since new botnets are constantly being developed for the Internet of Things (IoT), they are inappropriate [49]. PeerHunter [50], in contrast, does not rely on recognized bot categories when using the Louvain technique to analyze community behavior at the network flow level on a mutual contacts graph. PeerHunter, on the other hand, only uses the collected network traffic flow data for static botnet identification. This isn't enough for an IoT environment that is always changing and needs dynamic botnet detection.

Chapter 4

P2P Botnet Detection Systematic

Finding and disabling any botnet that could be active on our network is the first and most critical step in protecting both ourselves and our network from potential threats. Recently various botnets have been dissected in order to ascertain the distinguishing qualities shared by various botnets[18]. P2P botnets are a direct result of this, and as a direct consequence of this, a broad array of strategies to identify and battle P2P botnets have been created. It is true that the current methods for recognizing and combating botnets contain some intriguing ideas, but the capacity of these methods to deal with actual botnet threats in the real world is greatly hindered by these methods.

The monitoring and detecting of peer-to-peer botnet activity has been achieved through the utilization of a wide variety of detection approaches such as signature-based, honeypots, behavioral analysis, DNS traffic-based, community-based detection, graph-based detection, and a number of other methods in addition to these. These approaches have been combined with a number of others to create a comprehensive detection system. In accordance with these tactics, several researchers have developed their very own one-of-a-kind peer-to-peer botnet detection methodologies and algorithms, many of which are already being implemented in the well-established anti-threat systems that can be found all over the world. A comparison of some of the best approaches is the fundamental contribution of our study. This is followed by the derivation of the analytical result of the comparison and the selection of the best strategies among the offered models and algorithms. As a consequence of this, we choose some of the most effective methods of detection, such as AutoBot-Catcher[54], PeerGrep[55], Honeypots[44], SDN[53], and Neural Network[61], and evaluate them based on a variety of parameters, including efficiency, accuracy, the time needed, advantages, downsides, and constraints.

We began by isolating and implementing the several algorithms required for each p2p botnet detection strategy on their individual one-of-a-kind datasets. This marked the beginning of the process. After finding the right methods at the beginning, we were able to complete this task successfully. To accomplish this goal, the algorithms were disassembled into their parts and then put back together again. After isolating and understanding the core algorithms that were at the heart of each approach, we were finally able to complete this task in an efficient manner. Completing some manual analysis and comparisons of the data we had received up to this point in

the process was the next step in our approach. This was done with the data that we had obtained. For us to go to the stage that came after this one in our technique, this step had to be completed. In the end, we are able to build a system that is capable of evaluating both tactics while still maintaining the detection parameters that were established at the outset of the project.

4.1 AutoBotCatcher

The concept of "communities" formed by AutoBotCatcher's bots stems from the idea that bots belonging to the same botnet form networks[70]. AutoBotCatcher is able to identify botnets by detecting the communities of IoT devices through the network traffic flow. Detecting P2P botnets in the IoT that use blockchain technology, it employs agents and block generators. Subnets of the Internet of Things may include agents keeping tabs on things, and such agents might report their findings as blockchain transactions. The network traffic flows between IoT devices are used as the "blocks" in the block generators to depict the "connections" between the devices. . AutoBotCatcher takes advantage of Louvain to find clusters of related nodes in networks of indirect connections[11]. A fresh snapshot of the interconnected network is generated automatically whenever the contact details of IoT devices are modified. AutoBotCatcher stores snapshots of graphs representing mutual connections on the BFT blockchain, using the blockchain states as identifiers. The BFT blockchain used by AutoBotCatcher is a permissioned network in which the nodes that create blocks have already been verified. Only the blockchain network itself is accessible to the block creators.

There are two components: Agents and Block Generators. Agents serve as entry points into a network. Using network data transactions (NTs) and shutting down infected devices, AutoBotCatcher's agents keep tabs on IoT network activity on their subnet. Agents from AutoBotCatcher may be relied upon. Companies that make or sell devices, internet service providers, and government agencies all have a role in the generation of blocks. To identify P2P botnets, we assume that large enterprises devote sufficient computing and network resources to act as block producers.

4.1.1 Network Data Transaction

AutoBotCatcher uses a community detection approach that runs on the mutual contacts network in order to locate botnets. This method is based on the concept of mutual contacts. Gathering and analyzing meta-data on the net flow is a prerequisite for AutoBotCatcher in the process of creating a mutual connections graph. Because they are connected to a network, The Internet of Things devices are capable of doing this. The transactions that agents have uploaded to the blockchain are what they mean when they refer to meta-data. The transaction pool of the blockchain is a common database that all of the block producers are responsible for administering in order to store pending transactions. Auditing procedures are carried out with the use of this. To be more specific, each agent only interacts with a single block generator, and the block generator that gets a transaction will then broadcast that transaction

to all of the other block generators that are a part of the network. A network transaction code (NT) may be created using the following formula: NetFlow, which stands for "network traffic flow" may be written as

$$NetFlow = (IP_{src}, IP_{dst}) \quad (4.1)$$

where IPsrc and IPdst are the sending and receiving IP addresses of the devices involved in the flow of data. The following is an acceptable method to describe it if we take TxPooladdr to be the key of the transaction pool to which the sending agent is associated, and Deviceaddr to be the sending agent's one-of-a-kind public key: An explanation of the tuple structure of network data transactions is provided as follows: In accordance with this technique, NT corresponds to the same value as Deviceaddr, IPsrc, IPdst, and TxPooladdr. Block generators are responsible for grouping transactions into blocks. Throughout the rest of this text, we will refer to blocks as bm. Here m is the corresponding block number. Based on this, the block definition is as follows:

$$bm = NT_x, \dots, NT_y \quad (4.2)$$

Where, x, and y are the respective transaction numbers associated with the network data transaction represented by NT. To create a block, AutoBotCatcher selects a leader from among the block creators. The time that elapses between two successive block creations is denoted by the symbol.

AutoBotCatcher analyzes P2P botnets on a regular basis each time a new batch of blocks is produced. The time periods in question are called "rounds" here. Each round requires a given length of time, denoted by Δt . The worth of Δt is contingent on many factors, including the network's overall health, the blockchain's protocol, and the length of time it takes for block generators to complete botnet detection procedures on blocks. A round is denoted by the notation t , where t is a timestamp indicating when the round began. After a period of time t has passed, the new period, $t+1$, will begin. When round t comes around, the NTs from round $t-1$ is used to identify any botnets that may have been present. To be more specific, NTs transmitted between $t=t-1$ and $t=t$ are handled in round t .

Condition of an IoT device is a table containing information on the IP addresses and communities of other hosts with whom it may establish connections. AutoBotCatcher's state is not reserved on a blockchain but in an efficient Merkle tree[21]. A blockchain may change to a new valid region once a sequence of transactions have been completed in the previous state. AutoBotCatcher finds botnet networks by taking a snapshot of the previous blockchain's mutual connections graph. There is a transition in AutoBotCatcher's status after each iteration of botnet detection jobs. Assuming the aforementioned conditions are true, one state transition will occur for each round when the sequence of blocks is executed. When a round t completes, it moves from the t state to the $t+1$ state, where a new set of blocks has been added after processing. They characterize the evolution of the state like a system of rounds, where the state and the freshly created blocks serve as inputs. The set of freshly formed blocks is $[B_m, \dots, B_{m+z}]$, where m and z are integers denoting the block numbers, and t is the starting condition.

4.1.2 System Architecture

In order to do collaborative botnet detection in real time on big networks, AutoBotCatcher makes use of a permissioned BFT blockchain as a back end module. AutoBotCatcher’s execution sequence is shown in Figure 4.1 and discussed in detail below.

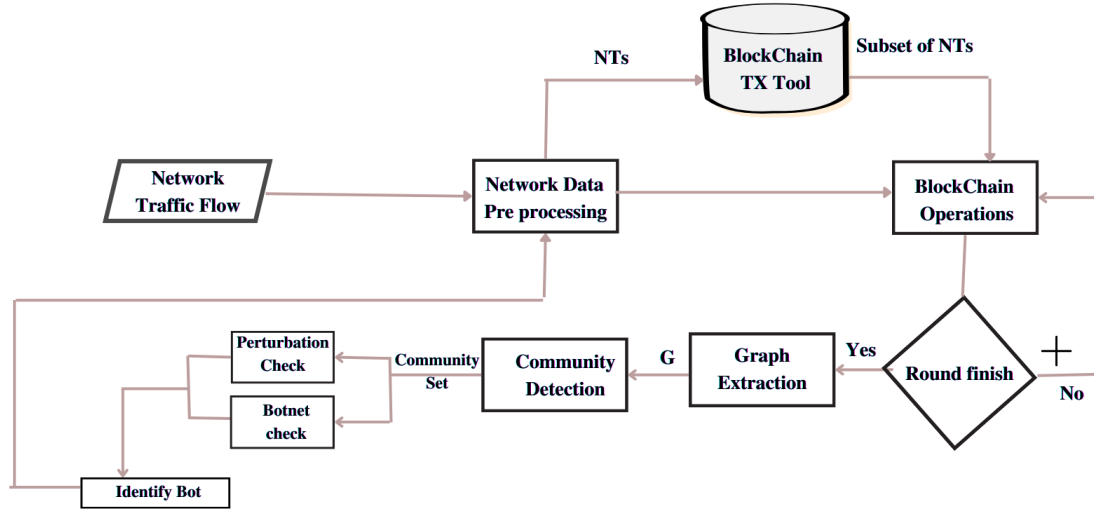


Figure 4.1: System Flow

To keep an eye on the flow of traffic from IoT devices, agents must first do data preprocessing inside the network. Agents maintain a blacklist and a whitelist of IP addresses in order to carry out these duties. Those Internet Protocol addresses that have been determined to be part of a botnet are recorded on blacklists. A whitelist, on the other hand, is a collection of IP addresses that are trusted by all devices on a network to be secure, such as the IP of the provider’s servers.

Blockchain activities are carried out by block generators, who create and transmit blocks containing NTs. First, in order to complete these steps in a value of t , a single block creator is chosen to produce many blocks. Each block is created by one block generator using a subset of the NTs from the total number of transactions on the blockchain. For a BFT blockchain to reach consensus, each block must be validated by two-thirds of its block producers, as we remember.

The mutual contacts graph G is built or maintained by block generators using graph extraction, which utilizes information on network traffic flows or processed NTs. Block generators in round $t+1$ update MCM_t (representing state t ’s G_t) with the NTs in blocks transmitted from t_1 to t_0 to reflect state $t+1$ ’s G_{t+1} . Three

activities must be carried out in order to convert from MCM_t to MCM_{t+1} : As new Internet of Things (IoT) devices join a network or a new IP address of host interacts with an IoT device, a network's topology may undergo three types of changes: (1) a change in the weight of connections between vertices that interacted in round t ; (2) new vertices being added and edges; and (3) the removal of vertices and edges. Block generators provide dynamic community discovery using the mutual connections graph G . (DCD). Snapshots of permissioned BFT blockchain states may be used with AutoBotCatcher to do dynamic community discovery using the Louvain method. If the network is snapshotted at different times, Louvain may produce two identical community structures, causing the system to lose track of the communities and downgrade its performance [64]. At time $t+1$, AutoBotCatcher employs the communities discovered at time t to kick off the Louvain method of community discovery. AutoBotCatcher employs the CommSett from state t to do community detection on G_{t+1} in round t .

4.1.3 Identifying Botnet

Botnet status is determined by block generators during the botnet check process. To locate botnets, AutoBotCatcher uses a dual-pronged approach.

- 1) Bots communicate with one another in order to coordinate actions; as a result, they are more interconnected than communities of decent humans are [24].
- 2) "Crucial nodes" [52] are the multiple nodes that botmasters or assault targets communicate with.

Across all communities, AutoBotCatcher calculates the typical amount of interactions between any two IP addresses. For communities where the result is more than, suspicions of botnet activity are raised. When searching for key nodes, AutoBotCatcher analyzes potential botnet communities. All potential botnet IP addresses are entered into a matrix of contacts by AutoBotCatcher (MCM). The block generator flags a network as malicious if it contains key nodes indicative of a botnet. It is the responsibility of the block generators to allocate IP addresses after the communities hosting botnets have reached consensus on the new status for bot blacklisting. After a state change, the last duty falls on the shoulders of the block generators to maintain accurate bot blacklists of agents. It ensures that all IoT device blacklists are up to date by using smart contract transactions to add and remove IP addresses from the local blacklists maintained by agents.

4.2 PeerGrep

PeerGrep differentiates between activities carried out by botnets and those carried out by P2P operations that continue for a long period of time by using a characteristic called persistence and periodicity. This feature gives enhanced detection capabilities and a reduced false positive rate. The qualities of tenacity and periodicity are what bring about the successful completion of this mission's aim. They make it possible to track down P2P bots, which is a useful tool for finding P2P bots that

maintain consistent and regular network activity. In a similar vein, this approach is able to differentiate between two different P2P apps that are simultaneously operating on the same host. In order for this procedure to be successful in delivering the intended result, it is essential to combine behavioral analysis with fingerprint examination. PeerGrep, the alternative method that was presented, does not need access to seed bot data, which is famously difficult to acquire. PeerGrep is a solution that does not require seed bot data.

4.2.1 System Design

P2P bots display network traffic patterns similar to those of non-malicious P2P clients because they use P2P protocol to construct CC channels. PeerGrep uses a two-step approach to identify P2P bots. First, they recognize all P2P hosts in the traced network. This includes both Peer to Peer and non-Peer to Peer traffic that is completely innocuous and Peer to Peer traffic that is associated with botnets. In the second step, they identify all of the Peer to Peer botnet traffic that was produced by P2P bots that were hosted on the previously listed P2P domains. PeerGrep’s organizational structure is stated in Figure 2.1

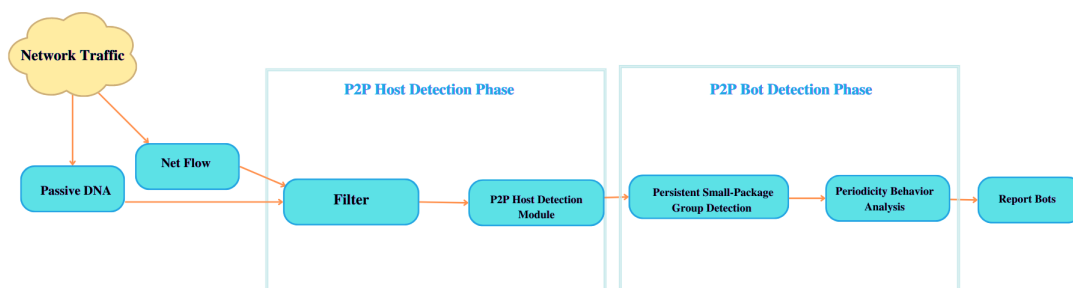


Figure 4.2: System Design

The traffic filter module processes data in its raw form, which includes P2P traffic as well as traffic from other sources. We use data from passive DNS in order to filter out traffic that isn’t peer-to-peer [20]. P2P clients will often seek for their IP addresses rather than their domain names in peer lists when they are conducting a search for a new peer. Passive DNS records and netflow flow records are both valid options for usage as input data. Dnet and DDNS both contributed data sets to the project. They begin by gathering passive DNS information from the network, and

from there they create the IPDNS IP range, which is made up of the IP addresses that the DNS server returned. Then, we will remove from Dnet any traffic whose destination IP addresses have been located via the use of IPDNS and filter out the remaining traffic. The remaining dataset, known as Dfilter, takes into account both peer-to-peer and non-peer traffic flows. It is possible to provide an algebraic representation of the filtered dataset Dfilter:

$$D_{filter} = \{flow | flow \in D_{net}, flow.dstIP/IPlist\} \quad (4.3)$$

The second part of the system is responsible for finding P2P hosts among the filtered flows. A number of packets are sent back and forth between each P2P peer [39]. Packets are created during the process of discovery, content requests, alarms, and data transfers [39]. All the packets, sizes, and protocols used in a given P2P flow originate from the same source. P2P clients in many networks often swap data packets with one another as part of normal network functioning. In P2P systems, the flow IPs are dispersed throughout several different networks. Utilizing Border Gateway Protocol (BGP) prefixes to categorize peer-to-peer communication. For each Dfilter flow, f , we calculate the host-related fluxes using the formula

$$v(f) = Spkt, Sbyte, Rpkt, Rbyte, Proto(f) \quad (4.4)$$

i th group of H is me . The vector-valued flows in $Gi(H)$ form a group. System $Gi(H)$. $v(Gi(H)) = Spkt, Sbyte, Rpkt, Rbyte, Proto$ is the feature vector for the group. Multiple network flows are created by P2P hosts. The host H 's flows are denoted by $G(H) = G1(H), G2(H)$, and so on (H). Time intervals of size T are used to split the flow groups on each host. The host H segments the flow groups throughout time, with each segment being designated by a j . The BGP prefix associated with each $Gi(H)j$. $bgp(Gi(H)) = bgp(ij)$. Thin slices with a low $bgpij$ are not accepted. $PGi(H)j$ with a focus on P2P. Cooperative Flow In the case of hydrogen, $Gi(H)$ equals $P Gi(H)1, PGi(H)2$, and $PGi(H)m$. $Gi P$ was wiped out (H). This formula states that $P G(H) = P G1(H), PG2(H), PGn(H)$. If $G(H)$ is a peer-to-peer host, then H is the guest. PeerGrep will be able to distinguish between bots and hosts.

For financial gain, botmasters use bots to do malicious acts. To prevent fragmentation into independent nodes, a P2P botnet requires a constant supply of online peers. P2P software allows users to decide when they want to go online. P2P file-sharing applications are designed to remain active until all downloads have been completed. You can tell malicious applications like P2P bots from harmless ones using active ratio, also known as $TP2P THost$. Applications like Skype and eMule may be programmed to launch automatically when Windows boots and remain active until the computer is shut down. Clients of P2P protocols may be spotted using this. eMule and BitTorrent need a lot of bandwidth, hence they have large packet sizes. P2P botnet usage is low. Small packets reduce network impact. P2P botnets must stay unnoticed and not download large files. We can find differences between large P2P packets, innocuous programs, and P2P botnets. Subsequently, we analyze the flow from each P2P host based on its timestamp tf . The coordinates ts and te define the beginning and end of H . To do this, we partition the time period T into k equal chunks called epochs (3 minutes). Where $v_i = 1$ if there is at least one flow in the i th epoch, and 0 otherwise, we get $Vact(H) = v1, v2, v3, vk$. The host's active time

may be estimated from the vector $V_{act}(H)$. We can estimate active V time using H' state. ($T_{host} = \sum_{i=1}^k v_i$) is the formula of THost. Attaining P2P-relevant groups may be used to make TP2P estimations. It's because, as long as the P2P program is running, it will regularly communicate with other peers by exchanging network control (e.g., ping/pong) messages [71]. We look at the collection of P2P-relevant groups $PG(H) = P G_1(H), PG_2(H), ,PG_n (H)$ for each host H that we previously categorized as P2P client. The same method used to determine THost is then used to determine the amount of time $PG_i(H)$ in $PG(H)$ is actively working, which is then given to the group as TP2P. Subsequently, they calculate an indicative ratio for the $PG_i(H)$ group as $r(P G_i(H)) = TP2P_i/ THost$ and $v(G_i(H))$ may estimate package size. They check out the PGs on each peer-to-peer client host to see if there is anything pertinent to the P2P protocol (H). We calculate the total bytes and packets sent for all groups in $PG(H)$, where I is the number of groups. Next, we get the average packet size by solving for $APS(PG_i(H)) = Sbyte/Spkt$. We have $r(PG_i(H))$ and $APS(PG_i(H))$ for each host H group $G_i(H)$. The groups that meet the following criteria are then retained, while those that do not are discarded. The average packet size threshold is denoted by size, whereas the active ratio criteria is denoted by act. Therefore, we consider the remaining groups to be PSGs, and we denote all PSGs on host H by the notation $PSG(H) = PSG_1(H), PSG_2(H)$. Identify P2P botnet-related and benign groups with extended active periods and tiny packet sizes. They're split. P2P botnet reduces network churn and maintains continuous communications to receive and execute commands. Peers of innocuous P2P apps are resource-hungry. P2P bots frequently contact the same hosts with tiny packets, whereas benign apps visit various addresses. This feature evaluates host connection durability and periodicity. H's little packaging groups survive. They convert $PSG_i(H)$ destination IPs to connection time series. $CT_iH(IP_j) = time_1, time_2, time_q$ denotes the time window (size = T) during which IP_i is linked. Calculate $CT_iH(IP_j)$. P2P bots contact the same hosts with tiny packets, stabilizing $CT_iH(IP_j)$ sequences. Thus, they compute the size and CV of each $CT_iH(IP_j)$, SiH_j , $CViH_j$ and keep those whose $SiH_j > S$ and $CViH_j CV$, where S and CV are size and CV criteria, respectively. P2P bots are the remaining hosts.

4.2.2 Evaluation

The Spark cluster [57] with one master and five slaves is used for the tests, Bgp assessment. If this quantity is low, non-P2P flows can be considered as P2P flows. P2P groups may be missed if it is too large. We independently test bgp 9-19. Figure 4.3 shows results. Set $bgp = 15$ and detect benign and dangerous P2P hosts, disregarding non-P2P sites.

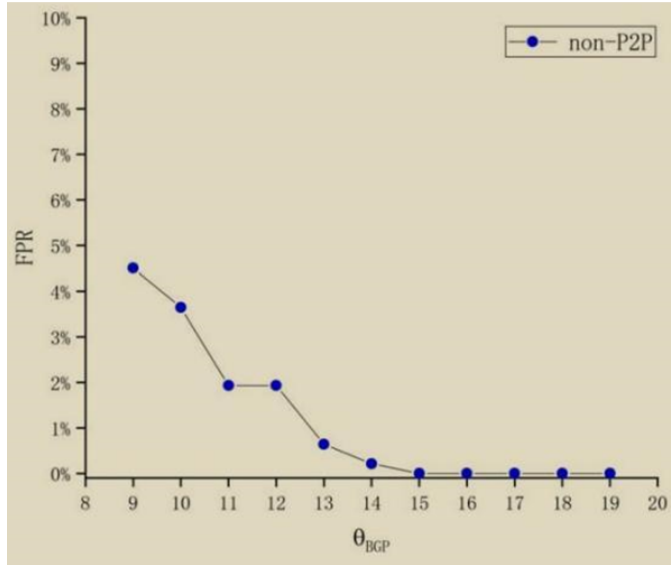


Figure 4.3: FPR of P2P host detection for different values of θ_{BGP} .

Now we evaluate the size and act of the detection thresholds for persistent small-package groups. If the act is really small, it may include non-persistent clusters. If the clusters are too large, they may be ignored. File-sharing and other peer-to-peer (P2P) software may generate large package clusters if their data sizes are excessive. Insufficient funding might lead to P2P botnet organizations being ignored. With a few exceptions, we evaluate the package size and set size = 100 of every P2P-relevant group. Once an assessment has been made, we go on to the act. From a value of 0.2 to 0.9, we test the act. Three illustrates the results. TPR = 100% and FPR = 8.46% are the results with act = 0.5. Possible long-term small-package groups generated by several P2P programs.

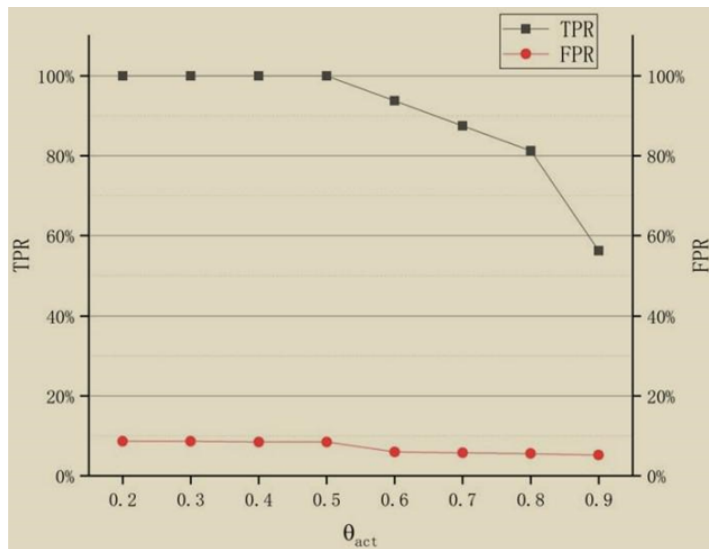


Figure 4.4: TPR and FPR of θ_{act} .

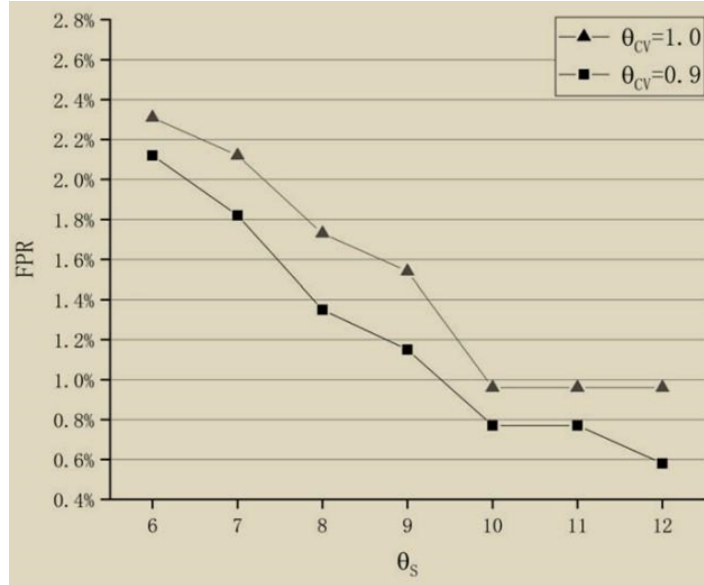


Figure 4.5: PR of θ_S and θ_{CV} .

For more reliable results, we need to exclude P2P communities that don't do any harm. S and CV are analyzed in periodicity behavior analysis. In order to preserve 100% TPR, we conducted the experiment again with S and CV values in the range of 6–12. The results are stated in Figure 4.4. On the other hand, where S=12 and CV=0.9, we find a TPR of 100% and an FPR of 0.58%. Low FPR is a feature shared by all P2P bots. All four infected dual-hosts correctly identify as P2P bots, while running a harmless P2P program. With such a high TPR, it's clear that PeerGrep's capacity to identify malicious P2P bots is unaffected by legitimate P2P software.

4.3 Honeypot

The (ATSRW) [23] [8] [22] consists of these two main parts: It's (1) a spearhead which already has the code and peer list. When a computer's security is compromised, the fault lies with this. The second (2) is the authorization key that may be used to add a freshly programmed bot to an existing botnet.

4.3.1 Proposed Method

Wang's [23] ATSRW authentication mechanism is incompatible with the suggested approach. Peer-to-peer botnet ATSRW. A hacked computer gets authenticated before joining a botnet. The authentication method requires verifying a bot's signed IP address with a spear phishing malware-generated IP address. Method uses each infected host's peer group. This peer list includes botnet-infected devices' IP addresses. Using the weakness that permitted infection, the technique provides remote peer host control. Taking over a hacked computer isn't meant to do damage, but to mislead the botnet's authentication mechanism. The honeypot might potentially disinfect the system after infection. After breaking down the procedure (1), we were able to- To begin with, A infects B with the spearhead code. B (2) will distribute malicious code to M unsuspecting hosts through the peer list it receives. Every hon-

eypot call goes straight to H. Using honeypots to suppress P is an effective strategy. The bogus IP addresses of M randomly selected hosts are generated by the honeypot H in (4). H connects via TCP to P using C's IP address. Fifthly, P will make an effort to connect using C's IP address. The answer of a TCP connection will have the correct sequence and acknowledgment number. (6) Because P is managed by the honeypot, we may reroute the response to H and provide the appropriate TCP sequence and acknowledgement number. H will deliver C's IP to B following the completion of the TCP 3-way handshake between H and C. P informs B of C's infection. As a result of step (7), B downloads the spearhead code M times. We can now confirm this. The components of our system are: This consists of (1) two honeypots, the first of which is running the OS that the bots like to use. Using P0f, Li et al. [15] were able to determine the bot's operating system. This software checks TCP SYN packets for OS versions. Table 2.1 shows the OS percentages. 92% of bots use Windows. Windows is a common and straightforward botnet target, therefore the first honeypot will focus on it. The second component (H) installs Linux, enabling the honeypot to mimic any of M hosts during the TCP 3-way handshake. (2) Traffic rerouting (such as PortMapper). Knowing the IP address of the target system and utilizing a vulnerability scanner like Nessus to generate a list of hosts and their vulnerabilities may help obtain access to a system. (3) The raw socket may be used to mimic the TCP 3-way handshake. The effectiveness of the counter mechanism is tested by comparing the spearhead code on honeypot B. P's signed IPs are compared to B's spearhead-generated IPs.

4.3.2 Analytic Results

Infected hosts eventually join ATSRW when they follow the main force. The botnet expands behind the worm's leading edge [60]. SpearT+M+AccessHT+MK TupleT+MK ReoprT. Time to compromise M hosts (AcessHT), Peer list transmission (PeerLT), tuple transmission (TupleT), report transmission (ReoprT), and spearhead transmission (SpearT). R-values. In our study, we choose a binomial distribution (invulnerable). The binomial criterion has been modified. No other tests were affected by the presence of vulnerable peers or hosts. Results from experiments (vulnerability). L has access to the network before P does (L). In the event where the host/peer time is L, the percentage to be used is P. (L). $K=P(L)$. About h% of people feel this way. $(n!/(M!(n-M)!)) hM(1-h)^{n-M}$. Human hosts are especially susceptible. $n=T/h$. It is assumed that hosts and peers will learn about each other at $[0,L]$. It takes $L/2$ time to find hosts or peers. Compromised hosts take a lot of time. $AccessHT=t+L/2M/h$. Honeypots might become part of a botnet that operates outside of ATSRW. Method execution is slow in Peerlist. SpearT+M(SpearT+PeerLT+ YAccessPT+ MY(TupleT+Spooft+ MY ReoprT). The Y is compromised by the group. AccessPT is peer compromise. Inertia leads to The Y variable lags. Infiltration of AccessPT by hackers. A quick handshake is important for Spooft.

To demonstrate the lag time of our approach, we'll use Zero Access's settings as an example P2P botnet. The time it takes to authenticate with ATSRW will then be measured against the time it takes to use our way to circumvent ATSRW. The longest possible lag time for a host joining a botnet using ATSRW is shown in Fig.

4.6. Each attempt to compromise a host takes 100 seconds, denoted by the delay parameter t . The time of this approach lowers as the magnitude of the susceptible probability rises, as illustrated in Fig. 4.7. To estimate the lag time of our strategy to circumvent ATSRW, we consider the best-case scenario, which is when honeypot becomes part of a botnet. There was no lag time throughout the spoofing operation, and the average time it took to compromise a peer for each experiment (t) was 720 seconds. Assuming there are 76 peers K available, the probability of discovering a susceptible peer $P(V)$ is quite close to 1 (calculated using (7)), yielding a success value of about 30% for the strategy. In Fig.4.8 we can see the total number of compromised peers. As can be shown in Fig. 4.9, the delay ($D1$) is greater than the absolute maximum delay of the ATSRW. As can be seen in figures 4.9, decreasing the latency may be achieved by restricting the number of accessible peers

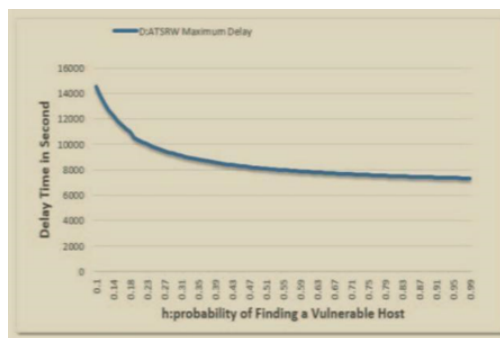


Figure 4.6: ATSRW

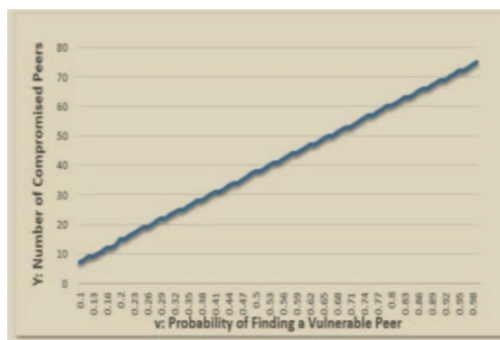


Figure 4.7: Number of Compromised Peer



Figure 4.8: Bypass STARW Delay.

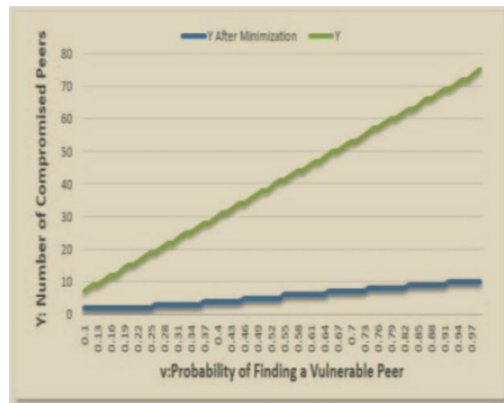


Figure 4.9: Y Values after Minimization

4.4 Software Defined Network

A software controller or application programming interface (API), which is in continual connection with the underlying hardware, governs and directs the flow of network traffic. This flow may be thought of as data moving across a network. SDN is the name given to this approach to networking. The nodes that make up the network are relieved of the duty of determining how data should be conveyed as a result of SDN’s separation of the control plane from the data plane. They will either transmit the packets on to their destination in line with the directions supplied by their controller, or they will discard them in accordance with the commands issued by their controller. OpenFlow [31] is an implementation of SDN southbound protocols that is extensively used and that manages the flow of information between controllers and switches. It is responsible for this movement of information.

4.4.1 SDN Based Detection

The architecture of the system’s network is shown in figure 4.8, which is a depiction of that design. The connection between the OpenFlow controller, which is represented as the Rule Arbitrator in Figure 4.10(1), and the OpenFlow switches, which

are displayed as the Data-link Bridges in Figure 4.8, is illustrated by the dashed lines. Figure 4.10(2) shows the Rule Arbitrator. When the programmable module is installed, the OpenFlow controller takes on the role of Rule Arbitrator and selects which protocol will be used to transmit data across Datalink Bridges. This occurs simultaneously with the installation of the module. When the programmable module is put into use, this occurrence takes place. If the Rule Arbitrator does not impose any limits on the data's forwarding, the Data-link Bridge will work as a conventional layer 2 switch. This is the only scenario in which this will occur. Figure 4.10(3) depicts how the Rule Arbitrator kicks off the process by issuing directives to the Datalink Bridges, which in turn transmit a copy of all incoming packets to the Detection Agent located in the surrounding area. For the purpose of flow identification, all packets are grouped together if they have the same 5-tuple, which consists of the same source IP address, source port number, destination IP address, destination port number, and protocol. This ensures that only one flow can be identified at a time. This takes happen in all case, despite the fact that the packets may be travelling in any direction. During the process of detection, it is the responsibility of the Detection Agent to carry out an activity that is often referred to as "flow identification." The classification of flow will be included into this technique as one of its components. It's feasible that if we use flow data to analyze a range of variables, we'll have a better grasp of how the network operates. Based on machine learning models that were constructed using data gathered from actual P2P botnet traffic as well as traffic from benign P2P applications, the Detection Agent determines whether incoming data flows are coming from a P2P botnet or a benign P2P application. Based on information gathered from actual P2P botnet traffic, this decision is taken. The Detection Agent will provide the Rule Arbitrator with the 5-tuple data and the type of P2P botnet or application that was found after the flow classification process is finished. Every time fresh information is received from the Detection Agent, the Rule Arbitrator updates the data flow tables in the Data-link Bridges. This ensures that the tables accurately represent any newly discovered information. This occurs each time a fresh piece of evidence is brought in for consideration. The main responsibility of a Data Connection Bridge is to get rid of any potentially harmful packets that have been classified as such by a separate component known as a classifier.

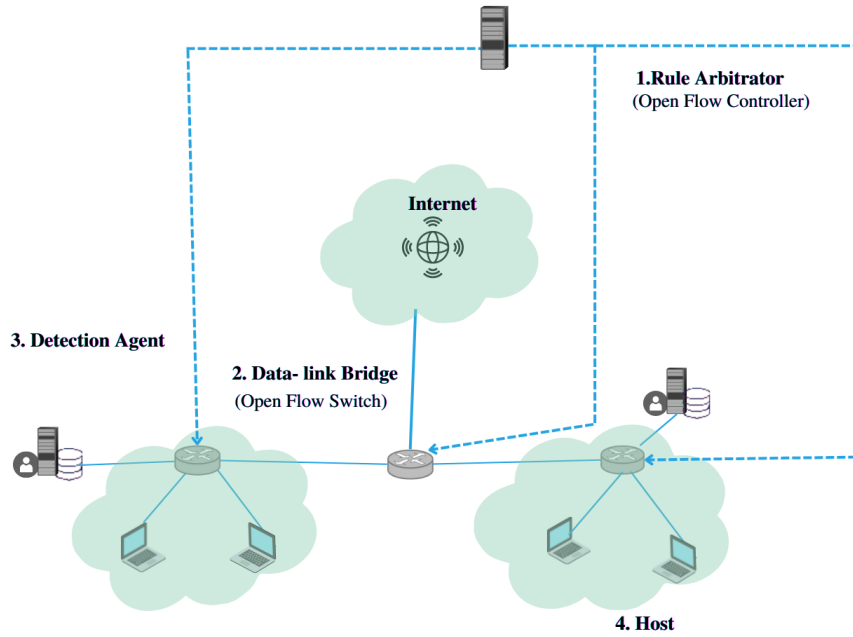


Figure 4.10: System overview

As is evident in Figure 4.11(a), this module utilizes the same 5-tuple information to classify inter-host network traffic, therefore combining mirrored packets into traffic flows. Once the packets have been converted into the traffic flow, the feature vectors may be developed and used in machine learning studies. This module captures network packets [34] [9]. The collection window must be provided (i.e., a flow duration). 600 seconds is the obvious winner after testing various flow lengths. At its heart, our design for this module (shown in Figure 4.11(b)) is based on the concept of categorization. The module collects feature vectors and then uses a machine learning algorithm to categorize the data. The following are the four steps of this procedure.

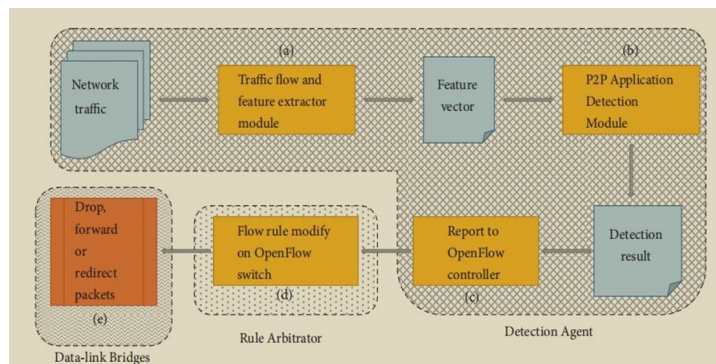


Figure 4.11: Flow diagram.

To be able to use a machine learning algorithm for categorizing P2P network traffic, appropriate training datasets and methods must first be found and created. This

is necessary in order to use the algorithm. We gather data on the overall amount of traffic produced by various P2P botnets and conventional P2P programs using the same techniques as detailed in PeerRush [44]. This allows us to estimate the severity of the issue this type of traffic poses. This larger module is composed of the Primary Architectural Module and the Secondary Classification Module. someone may think of both of these modules as constituent components of the whole. A binary classifier, shown in Figure 4.11(c), is used to carry out the process of doing the trace flow analysis. Since the Detection Method includes the binary classifier, this holds true. That allows us to examine feature maps that depict circulation patterns and share our findings with the world. Each of the five binary classifiers that we develop and make available to users is a part of the Primary Classification Module that we've created. We also use the whole data set, which includes many various types of remnants of network traffic, to create a multicast classifier as part of the Supplementary Classification Subsystem. As a result, we may construct the classifier in a manner that is both effective and efficient. Since this is the case, we may design the multicast classifier properly.

4.4.2 Network Traffic Analysis

Every packet delivered by the Data-link Bridge has a duplicate created on the port associated with the proper Detection Agent. The Detection Agent makes use of this duplicate copy. A Detection Agent submits a "registering" packet to the SDN controller before it begins operating, and this is how we establish a system that allows for communication between the two parties. This has to be finished before the Detection Agent enters its active phase. The purpose of this is to establish a connection between the two topics at hand. Due to the confidential nature of the data included in the registration packet, hosts cannot claim to be the Monitoring Agent. The Rule Arbitrator finds out the Detection Agent's MAC address, IP address, and data-link Gateway from the registration packet sent to it by the Recognition Agent. The Arbitrator for Rules receives this information package from the Detection Agent. The registration package serves as the only source of data. As seen in Figure 4.12, sequence diagrams may be quite helpful. In Figure 4.12(c), the Detection Agent provides a registration packet, which is collected after the Rule Arbitrator generates flow tables in Bridges for Data links(Figure 4.12(b)). The example may be seen in the center of Figure 4.12(a). If the Detection Agent's first flow table entries are hidden, as illustrated in Figure 4.12, it will be difficult for hostile hosts to create a copy of it. OpenFlow packet-in messages received by the Detection Agent shown in Figure 4.1(c) are sent to the Rule Arbitrator depicted in Figure 4.12(d) for processing.

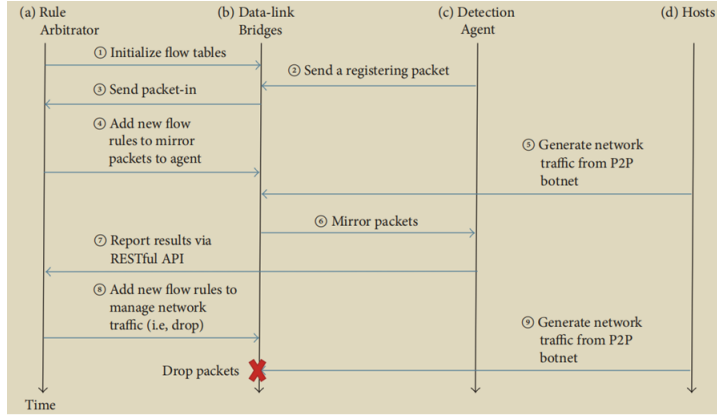


Figure 4.12: Sequence diagram of bot detection.

We have included a packet-in event handler inside the Rule Arbitrator’s module to get insight into the ingress port, the OpenFlow bridge ID, and other pertinent parameters. For example, the Rule Arbitrator may create a flow entry instructing the Data-link Bridge to reroute traffic from all ingress ports to the port belonging to the Detection Agent. This may happen if someone violates the rules. As depicted in Figure 4.12(f), the Detection Agent is responsible for keeping track of incoming packets via its network interface. The Detection Agent utilizes Netmate to record incoming packets, categorize them according to the 5-tuple information supplied in the packet header, and lastly organize those flows into feature vectors for analysis. If the results are unsettling, the Detection Agent will interact with the Rule Arbitrator via a Restful API. If a malicious packet matches the Detection Agent’s requirements, it will be deleted from the network after the Rule Arbitrator adjusts the flow tables of the Data-link Bridge. The purpose of adopting these procedures is to keep the Data-link Bridge safe from damage.

4.4.3 Evaluation

We test out several P2P botnet [44] and benign P2P app network trace files. We’d like to emphasize that the accuracy naturally results from the features vectors and learning techniques used. New learning algorithms provide an opportunity to significantly boost accuracy. We demonstrate our method’s effectiveness via assessment findings. We choose files with 24-hour time stamps from the traces of P2P networks. The data in these files is distinct from the one we used to train the classifiers. The information we utilized for analysis is summarized in Table 4.1. The reduced Zeus packet count is due to the fact that there is less Zeus traffic in the collected dataset. With the help of the Traffic Generator, we are able to re-create packets, which are then sent to the Detection Agent through Data-link Bridges. With the help of RESTful API, the Detection Agents examine the data streams in the network and communicate their findings to the Rule Arbitrator. The analysis’s findings inform the Rule Arbitrator’s assignment of unique flow rules to associated Data-link Bridges. Each and every traffic sample is tested separately. The detection efficiency is shown in Table 4.2. We combine network traffic trace data from P2P botnets and P2P apps to simulate a more realistic experiment. Once again, we are in a position to efficiently analyze and identify network traffic originating from a wide variety of

P2P botnets and benign P2P apps. A summary of the findings is provided in Table 4.1. The experimental findings suggest that our method can accurately identify P2P malicious packets compared to benign ones. Tables 4.1 and 4.2 show that the results are comparable. Despite the small sample size, there is a strong possibility of producing a reliable model from the Zeus training data. Thus, its precision is unparalleled. Both Storm and uTorrent have poorer accuracy than the rest of the pack. Due to the higher sample size provided by the test data, it is probable that more training data will result in a more accurate training model.

Class	Packets	Flows	Detection Accuracy
Storm	1747562	39281	92.824%
Zeus	21714	9098	98.299%
eMule	1104034	8905	95.046%
uTorrent	1067949	33087	92.752%
Skype	1177166	8121	95.024%

Table 4.1: Network traffic evaluation and detection accuracy.

Class	True positive rate	False positive rate
Storm	94.16%	1.41%
Zeus	98.46%	0.13%
eMule	96.04%	0.43%
uTorrent	91.75%	0.07%
Skype	93.47%	0.48%

Table 4.2: Real world evaluation result.

4.5 Resilient Neural Network

It may be said that the functions that are performed by the nodes of a neural network, which are linked to one another, are analogous to the activities that are performed by individual brain cells. These tasks may be accomplished via the use of the same methods by a neural network. They are able to accomplish this goal by using algorithms that are able to classify and cluster data, in addition to detecting correlations and patterns in that data that were not previously detected. Because of this, they are able to accomplish what they set out to achieve. In addition to

that, these algorithms are effective when it comes to categorizing and arranging the data that they are given to work with in the way that best suits their purposes.

4.5.1 Proposed Approach

The framework that is currently being provided is constructed with two basic principles serving as its foundation. Its only function is to monitor activity taking place on the network when it is still in its infancy. Second, it takes advantage of the fact that, due to the preprogrammed nature of Bots, during the propagation phase, Bots would exhibit frequent communication behaviors with their CC servers and peers in order to learn about new peers and acquire the most recent update of tasks. This is done in order for Bots to acquire the most recent information about tasks. This action is taken in order to get the most current information possible on the responsibilities that need to be fulfilled. In contrast to other types of malicious software, botnets work together in order to carry out coordinated assaults, and they depend significantly on a central hub in order to carry out the bulk of their communication. This is because botnets cooperate in order to carry out coordinated attacks. The Botmaster has access to a wide array of communication channels via which he may communicate with his Bots. For the identification of P2P Bots, it is recommended that a multilayer feed-forward neural network with a configurable

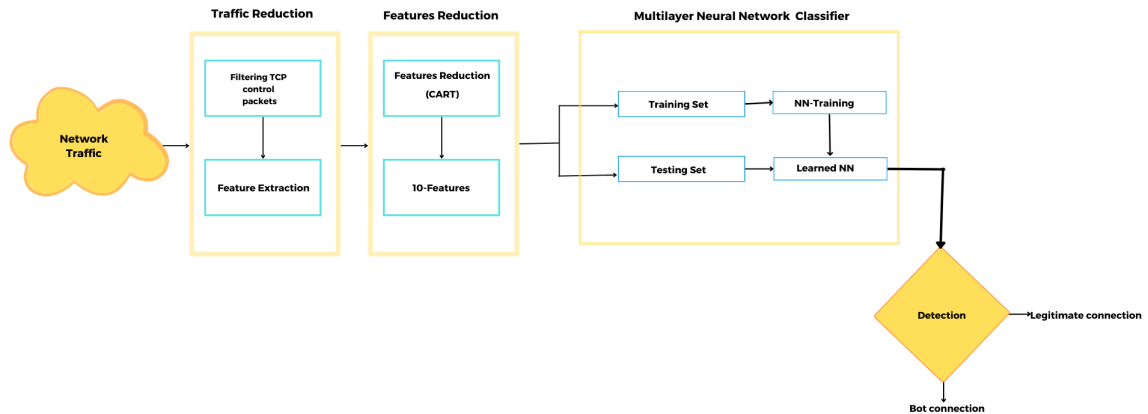


Figure 4.13: Block diagram of the proposed system

learning rate be used. This is because the vast majority of P2P Bots (such as Waledac Bot, Storm Bot, Conflicker, and Zeus Bot) communicate with one another using TCP connections [7]. During the course of our investigation, we developed a method that, by analyzing TCP control packets, allows us to infer information on TCP connections. Back-propagation is a method that can be used to speed up

the learning process. It is robust enough to survive changes that were made accidentally, thus it can be used to speed up the learning process. The most effective approach has been identified as resilient backpropagation, which has been shown to be superior in terms of convergence time, accuracy, resilience, and sensitivity to changes in the parameters of the training set. Take the graphical representation of the suggested system that is included in the document referred to as Figure 4.13, which itself includes the figure.

4.5.2 Performance evaluation

After analyzing the data, it is evident that the suggested method achieves the maximum detection and accuracy rate (about 99%) when using a neural network. The results of our tests show in Table 4.3 that the PCA-based feature set is the least effective in terms of accuracy and detection rates. Analyze the impact on the Bot detection system's FPR, PR, and F-measure of using a variety of feature selection approaches while maintaining the same TCP characteristics (Figure 4.14).

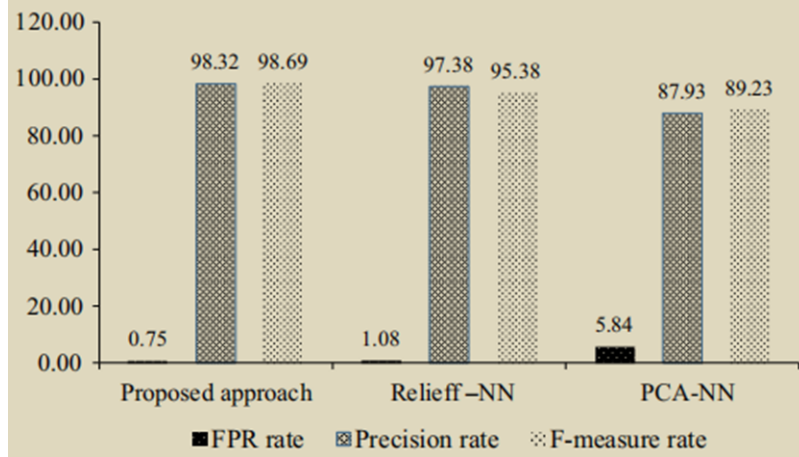


Figure 4.14: FPR, precision rate and F-measure comparison

Data analysis revealed that the suggested method’s F-measure averaged 98.32%, whereas the F-measure for PCA ranged from 87-93%. In addition, when compared to other approaches, the suggested strategy had the lowest false positive rate (0.75%). The receiver operating characteristic (ROC) curve depicting the compromise between false positive rates (FPR) and true positive rates (TPR) is provided to assess the performance of our suggested approach in identifying P2P Bots (TPR). Here, we have an FPR along the x-axis and a TPR along the y-axis. As can be shown in Figure 4.14, the area under the curve (AUC) for the neural network-based CART feature selection of TCP control packets is 0.994. Using TCP control packets, we categorize P2P Bot connection activity during a 30-second period to evaluate the efficacy of the suggested technique. Estimates of the processing times involved are used to evaluate the feature selection methods in terms of their practical usefulness. Furthermore, we present a numerical measure of the features selection algorithm’s output subset, expressed in terms of the time required to train a neural network. The research was conducted on an Intel i7-4.20 GHz Windows 11 computer. You can see a summary of the many methods you may choose attributes in Table 4.4 below.

Features Selection algorithm	CART	Relieff	PCA
Time (s)	2.5	11.3	4.9 height

Table 4.3: Features selection algorithm computation time

Neural network training time	CART features subset	Relieff features subset	PCA features subset
Time (s)	2.3	2.9	25

Table 4.4: Neural network training time with CART subset, Relieff subset and PCA

In order to choose a subset of features from the same dataset, the decision tree (CART) approach takes the least amount of time (see Table 4). In Table 5, we

can see how the training time required to develop an appropriate representation of the network varies depending on the characteristics utilized to construct the various neural network models. In contrast to the ReliefF subset and principal component analysis (PCA), the CART approach greatly increased the pace at which neural networks could be trained. However, the feature subset chosen for the ReliefF procedure determines the absolute maximum time involved.

Chapter 5

Analysis and Result

5.1 Analysis

There have been no exhaustive studies conducted so far on the best approaches for detecting P2P botnets. Nevertheless, a number of studies on the identification of botnets include a concise examination of the detection methodologies. In this part of the article, the surveys are broken down and studied so that we may illustrate how the various detection methods are categorized. A prior piece of research enumerated a variety of methods that can be used to identify P2P botnets. These practices are not examined in great detail by the investigation that is being conducted. In addition, the detection methods that were employed in past investigations are dissected one by one and both the advantages and disadvantages of employing these approaches are discussed here. The fundamental concepts, as well as the benefits and drawbacks, of these various methods are discussed. Methods for identifying P2P botnets have been studied based on their traffic characteristics. Detection techniques fall into the following broad categories: flow-based, machine learning, network behavior, traffic analysis, graph-based, and selection-based. The fundamental ideas behind these methods are broken down here. However, the talk did not focus on the pros and cons of these methods or the algorithms they employ.

5.1.1 Survey comparison:

The surveys show it's tough to classify detecting systems. Each poll highlighted distinct algorithm characteristics. Choosing a comparative criterion was difficult. Each survey's comparison technique was evaluated to identify its primary aspects. "Table 2.1" displays survey data. Summarize these dimensions:

- **Data sources:** Application logs, NetFlow logs, or network packets are examples of basic information.
- **Detection features:** The characteristics used to build the topology. According to these features encrypted botnet detection, botnet protocol identified, and rules required for detection.
- **Detection algorithms:** Algorithm helps to sort the task

- **Comparison algorithm:** compare between algorithms which one is more effective.
- **Advantages and disadvantages of algorithms:** the limitations and benefits of using the algorithms.

In addition, the survey classification uncovered limitations connected with the various detection methods that were applied: The fact that several polls use different wording makes the research more difficult to understand. No survey addresses the topic of data sources, as seen in "Table 6," whereas another survey just lists detection methods. This is because no survey addresses the topic of data sources. In most cases, documenting the processes takes precedence over evaluating them in any way shape or form. When comparing and evaluating the various methods, the vast majority of the research mentioned little more than a small number of sources.

We offer a comprehensive survey that consists of a detailed topology as well as a comparison to be able to execute an in-depth investigation of the detection methods in order to circumvent these constraints.

5.2 The Comparison Approach

Here, we provide a comprehensive analysis of the various detection proposals from various perspectives.

P2P botnet detection features are laid out in a topology map, which the detection ideas are organized within. The purpose of the map, which is depicted in "Table 4.4," is to shed light on the ways in which the detection proposals are distinct from one another. Researchers are able to rapidly find the publications that adopted each technique with the help of this map, which allows each manuscript to appear many times. In order to incorporate all of the relevant parts of the papers, the topology map is segmented into the following categories:

Analysis dimensions	This work	Related work				
		Peergreep	Autobotcatcher	SDN	Honeypot	CART
Detection methods'	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Advantages'	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Shortcomings'	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Detection 'algorithms	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Data sources'	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	partially	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Detection features'	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Partially
Comparison'	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	partially	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 5.1: Categories of comparing detection techniques

A) The detection algorithms are given below:

1. A technique that is based on neural networks is employed to develop supervised machine learning models, which is then utilized in order to identify P2P botnets. The approaches are evaluated with the use of machine learning algorithms such as decision trees [54], Bayesian networks [38], support vector machines [38], C4.5 decision tree classifiers [56], naive bayes [41], boosted decision trees [55], and flow correlation [55]. On the other hand, clustering divides the data into separate groups..
2. Individuals are gathered into subsets with similar characteristics by the clustering process. In contrast to classification, clustering can be taught even in the absence of a labeled training set [55]. Hierarchical clustering and BIRCH [51] are employed to categorize network traffic. This method is effective at identifying both malicious and legitimate hosts, and it does so without the requirement for signatures to be embedded inside the payloads themselves.
3. The machine learning algorithm attempts to recognize Bots by comparing incoming and outgoing packets, new outbound connections, and host application launches. [54] Regression trees (cart), k-nearest neighbors (knn), random forests (rf), and k-means clustering (kmc) are only a few examples of methods that can be employed for this purpose.
4. The back-propagation algorithm's resilient back-propagation method is often regarded as the most successful method with regards to convergence speed, accuracy, resilience, and training parameters. Comparisons are made to established methods like principal component analysis (PCA) and relief function (ReliefF) to see how well the suggested method performs. Many metrics, such as the false positive rate, the true positive rate, the accuracy, the precision, the recall, and the F-measure, can be used to assess the performance of a neural network identification system. [51].

B) Detection methods are defined by detection procedures:

Between-node fluxes in a network are what flow-based approaches focus on.

Methods based on monitoring attempt to recreate, over time, a P2P network's number of users sharing a given resource.

1. Feature set selection techniques are used to analyze the input/output flow.
2. Algorithms based on conversations may reveal how stealthy a P2P botnet is.
3. Approaches based on signature analysis probe data packets sent across a network
4. Graph-based methods find points of connection by comparing locations.

Comparison between algorithms:

SDN vs CART for(Random forest algorithm):

SDN uses traffic routing to separate the control and data plane. Also, it conducts new bandwidth and defines traffic failover. Additionally, Random Forest System is a categorizing technique based on decision trees. Random Forest Algorithm fits the challenge since it isolates confusing occurrences. Random Forest, like CART, allows variable selection, missing data management, nonlinear correlation finding, and variable interaction detection. Random Forest provides superior prediction ability and precision compared to a single CART model. So, in terms of Random forest algorithm SDN is better than CART.

SDN vs CART for (Machine learning Neural Network)

SDN separates control and data. Data plane contains network devices and an SDN controller. Network devices connect to the controller via OpenFlow. They employ flow tables instead of processing packets directly. Managers handle each device independently in traditional networks. Controller modules manage SDN devices automatically. Developers may build SDN modules to control data plane packet flows. Traditional security risks may increase in SDN if not controlled properly. Here cart is best as it takes less time , can detect a large amount of data, It doesn't have to examine all of the network traffic to accomplish signature matching, and it doesn't need deep packet inspection (DPI). It can recognize bots linked to any host, port, or IP address. It has been demonstrated that CART outperforms the ReliefF subset and PCA in terms of training times for neural networks. Maximum time depends on ReliefF's subset of features. The proposed solution eliminates 60% of input packets while maintaining high detection and low false positive rates.

Approaches	FPR %	Accuracy %
Fedynyshyn et al.	7.8	92.9
Wen-Hwa et al.	0	98
Proposed approach	0.75	99.20

Table 5.1: Comparison with other approaches

AutoBotCatcher vs Cart for (Decision tree Regression tree)

To identify and destroy botnets, AutoBotCatcher performs automatic network analysis of Internet of Things (IoT) devices. With a permissioned Byzantine Fault Tolerant (BFT) blockchain[51], AutoBotCatcher gathers and analyzes network traffic flows from IoT devices as blockchain transactions, allowing for collaborative and dynamic botnet detection. AutoBotCatcher is better than the slow and laborious Cart in this regard. If you compare AutoBotCatcher to a cart, you'll realize that it's clearly better.

PeerGrep vs AutoBotCatcher(hierarchical greedy algorithm clustering algorithm)

Both peergrep and autobotcatcher concentrate on cooperative bot detection. It's called "community detection." Based on our argument, we've determined that autobotcatcher is better in every way. But is it more accurate than peergrep at detecting botnet communities?

To increase modularity for community finding, AutoBotCatcher employs the Louvain method **39**, a greedy approach. The Louvain technique suits AutoBotCatcher because of its speed, adaptability, and ability to swiftly analyze large networks. AutoBotCatcher uses snapshots of the permissioned BFT blockchain's mutual contacts graphs to facilitate real-time community discovery using the Louvain method. This paves the way for the identification of hidden neighborhoods.. Louvain approach may build two comparable community structures if there are even tiny variations between two subsequent network snapshots, as illustrated in. AutoBotCatcher would lose track of the communities, reducing its performance.

Where in PeerGrep, Clustering data means comparable groupings or clusters. Clustering is preferable to classification since it doesn't need a labeled dataset.

The bot detection module uses clustering to group suspicious hosts with similar characteristics. Finally, we'll use a threshold to choose only one bot. P2P botnet detection solution uses RNNs and CNNs (CNN). Experiments showed RNN had 98.7BotScoop has limitations. Since the system has a set time window, increasing it will raise false positives and processing costs.

Finally, classification algorithms provide crucial network traffic flow components to discriminate between conventional and botnet traffic flows. DT, Naive Bayesian, and Support Vector Machines are used to assess whether network data is from a botnet. The authors say their method can detect encrypted botnets. DT, NB, and SVM had an accuracy of 95.86

Peergrep enables more accurate community discovery in less time than the other two methods.

5.3 Result

Having compared the peergrep system to the ML algorithm, the neural network, the regression and decision tree, and the clustering algorithm, we have determined that the peergrep system provides the most accurate detection of botnets in the shortest amount of time p. This was found by contrasting different ML algorithms.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Peer to Peer file carve up was instigated for the general public in 1999. Subsequently, a large number of cyberattacks in numerous sectors begin to grieve the entire world. The primary objective of this paper is comparing the most contemporary Peer-to-Peer botnet attack detection approach and determine the one that is most effective. In this, we analyze systems like PeerGrep, AutoBotCatcher, and HoneyPot. After conducting a thorough investigation, we determined that PeerGrep is now the best effective method for identifying Peer to Peer botnets. Since other approaches occasionally fail to identify the most recent peer-to-peer botnet and have other drawbacks including file size, file type, speed, adaptability and time requirements. Nevertheless, PeerGrep constructs the successor of the most recent and significant method, such as the community detection, in an effort to prevent issues with file size, format, FPR, and other factors. The PeerGrep approach is currently the most effective tool for detecting peer-to-peer botnets.

6.2 Future Work

We can use more modern techniques to construct the peer-to-peer botnet method and evaluate it using more parameters. Numerous changes, testing, and experiments have been put off due to a lack of time. For instance, real-world data-based analysis sporadically takes weeks or even months to complete a single run. The focus of future work will be on deeper investigation of certain mechanisms, novel ideas to test out novel methodologies, or simply plain curiosity. Additionally, we intend to add techniques that compile maneuver statistical network traffic aspects to the current botnet identification model.

Bibliography

- [1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” *ACM SIGCOMM computer communication review*, vol. 31, no. 4, pp. 149–160, 2001.
- [2] D. Tax, “One-class classification; concept-learning in the absence of counterexamples. ph. d thesis. delft university of technology, asci dissertation series, 2001. 146 p,” 2001.
- [3] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the xor metric,” in *International Workshop on Peer-to-Peer Systems*, Springer, 2002, pp. 53–65.
- [4] I. H. Witten, V. Profile, E. Frank, and O. M. A. Metrics, *Data mining: Practical machine learning tools and techniques with java implementations: Acm sigmod record: Vol 31, no 1*, Mar. 2002. [Online]. Available: <https://dl.acm.org/doi/abs/10.1145/507338.507355>.
- [5] D. M. Kienzle and M. C. Elder, “Recent worms: A survey and trends,” in *Proceedings of the 2003 ACM workshop on Rapid Malcode*, 2003, pp. 1–10.
- [6] F. C. Freiling, T. Holz, and G. Wicherski, “Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks,” in *European Symposium on Research in Computer Security*, Springer, 2005, pp. 319–335.
- [7] P. Barford, “An inside look at botnets, special workshop on malware detection,” *Advances in Information Security*, 2006.
- [8] C. Zou and R. Cunningham, “Honey-pot-aware advanced botnet construction and maintenance,” Feb. 2006, pp. 199–208, ISBN: 0-7695-2607-1. DOI: 10.1109/DSN.2006.38.
- [9] R. Alshammari and A. Nur Zincir-Heywood, “A flow based approach for ssh traffic detection,” in *2007 IEEE International Conference on Systems, Man and Cybernetics*, 2007, pp. 296–301. DOI: 10.1109/ICSMC.2007.4414006.
- [10] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, “Peer-to-peer botnets: Overview and case study,” *HotBots*, vol. 7, no. 2007, 2007.
- [11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, P10008, 2008.
- [12] C. R. Davis, J. M. Fernandez, S. Neville, and J. McHugh, “Sybil attacks as a mitigation strategy against the storm botnet,” in *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, IEEE, 2008, pp. 32–40.

- [13] G. Gu, R. Perdisci, J. Zhang, and W. Lee, “Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection,” 2008.
- [14] G. Guofoei, R. Perdisci, J. Zhang, and W. Lee, “Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection,” 2008.
- [15] Z. Li, A. Goyal, and Y. Chen, “Honeynet-based botnet scan traffic analysis,” in *Botnet Detection: Countering the Largest Security Threat*, W. Lee, C. Wang, and D. Dagon, Eds. Boston, MA: Springer US, 2008, pp. 25–44, ISBN: 978-0-387-68768-1. DOI: 10.1007/978-0-387-68768-1_2. [Online]. Available: https://doi.org/10.1007/978-0-387-68768-1_2.
- [16] B. Smith, “A storm (worm) is brewing,” *Computer*, vol. 41, no. 2, pp. 20–22, 2008.
- [17] P. Wang, S. Sparks, and C. C. Zou, “An advanced hybrid peer-to-peer botnet,” *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 2, pp. 113–127, 2008.
- [18] M. Feily, A. Shahrestani, and S. Ramadass, “A survey of botnet and botnet detection,” in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, IEEE, 2009, pp. 268–273.
- [19] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X.-y. Zhou, and X. Wang, “Effective and efficient malware detection at the end host.,” in *USENIX security symposium*, vol. 4, 2009, pp. 351–366.
- [20] A. Manasrah, A. Hasan, O. Abouabdalla, and S. Ramadass, “Detecting botnet activities based on abnormal dns traffic,” *International Journal of Computer Science and Information Security*, vol. 6, Nov. 2009.
- [21] T. Aynaud and J.-L. Guillaume, “Static community detection algorithms for evolving networks,” in *8th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks*, IEEE, 2010, pp. 513–519.
- [22] P. Wang, “Study of the honeypot-aware peer-to-peer botnet and its feasibility,” 2010.
- [23] P. Wang, L. Wu, R. Cunningham, and C. C. Zou, “Honeypot detection in advanced botnet attacks,” *Int. J. Inf. Comput. Secur.*, vol. 4, no. 1, pp. 30–51, Feb. 2010, ISSN: 1744-1765. DOI: 10.1504/IJICS.2010.031858. [Online]. Available: <https://doi.org/10.1504/IJICS.2010.031858>.
- [24] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari, and M. Zamani, “A taxonomy of botnet detection techniques,” in *2010 3rd International Conference on Computer Science and Information Technology*, vol. 2, 2010, pp. 158–162. DOI: 10.1109/ICCSIT.2010.5563555.
- [25] G. Fedynyshyn, M. C. Chuah, and G. Tan, “Detection and classification of different botnet c&c channels,” in *International Conference on Autonomic and Trusted Computing*, Springer, 2011, pp. 228–242.
- [26] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, “Detecting p2p botnets through network behavior analysis and machine learning,” in *2011 Ninth annual international conference on privacy, security and trust*, IEEE, 2011, pp. 174–180.

- [27] L. Xiao-nan, L. Yang, and Z. Hua, "Peer-to-peer botnets: Analysis and defense," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, IEEE, 2011, pp. 140–143.
- [28] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, and X. Luo, "Detecting stealthy p2p botnets using statistical traffic fingerprints," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, IEEE, 2011, pp. 121–132.
- [29] K.-S. Han and E. G. Im, "A survey on p2p botnet detection," in *Proceedings of the International Conference on IT Convergence and Security 2011*, Springer, 2012, pp. 589–593.
- [30] Y. Luo, "Efficiency study of sybil attack on p2p botnets," Ph.D. dissertation, Concordia University, 2012.
- [31] ONF, *OpenFlow switch specification (version 1.3.0 - wire protocol 0x04)*, Jun. 2012. [Online]. Available: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf>.
- [32] D. Zhao and I. Traore, "P2p botnet detection through malicious fast flux network identification," in *2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, IEEE, 2012, pp. 170–175.
- [33] C. Costarella, S. Chung, B. Endicott-Popovsky, and D. Dittrich, "Hardening honeynets against honeypot-aware botnet attacks," *University of Washington, US*, vol. 394, 2013.
- [34] S. C. Guntuku, P. Narang, and C. Hota, "Real-time peer-to-peer botnet detection framework based on bayesian regularized neural network," Jul. 2013.
- [35] H. Hang, X. Wei, M. Faloutsos, and T. Eliassi-Rad, "Entelecheia: Detecting p2p botnets in their waiting stage," in *2013 IFIP Networking Conference*, IEEE, 2013, pp. 1–9.
- [36] L. Li, S. Mathur, and B. Coskun, "Gangs of the internet: Towards automatic discovery of peer-to-peer communities," in *2013 IEEE Conference on Communications and Network Security (CNS)*, IEEE, 2013, pp. 64–72.
- [37] J. Zhang, R. Perdisci, W. Lee, X. Luo, and U. Sarfraz, "Building a scalable system for stealthy p2p-botnet detection," *IEEE transactions on information forensics and security*, vol. 9, no. 1, pp. 27–38, 2013.
- [38] T. Contributor, *What is peer-to-peer botnet (p2p botnet)? - definition from whatis.com*, May 2014. [Online]. Available: <https://www.techtarget.com/whatis/definition/peer-to-peer-botnet-P2P-botnet>.
- [39] J. He, Y. Yang, X. Wang, C. Tang, and Y. Zeng, "Peerdigger: Digging stealthy p2p hosts through traffic analysis in real-time," in *2014 IEEE 17th International Conference on Computational Science and Engineering*, 2014, pp. 1528–1535. DOI: 10.1109/CSE.2014.283.
- [40] P. Narang, S. Ray, C. Hota, and V. Venkatakrishnan, "Peershark: Detecting peer-to-peer botnets by tracking conversations," in *2014 IEEE Security and Privacy Workshops*, IEEE, 2014, pp. 108–115.

- [41] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, “Peerrush: Mining for unwanted p2p traffic,” *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 194–208, 2014.
- [42] —, “Peerrush: Mining for unwanted p2p traffic,” *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 194–208, 2014.
- [43] B. Soniya and M. Wilscy, “Fuzzy inference system based on entropy of traffic for bot detection on an endpoint host,” in *2014 International Conference on Data Science & Engineering (ICDSE)*, IEEE, 2014, pp. 112–117.
- [44] M. M. Al-Hakbani and M. H. Dahshan, “Avoiding honeypot detection in peer-to-peer botnets,” in *2015 IEEE International Conference on Engineering and Technology (ICETECH)*, IEEE, 2015, pp. 1–7.
- [45] Q. Yan, Y. Zheng, T. Jiang, W. Lou, and Y. T. Hou, “Peerclean: Unveiling peer-to-peer botnets through dynamic group behavior analysis,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*, IEEE, 2015, pp. 316–324.
- [46] P. Narang, C. Hota, and H. T. Sencar, “Noise-resistant mechanisms for the detection of stealthy peer-to-peer botnets,” *Computer Communications*, vol. 96, pp. 29–42, 2016.
- [47] M. Thangapandiyan and P. R. Anand, “An efficient botnet detection system for p2p botnet,” in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, IEEE, 2016, pp. 1217–1221.
- [48] K. Gai, M. Qiu, Z. Ming, H. Zhao, and L. Qiu, “Spoofing-jamming attack strategy using optimal power distributions in wireless smart grid networks,” *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2431–2439, 2017.
- [49] C. Koliass, G. Kambourakis, A. Stavrou, and J. Voas, “Ddos in the iot: Mirai and other botnets,” *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
- [50] D. Zhuang and J. M. Chang, “Peerhunter: Detecting peer-to-peer botnets through community behavior analysis,” in *2017 IEEE Conference on Dependable and Secure Computing*, IEEE, 2017, pp. 493–500.
- [51] M. Alauthaman, N. Aslam, L. Zhang, R. Alasem, and M. A. Hossain, “A p2p botnet detection scheme based on decision tree and adaptive multilayer neural networks,” *Neural Computing and Applications*, vol. 29, no. 11, pp. 991–1004, 2018.
- [52] M. Gadelrab, M. Elsheikh, M. Ghoneim, and M. Rashwan, “Botcap: Machine learning approach for botnet detection based on statistical features,” *International Journal of Communication Networks and Information Security*, vol. 10, pp. 563–579, Dec. 2018. DOI: 10.17762/ijcnis.v10i3.3624.
- [53] S. T. S. Su Y. Chen and Y. Lin, “Detecting p2p botnet in software defined networks,” *Security and Communication Networks*, 2018.
- [54] G. Sagirlar, B. Carminati, and E. Ferrari, “Autobotcatcher: Blockchain-based p2p botnet detection for the internet of things,” in *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, IEEE, 2018, pp. 1–8.

- [55] P. Wang, F. Wang, F. Lin, and Z. Cao, “Identifying peer-to-peer botnets through periodicity behavior analysis,” in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, IEEE, 2018, pp. 283–288.
- [56] T. Yu, L. Rui, and X. Qiu, “Sdn defender: A comprehensive ddos defense mechanism using hybrid approaches over software defined networking,” *Security and Communication Networks*, vol. 2021, 2021.
- [57] M. Siwach and S. Mann, “Anomaly detection for web log data analysis: A review,” *Journal of Algebraic Statistics*, vol. 13, pp. 129–148, May 2022.
- [58] A. D. Fisher and D. Fisher, *Waledac botnet now completely crippled, experts say*. [Online]. Available: <https://threatpost.com/waledac-botnet-now-completely-crippled-experts-say-031610/73694/>.
- [59] *Spam surge trojan.peacomm threat level increased: Cscb national office*. [Online]. Available: <https://cscb.ca/article/spam-surge-trojanpeacomm-threat-level-increased>.

