FAST FLUX BOTNET DETECTION BASED ON ADAPTIVE DYNAMIC EVOLVING SPIKING NEURAL NETWORK

Ahmad Al Nawasrah

Thesis Submitted in Partial Fulfilment of the Requirements of the Degree of Doctor of Philosophy

School of Computing, Science and Engineering

University of Salford, Salford, UK

2018

Supervisor:

Prof. Farid Meziane

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

First and foremost, I would like to express deep gratitude to my supervisors Prof. Farid Meziane and Dr.Ammar Almomani for their helpful contributions, encouragement, and guidance through my period of study.

Also, I would like to thank my brother Noun Alnawasreh for his support in every part of my life.

Finally, I wish to thank my parents, my wife Oroba, my siblings, and my dearest children Mera, Ameer, and Layan for their support.

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university, or institute of learning.

ABBREVIATIONS

A	The resolved record information returned from DNS
ANN	Artificial Neural Network
Auth_NS	Authoritative Name Server
AOR	Average Online Rate
C2	Command and Control
CDN	Content distributed network
DDoS	Distributed Denial-of-Service
DENFIS	Dynamic Evolving Neural Fuzzy Inference System
DeSNN	Dynamic evolving Spiking Neural Network
DNS	Domain name system/domain name server
DoS	Denial-of-Service
DyNFIS	Dynamic Neural Fuzzy Inference System
ECoS	Evolving Connectionist System
eSNN	evolving Spiking Neural Network
FF	Fast Flux
FFDD	Fast Flux Domain Detector
FFN	Fast Flux Network
FFM	Fast Flux Monitor
FFSN	Fast Flux Service Network

FN	False Negative
FP	False Positive
FQDN	Fully Qualified Domain Name
GRADE	Genetic ReAl-time approach for FFSN DEtection
HTTP	Hypertext Transport Protocol
ICCAN	Internet Corporation for Assigned Names and Numbers
IP	Internet Protocol
KFFS	Killer Fast Flux System
LSGD	Localized Spatial Geo-location Detection
MAR	Minimum Availability Rate
NS	Name Server
RDNS	Recursive Domain Name Server
RR	Resource Record
RRDNS	Round Robin DNS
SLD	Second Level Domain
SNN	Spiking Neural Network
SSFD	Spatial Snapshot Fast flux Detection
SVM	Super Vector Machine
SYN	Synchronous Packet

TLD	Top Level Domain
TN	True Negative
TP	True Positive
TTL	Time To Live

LIST OF PUBLICATIONS

A. ALnawasrah, A. Almomani, F. Meziane, M. Alauthman. Fast Flux Botnet Detection Based on Adaptive Dynamic Evolving Spiking Neural Network Algorithm, The 9th International Conference on Information and Communication Systems, (ICICT 2018), IEEE.

ABSTRACT

A botnet, a set of compromised machines controlled distantly by an attacker, is the basis of numerous security threats around the world. Command and Control (C&C) servers are the backbone of botnet communications, where the bots and botmaster send reports and attack orders to each other, respectively. Botnets are also categorised according to their C&C protocols. A Domain Name System (DNS) method known as Fast-Flux Service Network (FFSN) is a special type of botnet that has been engaged by bot herders to cover malicious botnet activities, and increase the lifetime of malicious servers by quickly changing the IP addresses of the domain name over time. Although several methods have been suggested for detecting FFSNs domains, nevertheless they have low detection accuracy especially with zero-day domain, quite a long detection time, and consume high memory storage. In this research we propose a new system called Fast Flux Killer System (FFKA) that has the ability to detect "zero-day" FF-Domains in online mode with an implementation constructed on Adaptive Dynamic evolving Spiking Neural Network (ADeSNN) and in an offline mode to enhance the classification process which is a novelty in this field. The adaptation includes the initial weight, testing criteria, parameters customization, and parameters adjustment. The proposed system is expected to detect fast flux domains in online mode with high detection accuracy and low false positive and false negative rates respectively. It is also expected to have a high level of performance and the proposed system is designed to work for a lifetime with low memory usage. Three public datasets are exploited in the experiments to show the effects of the adaptive ADeSNN algorithm, two of them conducted on the ADeSNN algorithm itself and the last one on the process of detecting fast flux domains. The experiments showed an improved accuracy when using the proposed adaptive ADeSNN over the original algorithm. It also achieved a high detection accuracy in detecting zero-day fast flux domains that was about (99.54%) in an online mode, when using the public fast flux dataset. Finally, the improvements made to the performance of the adaptive algorithm are confirmed by the experiments.

KEYWORDS

Fast-Flux, Zero-day domain, dynamic evolving spiking neural network, botnet detection.

CHAPTER ONE

INTRODUCTION

Chapter Overview

This chapter provides the definitions of the area of research, the gap in the knowledge, the research aim and objectives, the research methodology, the research motivation, the scope of the research, and the structure of the thesis.

1.1 Introduction

Botnets are networks of compromised computers that are controlled remotely by attackers and are the basis of numerous security threats, such as distributed denial-of-service (DDoS) attacks, identity theft, phishing, and spam (Almomani, Obeidat, Alsaedi, Obaida, & Al-Betar, 2015; Almomani, Wan, et al., 2013; Barford & Yegneswaran, 2007; Dagon, Gu, & Lee, 2008; Fabian & Terzis, 2007; Grizzard, Sharma, Nunnery, Kang, & Dagon, 2007; Gu, Perdisci, Zhang, & Lee, 2008; Karasaridis, Rexroad, & Hoeflin, 2007; Levy & Arce, 2006; Rajab, Zarfoss, Monrose, & Terzis, 2006). Fast flux networks (FFNs) are a special type of botnet being used by criminals in the same manner as those used in round robin domain name systems (RRDNSs) and content distribution networks (CDNs) to offer high availability and flexibility for their malicious websites (Alieyan, Almomani, Manasrah, & Kadhum, 2015). Botnet writers disguise their malicious activities and design new tactics and mechanisms to hide their communications. One such a method is the IP fast flux, which is a mechanism that frequently changes IP addresses corresponding to a unique domain name. Another method is the domain flux, which is a mechanism that automatically and periodically generates domain names related to a URL of a command and control (C&C) server. The core idea of FFNs is to use bot computers as proxies (flux agents) that forward user queries to the backend servers called "motherships." A recurrent and fast change in the IP addresses of proxies is essential to evade detection and a potential shut down and to ensure high availability to those backend servers.

FFNs are considered to be a new development in the operation and management of spam campaigns. Along with campaigns, spammers send thousands of emails that contain interesting advertisements of products or services (e.g., pharmaceutical, adult content, and phishing) to users' email inboxes (Al-Duwairi & Al-Hammouri, 2014). These advertisements generally contain hyperlinks to malicious websites for the campaigns. Until recently, only a single static IP address is related to a website for a certain period of time; such a characteristic provides the security defenders the chance to take down that website. According to FFNs, a domain name of a malicious website points to more than one IP address (FF-agents), which is frequently and rapidly changing.

According to Kalige, Burkey, and Director (2012), HTTP botnets are considered dangerous because they attack and exploit systems. Current HTTP botnets use the strongest techniques to perform attacks. An example is the Asprox botnet, which has affected about 3.5 billions computers in the United States. The Asprox botnet uses an advanced double fast flux, called the hydra fast flux, as its main technique (Al-Bataineh & White, 2012). This technique renders efforts to take down and defeat C&C servers useless. Additional details are presented in Subsection 2.3.

The report of the Cost of Cyber-Crime Study (Enterprise, 2015) points out that the mean annualized cost of cyber-attacks for 252 benchmarked organizations is $7.7 million/year. The report also shows that these attacks are carried out with or supported by either a botnet or a web-based attack, and fast flux is used as an evasion technique to provide availability and resilience.

The report mentions that the most dangerous cyber-crimes are those caused by denial of services (DoS) and web-based attacks. The fast flux evasion technique has been widely used in botnets and web-based botnets to carry out DoS and others attacks (e.g., phishing and spam), with fast flux serving as the backbone C&C communication between the compromised computers and the mothership/malicious website.

Cyber-criminals have stolen around $78 million through various means using financial malware (Marcus, 2012). In addition, McAfee stated that previous fraud cases in Eastern Europe could be attributed to Zeus and SpyEye activities; after tracing some of these attacks, they found a highly complex fast flux botnet, as well as hidden compromised servers supporting the

website's long life (Marcus, 2012). Botnets are also responsible for spam e-mails. Spammers earn an average annual income of $50,000 to $100,000 (Su & Tsai, 2012). Fake online pharmacies are one of the many illegal activities available on the Internet; such activities are notorious for selling fake or inefficient medications and are involved in identity theft cases (Spamwiki, Online). A report from the Fortinet Global Cyber Security Research Team states that the fast flux technique has been used in fake Canadian online pharmacies to avoid detection (Pharmacy, Online). Security researchers have recently reported that a new variation of the "Gameover ZeuS" botnet makes use of the fast flux technique to protect its C&C servers (Inc, Online).

One of the core problems in botnet detection is the so-called unknown "zero-day" fast flux domain. Zero-day domains are defined as those related to bots (FF-agents) that are not blacklisted (Lin, Lin, & Chiang, 2013). A fast flux attack is a complex evasive technique that cannot be identified by many current techniques because attackers can use new and previously unseen bots. A number of potential solutions to fast flux botnet attacks have been proposed, but these solutions are not yet effective. These solutions range from passive, to active, to real-time approaches. The misclassification of malicious and legitimate domains increases with time, especially when dealing with unknown zero-day fast flux botnet domains. The proposed approach exploited the adaptive DeSNN to detect these zero-day fast flux domains, experiments are conducted to compare and show the improvement of adaptation made on the DeSNN algorithm on the performance of the algorithm itself, and on the process of detecting FF domains. Furthermore, other improvements are made on the adaptive algorithm to enhance the testing criteria, as well as making a contribution in the field of parameters customization.

The rest of this chapter is arranged as follows. A research problem is provided in subsection 1.2. The research motivation is detailed in subsection 1.3. Research aim and objectives are discussed in subsection 1.4. Research methodology is discussed in subsection 1.5. Subsection 1.6 shows the contribution of the research. The scope of the research is presented in subsection 1.7 Finally, the thesis structure is shown in subsection1.8.

1.2 Research Problem

There is a myriad of security threats that are caused by botnets, such as distributed denial-of-service (DDoS) attacks, identity theft, and spam (Barford & Yegneswaran, 2007; Dagon et al., 2008; Fabian & Terzis, 2007; Grizzard et al., 2007; Gu et al., 2008; Karasaridis et al., 2007; Levy & Arce, 2006; Rajab et al., 2006). Referring to the FBI's report of the "Operation Bot Roast" project, more than a million IP addresses belonging to normal users had been identified on the Internet, while the number continuously increases. Other statistics display that botnets generate large revenues for bot herders. Gartner estimated that the economic loss generated solely by phishing attacks is about 3 billion US dollars per year (Hsu, Huang, & Chen, 2010). Fast-Flux Service Networks (FFSN) are the core of certain botnet types and play the role of command and control carrier between the mothership and its bots. Fast-flux networks forward and host a scam service to provide a website (back-end server) with high availability, which helps them avoid being tracked and shut down by security professionals (Qassrawi & Zhang, 2012). Risks analytics report from 2016 identified that 84 percent of the campaigns analyzed in Ukraine, host a fast flux proxy infrastructure (Doborjeh & Kasabov, 2016). An attacker earns many benefits from the botnet fast flux techniques (Otgonbold, 2014). The first benefit is simplicity; the attacker can use just a few powerful back-end servers as motherships. FF-agents can also add an extra layer of protection against tracking and discovery. Finally, the extra layer of protection of these FF-agents extends the life span of the motherships.

The 2018 internet security threat report by Symantec was still very much concerned with redirecting the resolution of the DNS responses and the IPs to malicious websites (Semantec, 2018). Due to fast flux service networks, the biggest problem is distinguishing between malicious and benign FFSNs. Looking back to the related work, many researches have tried to differentiate between benign and malicious FFSNs, but they still need to increase the true positive (TP) and true negative (TN), while also trying to achieve an acceptable and accurate ratio of the classification of benign and malicious FFSNs (Martinez-Bea, Castillo-Perez, & Garcia-Alfaro, 2013; Perdisci, Corona, Dagon, & Wenke, 2009; Qassrawi & Zhang, 2012).

Passive, active, and real time approaches are used in fast flux botnet detection. According to Al-Duwairi and Al-Hammouri (2014), the main drawback of the passive approach is the need to deal with a huge amount of DNS traffic traces that correspond to legitimate and non-legitimate domain names. In contrast, the active detection-based approaches deal with less DNS traffic traces that correspond to non-legitimate domain names in most cases. Finally, the real-time approaches suffer from high FP and FN rates. Besides, none of the previously mentioned approaches helped to detect zero-day malicious domains and FFSN while simultaneously keeping track of the detection accuracy and the time required to detect such a botnet fast flux domain.

From the algorithmic point of view, many researchers have indicated that the DeSNN algorithm is one of the eSNN algorithms that has numerous features such as speed, which helps in detecting zero-day fast flux domains in a reasonable period (Hagras, Pounds-Cornish, Colley, Callaghan, & Clarke, 2004; Kasabov, Dhoble, Nuntalid, & Indiveri, 2013; Nuntalid, Dhoble, & Kasabov, 2011; Schliebs & Kasabov, 2013). However, the DeSNN algorithm suffers from the fact that several parameters must be set before running the algorithm. Contributing to this disadvantage, the sub-process of setting the initial weight of the spiking neural network based on the Rank Order (RO) may lead to the misclassification of incoming inputs. Besides, determining the best chosen value for the parameters is a significant problem.

Overall, the problems that are explored and solved in the current thesis are:

- How to adapt the DeSNN algorithm to improve the classification performance.
- How to detect the fast flux domains in online mode using the adaptive DeSNN algorithm.
- How to choose a feature set that maximizes the classification performance.
- How to improve the testing criteria of the DeSNN.
- How to minimize the number of DeSNN parameters.
- How to minimize the memory storage used.

1.3 Research Motivation:

The detection of botnet fast flux zero-day domains that are not caught by existing methods is a significant challenge. This challenge motivated the proposal of a new methodology which would be able to detect the unknown "zero-day" fast flux domains in an online mode. Further motivations include:

- The enhanced robustness of malicious websites. Fast flux assistance attackers keep their sites active as long as possible using victim machines.
- There is an increasing interest in adaptive auto-learning approaches as an effective technology in Internet security, which can be applied to distinguish between malicious and legitimate domains in online mode and high speed.
- The availability of a suitable online approach which is applicable to work in the real world for a lifetime with small memory usage.
- The classification process of DeSNN needs to be modified to help in minimizing the damages made by fast flux attack.

1.4 Research Aim and Objectives

The aim of the proposed research is to build a novel approach for fast-flux botnet detection that utilizes life-long learning, leads to improve classification performance and various capabilities to solve the problem of Fast flux domain detection. To achieve this aim, the following objectives have been defined:

- Develop an approach to solve the problem of Fast-flux service network, especially to detect the malicious unknown zero-day FFDN in online mode, with the minimum memory usage.
- Propose an approach to detect fast-flux domain in online mode using a learning method with a minimum number of parameters used in the proposed algorithm.
- Select the features that will lead to greater accuracy in detecting FF domains.
- Evaluate the proposed approach by comparing it with existing approaches.
- Improve the performance and accuracy (reducing FP and FN rates) of the classification by changing the weights initialization and the classification criteria.

1.5 Research Methodology

The research methodology defines the stages of how the research will be conducted. So, the stages are designated so that each step has a defined set of inputs and the expected output, and how the outputs of each stage helps in the next stage. Besides, the feedback from the front stages to the back ones will help in the process of modifying the errors and improve the performance in order to gain high accuracy or minimize errors. The research methodology employed in this research is depicted in Figure 1.1.

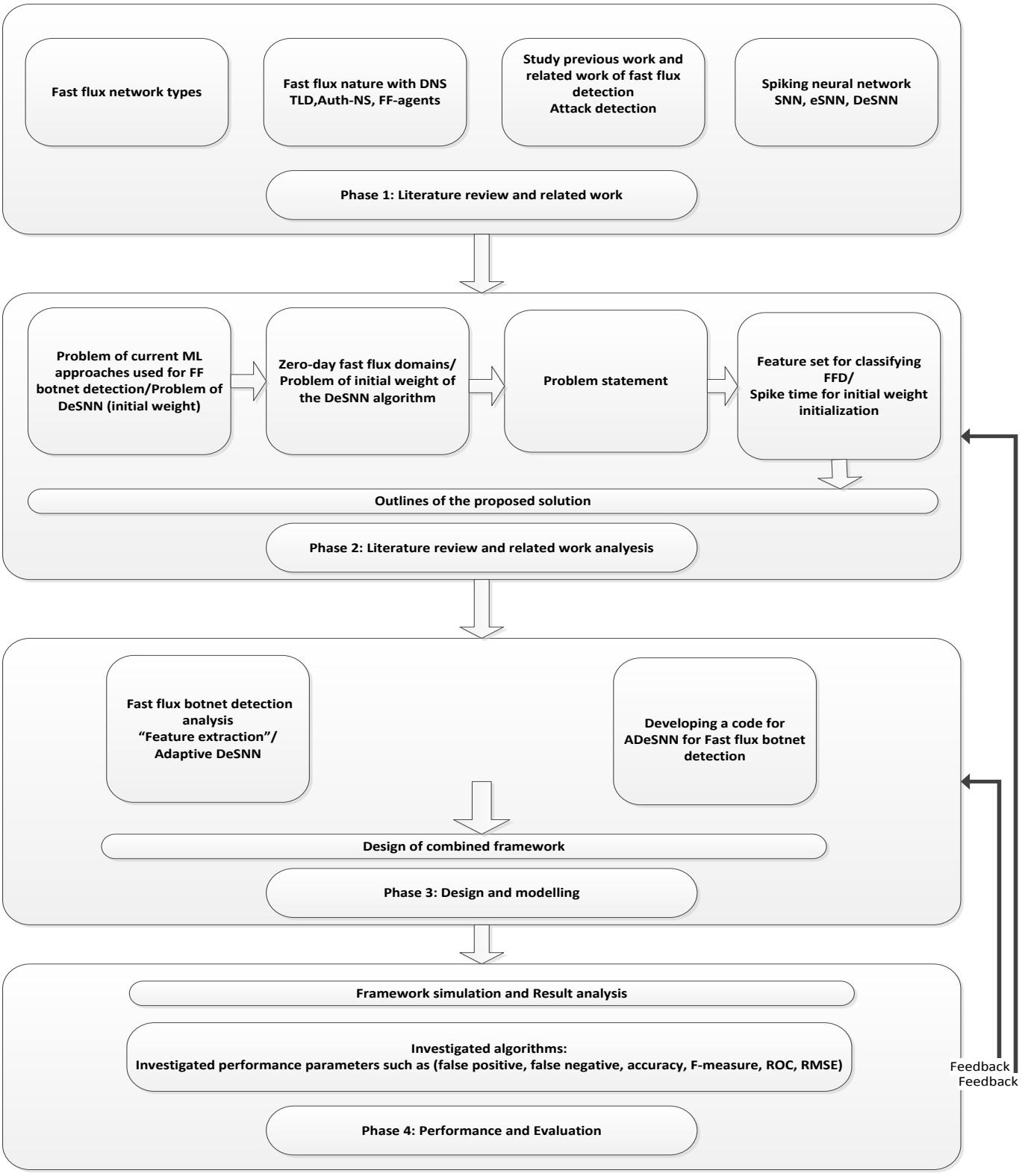


Figure 1. 1 The Research Methodology

The following subsections describe in detail the activities performed during each phase.

Phase 1 (Literature Review)

The literature review discusses the currently developed approaches to filter fast flux botnet attacks, and outlines the most used techniques in detecting FF attacks. In addition, the literature review displays the spiking neural network algorithms and their development to serve the new adaptation proposed in this thesis.

Phase 2 (Literature Analysis)

This phase evaluates the major approaches against FF attacks, which are classified according to the detection method used. This section offers a better understanding of the current problem, possible solutions, and the scope of future studies to detect FFDs.

Phase 3 (Design and Modeling)

This shows the analysis process of a fast flux botnet and how the adaptation of DeSNN positively affects the learning performance, which added a value to the proposed approach which is dynamically used to detect fast flux domains. Also, this section compares the other algorithms used to solve this problem.

Phase 4 (**Performance Evaluation)**

The final phase presents the performance evaluation, the experimental environment of the proposed solution, and the dataset used to show the proposed approach's effectiveness in detecting fast flux domains, especially the zero-day domains. Moreover, the accuracy measures used to prove the enhancement of the improvements made on the DeSNN algorithm according to the proposed adaptations are shown here.

1.6 Contributions of the Research

The main contribution of this thesis is to develop a novel approach called Fast Flux Killer Approach (FFKA), which adapts DeSNN algorithm. This approach has many sub-contributions in the field of fast flux domains detection and are be summarized as follows.

- Increase the detection performance using the adaptive fast one-pass algorithm (ADeSNN). Employ the spike time as the initial weight, then the achieved performance is evaluated using true positive, true negative, recall, precision, f-Measure and overall accuracy.
- Improve the detection accuracy, especially the classification criteria by conducting a similarity measure between the new and already trained inputs.
- Design of a new feature set which can be used with the suggested algorithm to accurately classify fast flux domains.
- Introduce a new adaptive dynamic classification threshold in order to classify new incoming inputs, as well as minimizing the memory storage used.
- Adaptive life-long learning approach able to detect dynamically the unknown zero-day fast flux domains.

1.7 The Scope of the Research

The scope of this work is presented in two tracks. First, this research is about fast flux botnet detection aiming to detect those domains in which they behave like malicious fast flux domain. Second, this proposed approach is implemented as a host-based approach where it is able to be implemented at the local DNS server in order to work as a defender in case of threats and risks.

1.8 Thesis Structure

This thesis is divided into 6 chapter where the word developed to achieve the aim and objectives of this research is described. The next subsections will summarise each of these chapters.

Chapter One:

This chapter gave a brief introduction about fast flux botnets and listed the motivations of the researcher to pursue this study. The research problem, research aim, and objectives are also

listed here. Finally, this chapter presented the research methodology and the structure of the thesis.

Chapter Two:

A solid background about different types of fast flux and domain flux are discussed in this chapter. A rich literature review of what have been done in the area of fast flux is detailed. The author structured this chapter based on the scope of previous methods and solutions that tried to solve the problem of fast flux botnet detection. Additionally, the previous work done related to the proposed algorithm is mentioned. Furthermore, some brief information regarding the evolving spiking neural network is presented. This information is the foundation of the algorithm proposed in this work.

Chapter Three:

Part of the proposed solution was the changing of the initial weight. This is discussed in this chapter. In addition, the adaptive dynamic evolving spiking neural network and the original DeSNN are compared based on two public datasets. The results of the adaptive version with the original algorithm are then discussed in details.

Chapter Four

Here the first part of the proposed FFKA is introduced, the supervised phase which works offline to train the ADeSNN algorithm to detect the fast flux domains based on labelled data. Moreover, the testing criteria was introduced to use the similarity measure to classify the fast flux domains. This chapter also introduces the classification threshold that will be used in the next phase of the FFKA approach in an online mode in chapter five. The chapter concludes by a discussion and chapter summary.

Chapter Five:

This chapter presents the proposed FFKA approach to detect the zero-day fast flux domains in online mode supported by offline mode to enhance the classification performance. Moreover, this chapter shows the improvements on the classification process and the parameters

customization process and compares the results of the proposed Hybrid FFKA approach with the supervised phase in chapter four. The chapter concludes by a discussion and chapter summary.

Chapter Six:

This chapter give the overall discussion and present the conclusion of the work. In this chapter, we state the limitations of this research, recommendations, and the possible future works.

CHAPTER 2

BACKGROUND LITERATURE REVIEW AND RELATED WORK

Chapter Overview

This study covered the literature review regarding the fast flux botnet problem. This chapter is organized based on the three solution scope of the literature approaches which they were the host-based, router-based, and the DNS-based. Then the related work from the literature was discussed.

2.1 Background

Numerous websites provide commercial services to users. The efficiency of these services is highly dependent on their availability. Server systems are distributed to large redundant service networks in multiple areas to achieve high availability (Scharrenberg, 2008). The DNS is a hierarchical distributed naming system for computers and resources that are connected to the Internet (Shaikh, Tewari, & Agrawal, 2001). A browser usually automatically acquires the IP address of the desired host name to access a website. The DNS server typically returns the same reply each time. Thus, the same IP address is returned each time a host name is requested. Some requests, such as RRDNSs, CDNs, and fast flux service networks (FFSNs), do not work in the same manner as previously described. RRDNSs, CDNs, and FFSNs share similar characteristics, such as a low time to live (TTL) feature. RRDNSs and CDNs are DNS-based methods for load balancing that provide a high degree of performance, availability, and scalability for content websites. RRDNSs distribute user requests to their distributed servers by swapping the IP addresses of the DNS response of the same domain each time to provide load balancing. CDNs represent a network of globally distributed nodes to return the IP address of the nearest accessible node to the client; they thus support service speed and availability. Similarly, fast flux uses a similar concept of frequently changing IP addresses that correspond to a specific domain. This strategy helps cyber-criminals to remain undetected. The main

difference between FFSNs and CDNs is that CDN nodes are fully administered machines, whereas FFSNs are malware-infected computers (Lin et al., 2013).

The business side of fast flux hosting begins with malware authors. By developing phishing kits, this software package can be used to deliver phishing emails to a set (list) of victims and host an illegal website to which those emails are directed. Others sell lists of addresses for spam purposes, whereas others improve Bot software. A flexible, remotely controllable software known as bot software enables subsequent downloads on a particular computer once it has been installed on a victim's computer. E-mail-borne worms are used by bot herders to infect and exploit thousands of computers. Such tools are the most valued these days by malware authors and cyber-criminals. Malware authors and bot herders are significant sections of the cyber-criminal community (ICANN Security and Stability Advisory Committee (SSAC), March 2008).

FFNs provide high availability and reliability to scam websites ("GNSO Fast Flux Hosting Working Group Publishes Final Report," 7 August 2009). The ICCAN report (ICANN Security and Stability Advisory Committee (SSAC), March 2008) defines a fast flux technique as one in which multiple IP addresses (sometimes hundreds or even thousands) are assigned and re-assigned to a single fully qualified domain name (FQDN), such as www.example.com. The URLs and domain names for the announced content are not resolved to any IP addresses of back-end servers. Instead of pointing to back-end servers, the URLs and domain names addresses are changed among many front-end agents, which serve as redirectors; thus, the content is forwarded to the back-end servers (the mothership) (Gasster, 2008; "GNSO Fast Flux Hosting Working Group Publishes Final Report," 7 August 2009; ICANN Security and Stability Advisory Committee (SSAC), March 2008).

Fast flux mainly involves two techniques, namely, the IP fast flux and the domain flux. The IP fast flux comes in two types as depicted in Figure 2.1: the single fast flux and the double fast flux. An extension type of the double flux is called the hydra flux (Subsection 2.1.3). The details of these techniques are discussed in the subsections that follow.

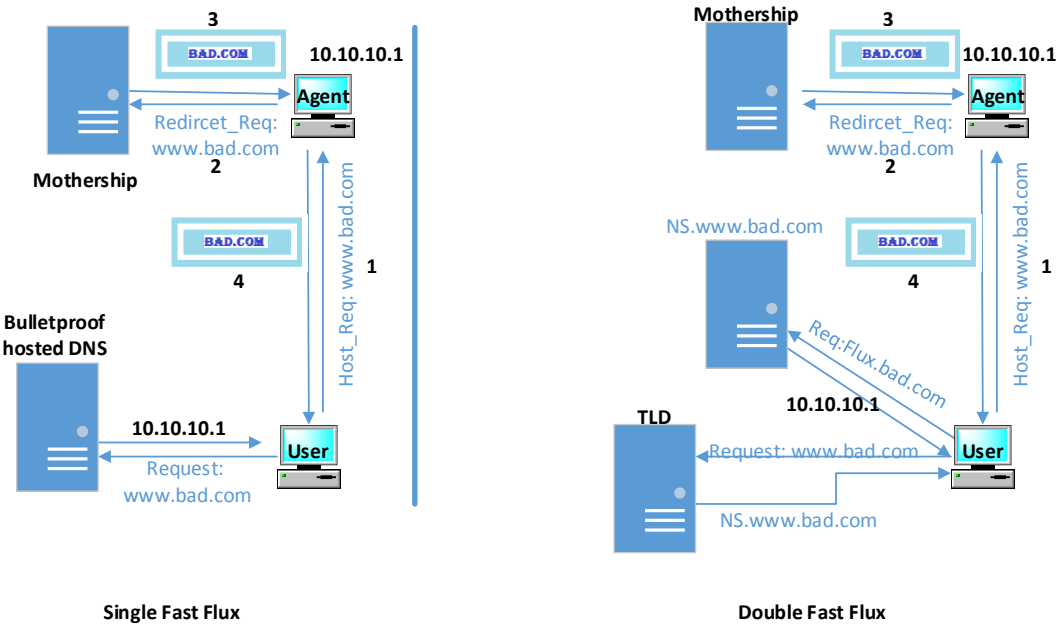


Figure 2. 1 Comparison of IP resolutions of fast flux techniques

2.1.1 Single Fast Flux

Domain names are registered in an official registrar by an attacker for use in illegal activities by an official registrar. The attacker registers a domain name for an FFSN referring to illegal websites (e.g., bad.com) and another domain name (Resolvernameserver.com) to serve the mapping domain name resolution services. As mentioned previously, the attacker adds IP addresses to the bulletproof server and then provides the control of the FFSN to a mothership.

In a single fast flux as displayed in Figure 2.2, the attacker deploys a bulletproof server to host the zone file. The bulletproof web hosting server leads customers to the desired malicious website. Such services are well-known among botnet owners, who need a reliable environment, and assist in deploying a botnet C&C server.

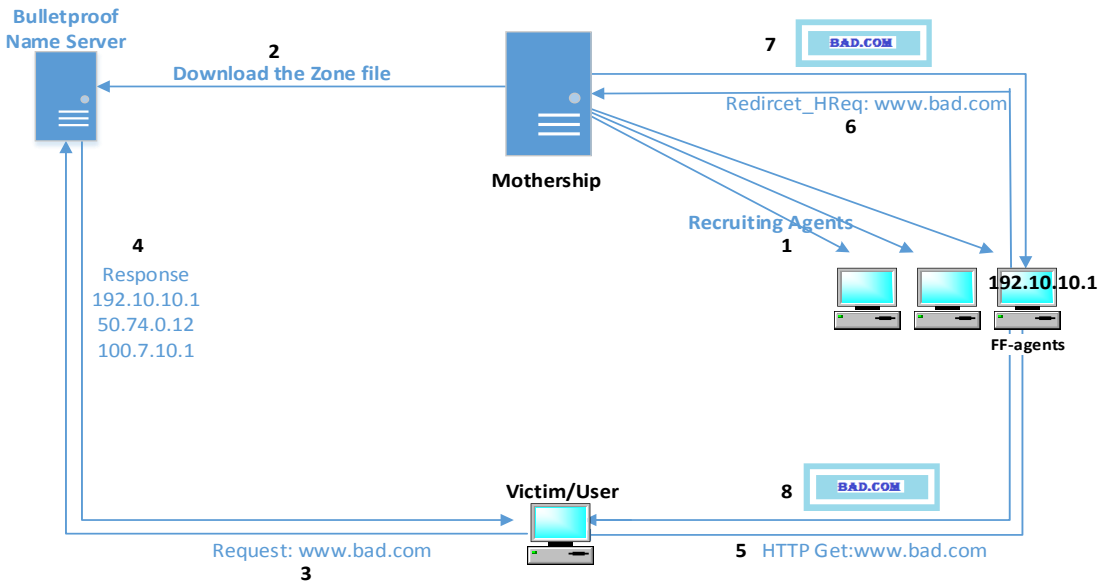


Figure 2. 2 Single fast flux of IP addresses of a malicious website

Figure 2.2 shows the process of a single fast flux of the IP addresses of a malicious website.

1. The attacker recruits some of the compromised computers to work as proxies, which directly redirect user requests to the mothership/operator.
2. The attacker adds the name server (Resolvernameserver.com) and records of the malicious website (www.bad.com/mothership) to the zone file via the registrar.
3. The victim (user) requests the FQDN (www.bad.com). Hence, a request is sent to the DNS looking to resolve the FQDN. Assuming the absence of caching, a recursive DNS server asks for the authoritative name server for this FQDN. The part of the recursive process from the top-level domain (TLD) to the authoritative server is omitted.
4. Instead of sending the IP address of the FQDN (www.bad.com), the authoritative name server sends back a list of the IP addresses of the proxies to the user.
5. The user initiates a GET message to one of the IP addresses in the list.
6. The FF-agent (proxy) simply redirects the message to the malicious web server (the mothership) to handle the message.
7. The malicious web server sends the response (answer) back to the FF-proxy.
8. The FF-agent returns the response to the user.

The A records of the web servers are constructed with short TTLs (Holz, Gorecki, Rieck, & Freiling, 2008), "A" is the resolved record information returned from DNS. The FFSN operators directly provide a new set of A records to replace the old set of records (of the FF-agents) when the TTLs of the request expire. Thus, there is very little chance of identifying and shutting down the web servers, which are supported by this FF technique. The records associated with the illegal website in the zone file of the DNS bot (Resolvernameserver.com) might appear as follows:

```
bad.com. 180 IN A 192.10.10.1

bad.com. 180 IN A 50.74.0.12

bad.com. 180 IN A 100.7.10.1
```

The TTL for each RR is clearly very low (180 s). The RRs are directly replaced with new bot (FF-agents) IP addresses when the TTL expires. The zone file might be read as follows after a time of TTL+1:

```
bad.com. 180 IN A 155.1.1.14

bad.com. 180 IN A 180.88.0.9

bad.com. 180 IN A 120.1.1.2
```

2.1.2 Double Fast Flux

Furthermore, the fast flux mothership/operator identifies the abovementioned domains, which correspond to its FFSN. The FF-agents in the two FFSNs are separated to simplify the understanding of the idea behind the double fast flux because FF-agents are commonly used to serve both DNS and HTTP requests at the same time (Xu, Wang, & Xie, 2013) as the mothership/operator. Figure 2.3 shows the double fast flux process.

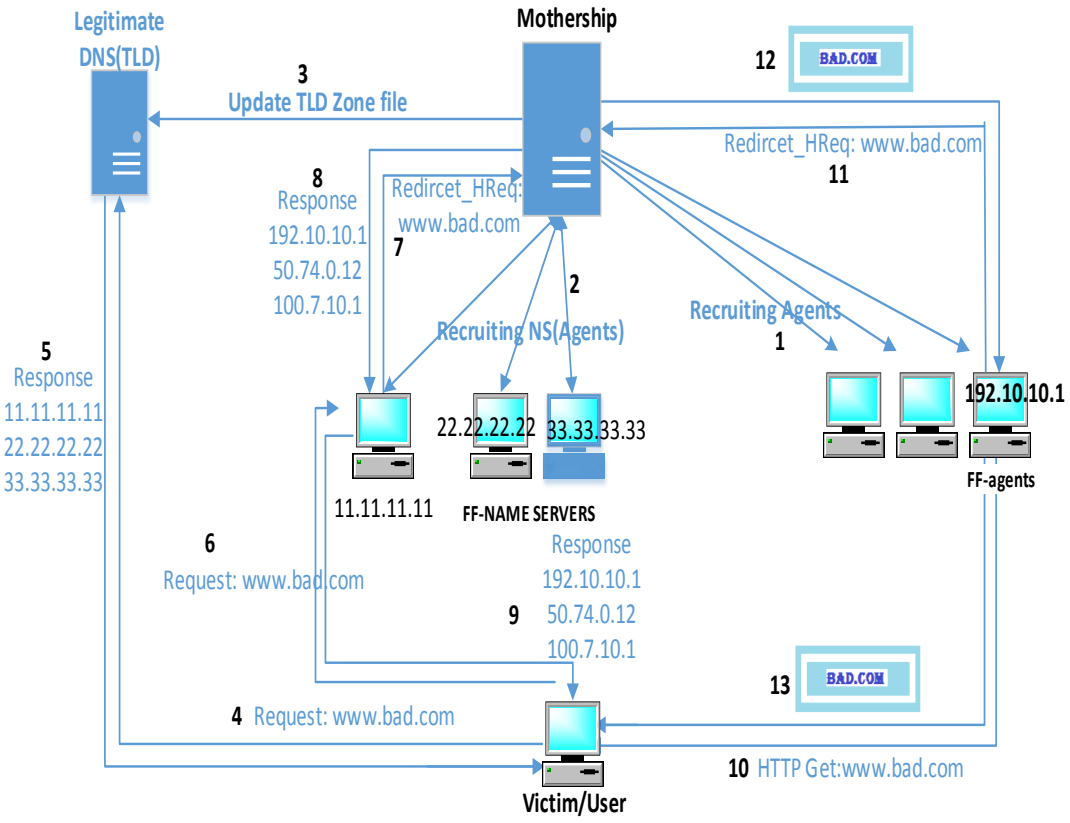


Figure 2. 3 Double FFSN of name server and IP addresses of the malicious website

Figure 2.3 summarizes the double fast flux process of the IP addresses of the malicious website and the authoritative name server.

1. The attacker recruits some of the compromised computers to work as proxies, which directly redirect the user request to the FF mothership/operator.
2. The attacker recruits some of the compromised computers to work as NS proxies, which directly redirect the DNS request to the mothership/operator.
3. The attacker adds the name server records (Resolvernameserver.com) to the TLD zone file via the registrar and keeps updating the legitimate DNS RR of the authoritative name servers of the malicious domain.
4. The victim (user) sends a request for (www.bad.com) to the DNS server to resolve the FQDN.
5. The DNS returns a list of authoritative name servers for this FQDN, which are a part of the malicious compromised pool of NS agents.
6. The user sends the authoritative NS asking for the IP address of the FQDN.

7. The authoritative name server forwards the DNS request to the mothership instead of resolving and directly returning the IP address of the FQDN.
8. The mothership returns a list of IP addresses that are FF-agent proxies of the website server (mothership).
9. The authoritative name server sends the IP addresses back to the user.
10. The user initiates a GET message to one of the IP addresses in the list (which is actually one of the FF-agents).
11. The FF-agent (proxy) simply redirects the message to the malicious web server (mothership) to handle the message.
12. The malicious web server sends the response back to the FF-agent.
13. The FF-agent returns the response to the user.

The attacker continuously updates the NS records of the TLD. Through the registrar, the domain owner has the ability to modify the domain information. The attacker frequently changes the IP addresses of the NS servers to point to different hosts and sets the TTL value for these NS servers to a very small value (e.g., 180 s). The RRs of the NS might be shown in a TLD zone file as follows:

```
bad.com. NS NS1.Resolvernameserver.com
bad.com. NS NS2.Resolvernameserver.com
NS1.Resolvernameserver.com A 11.11.11.11
NS2.Resolvernameserver.com A 10.0.0.2
```

The attacker automatically replaces the A records of the NS when the TTL expires. Therefore, the RRs of the NS might be shown in a TLD zone file as follows:

```
bad.com. NS NS1.Resolvernameserver.com
bad.com. NS NS2.Resolvernameserver.com
NS1.Resolvernameserver.com. A 22.22.22.22
NS2.Resolvernameserver.com. A 10.10.10.233
```

Consequently, there is very little opportunity to detect and shut down the name servers that support this fast flux attack. Combining the two FFSNs is an effective method for keeping the website alive for longer periods than websites that do not use the same techniques.

2.1.3 Hydra Fast Flux

The new advanced FFSN does the same thing as the traditional FFSN, but taking it down is impossible. Similar to the traditional FFSN, the mothership of the new advanced FFSN can be deactivated by law enforcement, but the bots have an alternative IP address to another mothership related to the same FFSN. As depicted in figure 2.4 the Asprox botnet, the bots download a list of available motherships. Ultimately, alternative IP addresses adds a multilayer of double fast flux to the botnet and maintains extra availability to the malicious content. Figure 2.4 depicts the multilayer FFSN of the Asprox botnet, which is usually denoted as a hydra-flux service network.

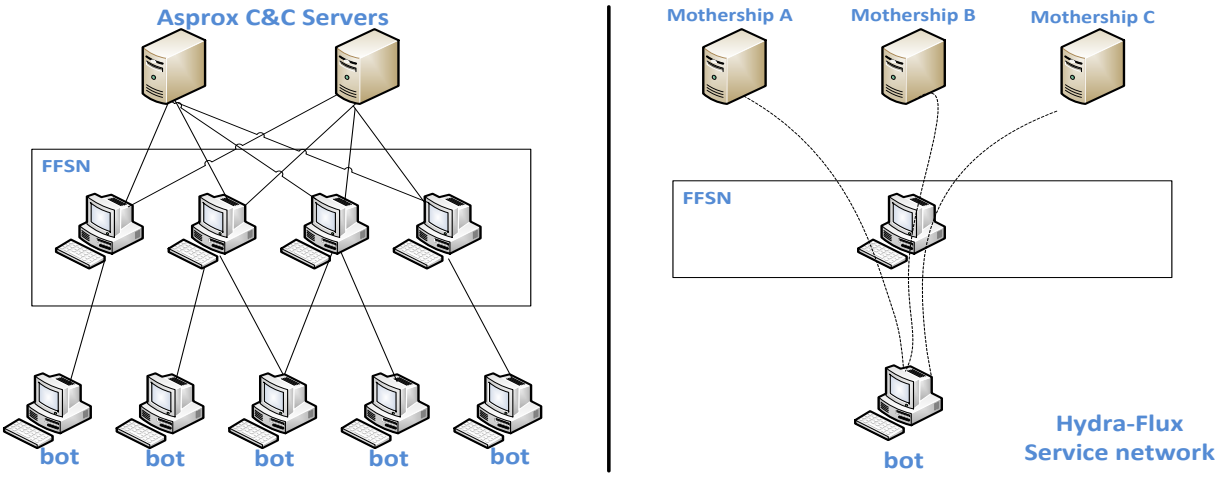


Figure 2. 4 Multilayer FFSN of the Asprox botnet and hydra-flux service network

2.1.4 Domain Flux

Another type of fluxing technique is the domain flux. In contrast to the fast flux of the IP addresses related to a domain name, the domain flux is the process of fluxing domain names related to a URL of the C&C server.

The domain flux is used by bots to contact the C&C server. The domain generating algorithm generates the same domain names for both the C&C server and its bots when seeded with the same value. The C&C server is used to register some of the auto-generated domains. Stone-Gross et al. (2009) revealed that the Torpig botnet calculates domain names by combining the current week and year and adding the TLD (e.g., "weekyear.net") to them. These auto-generated domains are then used by bots to contact the C&C server; if the connection fails, then the bots attempt to use the day information to produce the daily domains. If all the domains fail, then the bots use the hard-coded domain names in their configuration file as a last resort (Stone-Gross et al., 2009). All of these generated domain names are sent to the DNS server in an attempt to resolve it. The bots then establish contact with the C&C server. This process of failed requests generates a high observable number of non-existing domain responses in the DNS traffic that create a footprint of these bots that send most of the failed DNS requests (Jiang, Cao, Jin, Li, & Zhang, 2010; Pappas et al., 2009; S. Yu, 2014).

2.2 Literature Review

Numerous studies have explored botnet detection, especially fast flux botnet detection (Al-Duwairi & Al-Hammouri, 2014; Chahal & Khurana, 2016; Z. Chen, Wang, Zhou, & Li, 2011; Scharrenberg, 2008; Yu, Zhang, Kang, & Chen, 2012). Most previous researches discussed the detection of FFSNs or malicious fast flux domains, which serve as the main element of the fast flux botnet technique. The related works on fast flux argued about fast flux in terms of what is fluxed or what technique is used to detect an FF domain. However, to the best of our knowledge, the present study is the first to investigate fast flux botnet approaches on the basis of the solution scope of detection techniques as depicted in Figure 2.5.

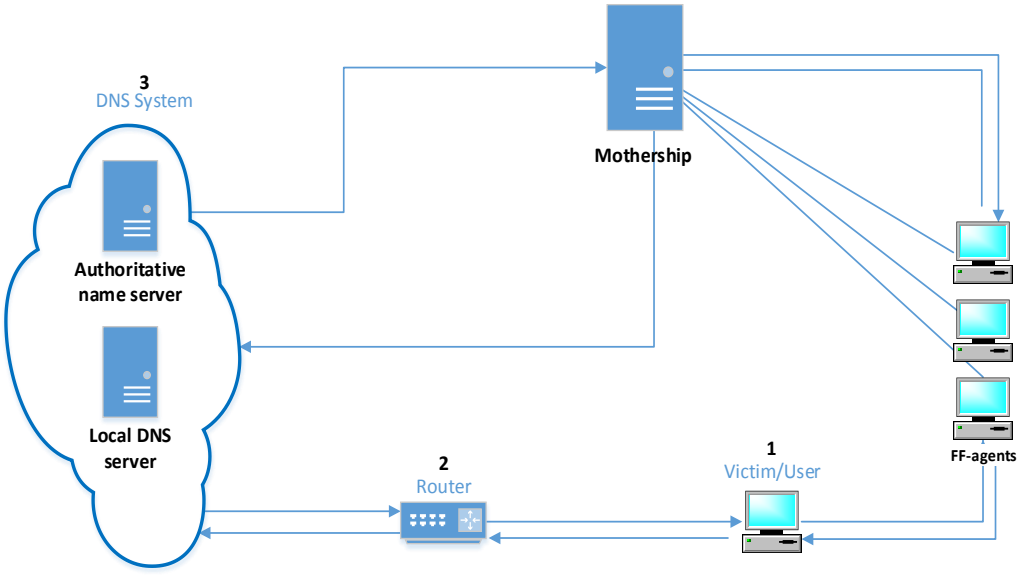


Figure 2. 5 Solution scope of FF botnet detection methods

Figure 2. 5 presents the solution scope of fast flux botnet detection. In addition, the current study classifies fast flux botnet approaches according to the solution scope. Hence, number 1 in Figure 2.5 refers to host-based methods, number 2 refers to router-based methods, and number 3 refers to DNS-based methods. Moreover, the current study discusses the mode of each detection technique and identifies whether it is active, passive, or real time as depicted in Figure 2. 6. Within these parts, each approach discusses the features, the datasets, and the classifier used.

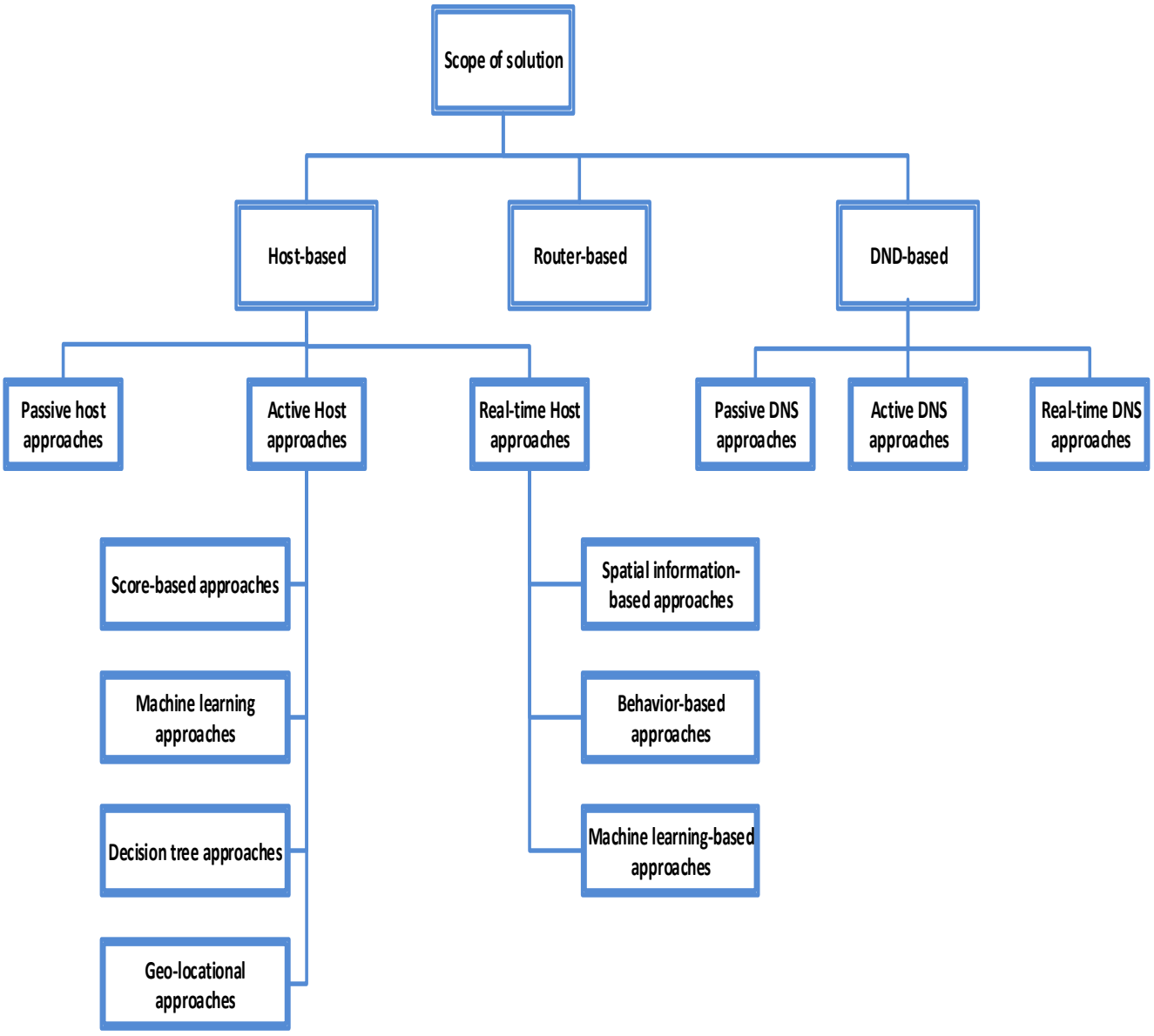


Figure 2. 6 Chart of the solutions scope

2.2.1 Host-based Detection Methods

Host-based means that the proposed approach is applied to a host device or a set of devices from the user point of view. According to the literature, the majority of the reported works were using the host-based detection approach. These approaches are divided into three subgroups: passive, active, and real-time approaches.

2.2.1.1 Passive Host-based Approaches

The idea behind passive approaches is that they rely on the monitoring part of a specific network area for a period of time. The collected data are then analyzed to prove predefined propositions. Passive monitoring provides the detection methods the advantage of not being noticed by attackers and adds no extra traffic flows to the network.

A Bayesian method is proposed to detect bots on the basis of DNS traffic similarity (Villamarín-Salomón & Brustoloni, 2009). The proposed approach relies on the idea that a bot at the same botnet has the same traffic similarity as the other botnets. One bot should be known at the beginning; then, the search for other bots with the same traffic similarities in the DNS traffic is initiated. However, the poor tuning of parameters generates large false positive (FP) values (Villamarín-Salomón & Brustoloni, 2009).

Another method of using decision trees to identify malicious FFSNs was proposed in (Zhao & Traore, 2012). The classifier begins by classifying malicious domains, and then monitors the suspicious ones for a longer period. The proposed approach may be able to identify legitimate and malicious FFSNs, but it may not easily classify them on the basis of malicious website behaviors. The proposed approach is also unable to detect unknown FFSNs, as well as unknown zero-day domains. In addition, the author suggested generating a new system that can develop its classifier while running on the basis of an existing dataset and newly generated data, which would enable the system to identify new threats (Zhao & Traore, 2012). Table 2.1 summarizes the passive approaches.

Table 2. 1 Summary of passive approaches

Authors	Algorithm	Mechanism	Advantages	Disadvantages
Zhao and Traore (2012)	Decision tree	Monitoring malicious domains to detect FFSNs	Low computational complexity	-Classification problem -Unable to detect unknown zero-day domains
Villamarín-Salomón and Brustoloni (2009)	Bayesian method	Detecting bots based on DNS traffic similarities and known bot traffic	Effective and robust	Parameter tuning causes FP

Overall, fast flux botnet domains still need to be detected in a short time because of the quick change in the IP addresses of motherships that hampers the easy tracking of their locations. Thus, detecting this type of "Fast Flux zero-day" domains as quickly as possible is important. Moreover, passive approaches deal with a huge amount of data and are thus unsuitable for fast processing in a short time with few resources.

2.2.1.2 Host-based Active Approaches

In contrast to passive approaches, active approaches require assistance from third-party data sources, such as the WHOIS or GeoIP database. Such third parties provide additional necessary information (e.g., IP address registrar name and creation data). The following subsections describe related works that applied host-based active approaches.

A) Score-based Approaches

Many fast flux domain detection approaches are based on the flux score calculation of a set of features adopted (Al-Duwairi & Al-Hammouri, 2014; C.-M. Chen, Cheng, & Chou, 2013; Holz et al., 2008; Hsu et al., 2014; Koo, Chang, & Chuang, 2012; Otgonbold, 2014; Sheng, Shijie, &

Sha, 2010). Holz et al. (2008) proposed a system that measures and detects a FFSN on the basis of the calculated flux score. Their proposed system takes malicious domains from spam emails and then uses the Dig tool to generate DNS lookups and reverse DNS lookups and thereby obtain necessary information about a feature set (number of A records, number of autonomous system numbers (ASNs), and number of (NS). Thus, the flux score calculation is fed for use later in distinguishing between malicious FFSNs and legitimate ones. Their results showed that the proposed system achieves a detection accuracy of 99.98%. However, the coefficients used in the score calculation require modification to ensure the highest possible accuracy of the detection system. Moreover, the set of features chosen cannot purely distinguish between FFSNs and CDNs.

Hsu et al. (2014) proposed a fast flux domain detector (FFDD) system that adds to Holz's source of malicious domains and taking unknown URLs from spam or social networks. The FFDD system is used to calculate the flux score on the basis of the response time series between each of the two subsequent requests from a host to the FF-agent. The FFDD is a lightweight standalone system that does not need support from other parties. Consequently, the FFDD can accurately detect a fast flux domain with 3% FP and 2% FN in less than 20 min. Therefore, this technique is not suitable for fast flux detection.

Sheng et al. (2010) proposed two metrics, namely, the average online rate (AOR) and the minimum availability rate (MAR) to detect fast flux agents on the basis of the agents themselves. The calculations of these two methods are initiated from the beginning of the monitoring process. The monitoring is extended for 1 h using the AOR and MAR calculation once a malicious domain is detected. The results show that most FFSNs have lower values than legitimate ones. Moreover, these methods are easy to implement and deploy and are useful for distinguishing between benign and malicious FFSNs but not for FFSN detection. However, the metrics may work incorrectly if the group of agents is small or only a few agents are found (Sheng et al., 2010). According to Sheng, the metrics depend on the quality of the HTTP service, which may affect network accessibility and thus stop reaching agents.

The Google search engine has also been used as a technique to classify malicious domains by feeding the search process with IP addresses of suspicious domains (Al-Duwairi & Al-

Hammouri, 2014). The number of hits is then observed. As expected, the number of hits comprising domains associated with FFSNs would be much less than the number of legitimate domains. The new legitimate domain could also mislead the classifier. The proposed system is still at its infancy and thus needs other features to confirm its detection accuracy.

Koo et al. (2012) proposed a computed formula to detect malicious domains being used in FFSNs, with the domains obtained from a malware domain list. They explored the actual status of FFSNs employed in cyber-crimes and analyzed the distribution of compromised computers. Consequently, the detection accuracy is high. However, their data were not sufficient to estimate the scope of the FFSN. Thus, their proposed procedure may lead to misclassified domains.

(Chen et al., 2013) proposed a probability formula to detect malicious fast flux domains. The network behavior of malicious domains are formalistic based on the time–space behavior of malicious FF-domains. In addition, an analysis was proposed to reduce the time complexity of feature modeling. The results of this study show that the proposed solution performs better than the blacklists. However, a threshold is still needed to compute the probability formula. Moreover, gathering information about domain names requires more time, which affects detection performance.

Otgonbold (2014) proposed a fast flux formula to help detect fast flux domains in the wild. The proposed ADAPT system takes inputs from the domain zone file to collect the DNS information needed in the detection system. The zone file is targeted because it contains domains scattered all around the globe using the Tor network as shown in Figure 2. 7. The system's clients gather suspicious domains from various DNS servers over the Tor network and then analyzes the collected information. Thus, the decision is made as to whether the domain needs further scanning to confirm its maliciousness. The results of this study indicate that the proposed system is capable of detecting malicious fast flux domains in their infancy. However, the RDNS server should be queried to collect full DNS information, and such requirement could affect detection performance. The current version of Grails also shows a memory leak problem, which causes out-of-memory exceptions and long-running tasks. Table 2. 2.2 summarizes the calculated score-based approaches.

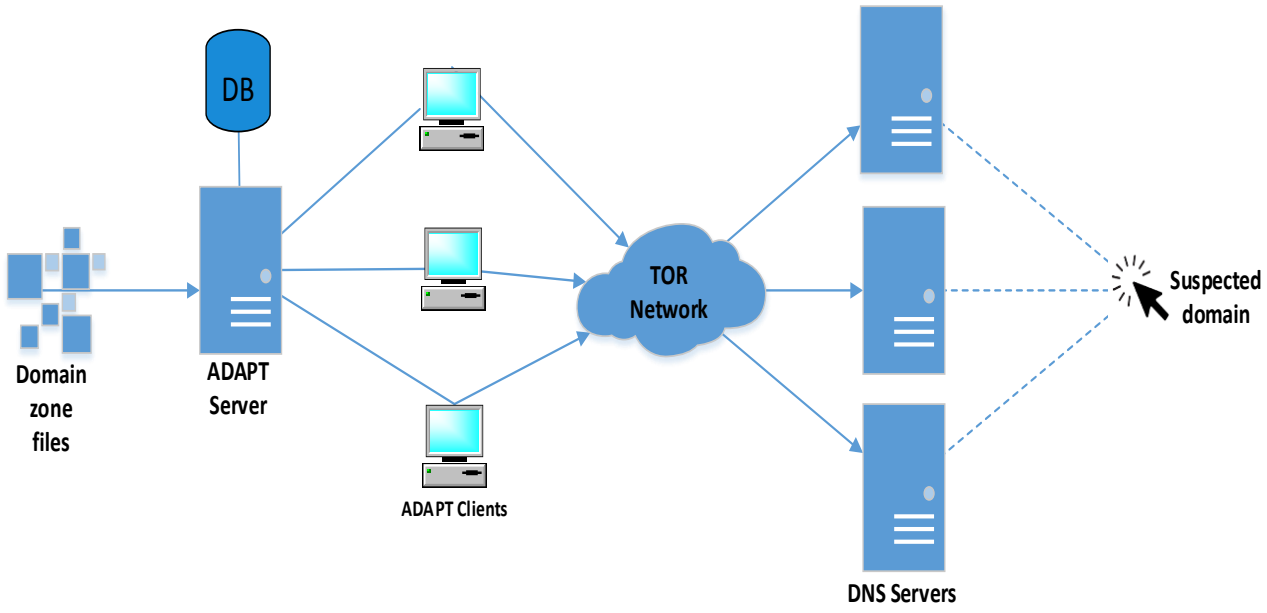


Figure 2. 7 ADAPT system architecture(Otgonbold, 2014)

Table 2. 2 Summary of score-based approaches.

Authors	Algorithms	Mechanism	Advantages	Weakness
Holz et al. (2008)	Flux score	The flux score is computed on the basis of DNS records	Uses two consecutive DNS lookups	- Coefficients require periodic adjustment - Feature is not distinguishable
Hsu et al. (2014)	Flux score	Fast flux score is computed on the basis of the response time differences of subsequent requests of FF domains	Lightweight stand-alone system	Long detection time

Sheng et al. (2010)	AOR, MAR	Once the existence of a fast flux domain agent is discovered, its activities are monitored every hour using calculations based on AOR and MAR	Easy to implement and deploy; metrics are time saving	- Inaccurate result - Based on the quality of the HTTP service
Al-Duwairi and Al-Hammouri (2014)	Number of hits in the Google search engine	Depending on the number of hits of query responses using the Google search engine	Lightweight approach	- Still in the development phase - Needs more features to confirm detection accuracy - Misclassifies new domains as malicious
Koo et al. (2012)	Calculated formulas	Calculated formulas based on the actual status of the FFSN being employed	High detection accuracy	- Data problem - Misclassified domains
Chen et al. (2013)	Probability formula	- Time–space behavior of malicious FF domains and network behavior of domains are formulistic	Outperforms blacklists	- Threshold is needed - Long detection time
Otgonbold (Otgonbold, 2014)	Flux score formula	- Detection system collects domains from DNS zone files	Detects malicious fast flux domains in their infancy	- RDNS servers should be queried,

		- Anonymously provides domains all around the globe in a short period of time with little resource using the Tor network		which could affect performance - Out-of-memory exception

B) Machine Learning-based Approaches

A number of machine learning algorithms are used to classify domains as either malicious or benign (Chen, Huang, & Ou, 2014; Passerini, Paleari, Martignoni, & Bruschi, 2008) as summarized in Table 2.3. In the naïve Bayes classifier proposed by Passerini et al. (2008), all malicious domains are collected from spam emails. Their detection and monitoring "FluXOR" system relies on the idea of a host being a victim to such scam. The system begins to send requests and gathers the feature set information to feed the naïve Bayes classifier as in Figure 2. 8. The naïve Bayes classifier is a supervised algorithm, which is not suitable for detecting unknown attacks. FluXOR reduces the time of detection to 1–3 h, which is still relatively long; a domain with a TTL of more than 3 h is still considered legitimate (Huang, Mao, & Lee, 2010).

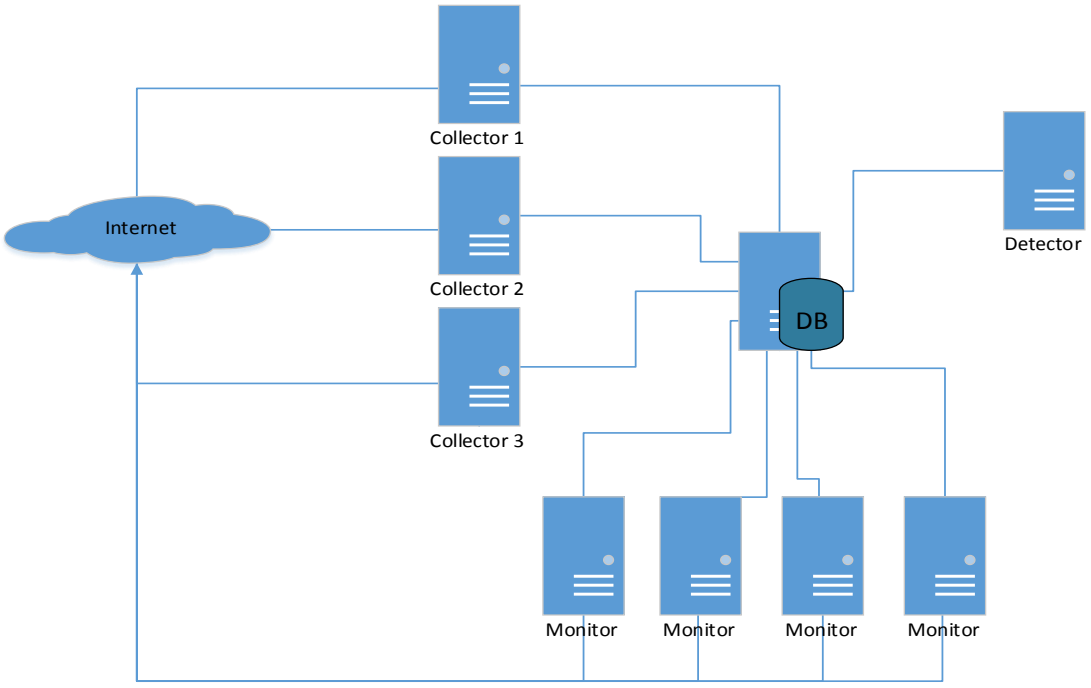


Figure 2. 8 FLUXOR system deployment(Passerini et al., 2008)

Chen et al. (2014) proposed a Bayesian probability theory to distinguish between benign and malicious domains using dissimilar ASNs, reverse DNS lookups, and domain registration time features. They aimed to detect a fast flux website on the basis of its fluxed characteristics. The result of this proposed system presents its ability to identify possible threats. Nevertheless, their judgment was not perfect enough to reflect the good precision of the proposed system.

Chen et al. (2011) used the k-nearest neighbor (KNN) and random forest (RF) as sampling techniques to solve the imbalanced problem, with respect to FFSN detection. In addition, they proposed a sampling technique that is combined with feature extraction from datasets for use in fast flux detection. The result showed that the TTL is an important feature to the classification of the proposed technique. However, its detection accuracy in the case of a long TTL is affected.

The support vector machine (SVM) was proposed by Yu et al. (2012) to detect fast flux botnets by analyzing the patterns of DNS queries from FF botnets. They extracted six features to build the weighted SVM classifier for use in distinguishing legitimate and FF botnet domains. They noted that using SVM to identify fast flux botnets is effective and provides a satisfactory

detection accuracy. Overall, the proposed method entails a long detection time because it waits for additional information from a third party. Moreover, such a supervised method is not helpful in detecting new and unknown zero-day attacks.

Table 2. 3 Summary of machine learning approaches.

Authors	**Algorithms**	**Mechanism**	**Advantages**	**Weakness**
Passerini et al. (2008)	Naïve Bayesian classifier	Analyzes a set of features observed from the victim's point of view on botnet scams	Reduces detection delay	- Long detection delay - Unable to detect zero-day domains
Chen et al. (2014)	Bayesian probability theory	Uses different characteristics to distinguish benign and malicious domains	Enhances detection accuracy of web-based botnets	- Achieves inaccurate precision
Chen et al. (2011)	KNN and RF	Use the resampling technique to solve the imbalanced classification problem with respect to FFSN detection	Solve the imbalanced dataset problem	- Long TTL affects detection accuracy
Yu et al. (2012)	Weighted SVM	Extracts six features to the weighted SVM by analyzing the patterns of DNS responses to FFSNs	Satisfies detection accuracy	- Earlier domains create FP - Unable to detect zero-day domains

C) Decision Tree-based Approaches

Celik and Oktug (2013) proposed the C4.5 decision tree algorithm to evaluate various DNS feature sets and put forward a detection approach, which is a high-dimensional feature vector with various features, including timing network, spatial, and NS and DNS response information. C4.5 evaluates each feature set of previous vectors and decides which one is the best feature vector on the basis of detection accuracy. Combining all features together provides a detection accuracy of 98.9%. However, the detection is unaffected in those timing and domain name feature sets. The C4.5 unsupervised algorithm depends on clustering and is good for detecting unknown attacks; however, it suffers from a low accuracy level in most applications (Almomani, Gupta, Atawneh, Meulenberg, & Almomani, 2013).

D. Zhao and Traore (2012) proposed another method (REPTree) for botnet detection using a decision tree with reduced error pruning. This type of machine learning decision tree is used to classify and identify malicious FFSNs by defining and computing some of the network metrics captured from network flows. Although decision tree-based classifiers are considered as a well-known classification technique with low computational complexity, the authors were not sure of the results because some benign websites were misclassified as malicious websites. They also searched for other reliable evidence. Table 2. 4 summarizes the approaches using the decision tree algorithm.

The classification and regression tree algorithm is used in the method proposed by Y. Zhao and Jin (2015). This method uses a small dataset to quickly distinguish legitimate and malicious FFSNs. This method is mainly based on FFSN domains, DNS, and the process of HTTP visiting. The domain distinct features are shown in Figure 2. 9. Another researcher used distinct mapping of features (Pa, Yoshioka, & Matsumoto, 2015). The classification process needs less than a few days, and the detection accuracy is 90%. The detection time is also relatively long, and other detection methods exhibit higher accuracy and lower detection time. Moreover, this method cannot detect zero-day domains.

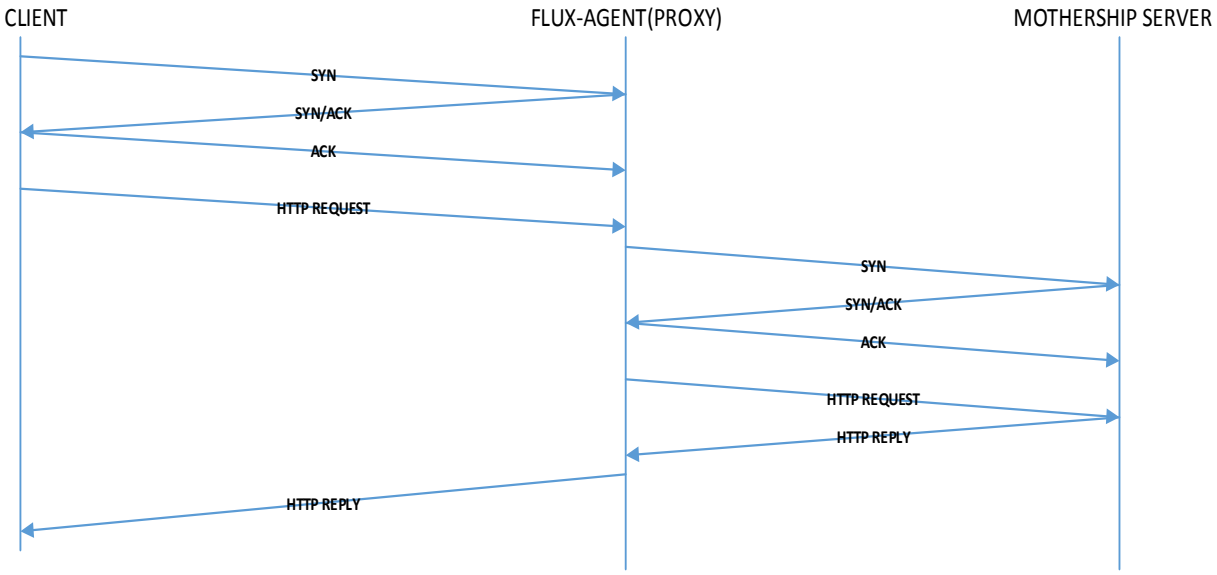


Figure 2. 9 The process of visiting a domain(Y. Zhao & Jin, 2015)

Table 2. 4 Summary of the approaches using decision tree algorithms.

Authors	**Algorithms**	**Mechanism**	**Advantages**	**Disadvantages**
Celik and Oktug (2013)	C4.5 decision tree	A number of feature sets are experimented on to detect FFN.	Detect unknown attacks	- Unaffected by some of feature sets - Low level of accuracy
D. Zhao and Traore (2012)	Decision tree using reduced error pruning (REPTree)	Computed metrics of captured network flows that are analyzed using REPTree.	Low computational complexity	-Misclassification -Needs additional discriminators

Y. Zhao and Jin (2015)	Regression tree algorithm	Detect FFSN domains on the basis of the intrinsic features of DNS analysis and the process of HTTP visiting.	Ability to classify FFSN domains	**-Long detection time** **-Low accuracy rate** -Unable to detect zero- day domains

D) Geo-informational Based Approaches

A constraint-based geolocation technique was employed in (Castelluccia, Kaafar, Manils, & Perito, 2009), and the proposed approach utilizes a geo-localized fast flux hidden server. Thus, mean error distance is used in this approach to determine the physical location of the mothership server. As a result, their approach localizes the mothership with a mean error of below 100 km. However, the system requires extensive resources, achieves low precision, and is incomplete. Buhariwala (2011) used the same technique and determined that the 100 km mean error is inaccurate; moreover, the result indicated that the right error value is 1,000 km from the mothership server. A virtual private proxy server was proposed to decrease the overhead of requesting data from the content server. The result indicated a 300 km mean error, which is three times better than that obtained by Castelluccia et al. (2009). A large mean error rate for physically localizing the mothership server still exists. Table 2. 5 summarizes the methods that utilize geo-information.

The system proposed by Stalmans, Hunter, and Irwin (2012) used Moran's I and Geary's C features to produce classifiers to detect the fast flux C&C domain names of C2 servers. The proposed system can detect domain names on the basis of the geographic locations of C2 servers. Moran's I assumes that close geographical C2 servers are similar, whereas Geary's C measures the spatial autocorrelations between C2 servers. Their system can reliably detect FF domains with a small FP rate. Moran's I measurement is influenced by the number of white spaces at a large scale (Stalmans et al., 2012).

Stornig (2013) employed another approach in which Moran's I of spatial autocorrelation and spatial service distance are used to classify legitimate and non-legitimate fast flux domains. This approach is based on the geo-information of the distributed IP addresses of FF-agents. The spatial autocorrelation between two distant geographical points means that they are not similar, and close points share more similarities. The spatial service distance denotes the average distance between the geolocation of the IP addresses that is correlated with the same domain and the geolocation of the IP addresses of the name server. As a result, the author was convinced that the proposed approach is accurate and lightweight for detecting fast flux domains with low FPs. However, botmasters could cause the detection approach to yield to misclassified results by changing the distribution of the IP addresses of the agents.

Table 2. 5 Summary of approaches using geo-information.

Authors	**Algorithms**	**Mechanism**	**Advantages**	**Disadvantages**
Castellucci a et al. (2009)	Constraint-based geolocation technique	Determines the physical location of the FF mothership on the basis of network measurements	Can localize with a mean error distance below 100 km	-Requires extensive resources to set up -Less precise and less complete
Buhariwal a (2011)	Constraint-based geolocation technique	Determines the physical location of the FF mothership on the basis of network measurements	Decreases the overhead of requesting content servers	Inaccurate rate (300 km)
Stalmans et al. (2012)	Time zone, UTM, MGRS	Identify fast flux domains on the sole basis of the	Only a small percentage of FPs	Classifier is affected by a large

		geographic locations of C2 servers		amount of whitespace
Stornig (2013)	Moran's I of spatial autocorrelation and spatial service distance	Utilizes methods of geo-information and spatial statistics	-Lightweight system -Avoids FPs	Could be misclassified by botmasters

The problem with active detection-based approaches is that they deal with minimal DNS traffic traces, which correspond to non-legitimate domain names in most cases. According to the nature of active approaches that mostly deal with malicious domains, they are obviously unable to detect unknown zero-day domains.

2.2.1.3 Host-based Real-time Approaches

The previous methods involve passive and active approaches, which presented many detection techniques to detect malicious fast flux botnet domains and FFSNs. Fast flux detection requires a fast and accurate approach to identify malicious domains before they change their IP addresses. Thus, a new era of real-time approaches have been developed to increase the power of detection techniques. The main idea behind employing real-time approaches is to reduce the time needed to detect attacks to real-time processing.

A) Spatial Information-based Approaches

Caglayan, Toothaker, Drapeau, Burke, and Eaton (2009) were the first to conduct a related empirical study. The authors presented a fast flux monitor (FFM) that could detect and classify FFSNs in real time within minutes. The FFM comprises active and passive DNS monitors, which reduce the long-term observation of FFSNs. Using active and passive monitoring can reduce observation duration, but the system still requires a few additional minutes. Obtaining extra information from a data center helps classify botnet domain names.

Huang et al. (2010) proposed a real-time system called spatial snapshot fast flux detection (SSFD). SSFD detects FFSNs by extracting the IP addresses of the hosts (agents) from the DNS responses and determining the geographical traffic patterns of these agents in a geographic coordinate system. Two spatial measures were used: spatial distribution estimation and spatial service relationship evaluation. A Bayesian network classifier was also employed to distinguish FFSNs from benign networks. The experimental results indicated that SSFD is effective (less than 0.5 s) and yields lower FP rates than flux score detection systems through their data sets. However, SSFD suffers from a single IP problem and missing geographical information problem, which may cause the system to malfunction. The experiments verify that the detection accuracy is 62% (H.-T. Lin et al., 2013).

A Bayesian network classifier algorithm classifier was proposed by Horng-Tzer, Ching-Hao, Kuo-Ping, and Hahn-Ming (2012) to detect FFSNs in real time. The authors believed that the grid distribution of the localized spatial-locating capability is ideal to depict the spatial relationship between the resolutions of IP addresses. To enhance the localized geo-locational characteristics, the proposed system incorporated ASNs, localized spatial geo-location detection (LSGD) system, and DNS to achieve the identification of potential FFSNs. The authors believe that the detection capability of the LSGD system is better than spatial or temporal detection approaches. The LSGD system exhibits a lower FP rate than the spatial snapshot system in real-time detection, which is completed within seconds. However, the highest FP rates are caused by CDNs, which have a similar localized spatial distribution signature that affects accuracy. Table 2. 6 shows the summary of spatial informational approaches.

Table 2. 6 Summary of spatial informational real-time approaches.

Authors	**Algorithms**	**Mechanism**	**Advantages**	**Disadvantages**
Caglayan et al. (2009)	Bayesian belief network	-Bayesian classifier employs multiple active and passive DNS sensors	Reduces the observation period	-Long time -Data center help is needed

		-Generates a probabilistic assessment of the existence of a FFSNs		
Huang et al. (2010)	Bayesian network classifier and K2 algorithm	Determines the geographic traffic patterns of hosts and maps the IP address of a DNS response in a geographic coordinate system	Lower FP rate than flux score-based detection	-Single IP problem -Missing value problem
Horng-Tzer et al. (2012)	Bayesian network classifier and K2 algorithm	Propose LSGD system for identifying FFSNs in real time	Better detection capability than spatial or temporal detection approaches	-Misled by CDN service sites

B) Behaviour-based Approaches

Many researchers have studied the behavior of the changes in fast flux domains. Caglayan, Toothaker, Drapaeau, Burke, and Eaton (2010) modeled the behavior pattern of FF botnets on the basis of DNS resource records using a Bayesian classifier. The authors determined that botnets exhibit common characteristics and form clusters according to botnet size, growth, and operations. Their findings show that a majority of fast flux botnets operate in at least five countries and between 20 and 40 countries on average. Unfortunately, their approach is misled by benign servers, such as CDNs, thus resulting in a high number of FPs (Caglayan et al., 2010).

B. Yu, Smith, and Threefoot (2014) addressed the behaviour of fluxed domain changes and proposed a novel time series model on the basis of carefully selected features. Their model uses

network security and a semi-supervised training approach to overcome and identify difficulties in known supervised machine learning approaches. A horizontal scalable online system was proposed to deal with the large amount of data that passes through a network in real deployment. Their system can identify flux domains despite the presence of long TTLs or a limited number of mapped IP addresses. Actual latency is determined by an online system (10 min) given a domain name, whereas most active threats can be detected in less than 10 min. Their approach does not address the FN rates in the evaluation of the results. Table 2. 7 shows the behavior-based approaches.

Table 2. 7 Summary of behavior-based approaches.

Authors	**Algorithms**	**Mechanism**	**Advantages**	**Disadvantages**
Caglayan et al. (2010)	Bayesian classifier	Modeling the behavioral patterns of fast flux botnets using DNS records	-Botnets operate in 20 to 40 countries - < 250 ASNs	**- Misled by CDNs** -High number of FPs
B. Yu et al. (2014)	Time-series model	-A time-series model –A horizontally scalable online system	Captures fast flux domains	-Long detection time. -FN rate is not considered

C) Machine Learning-based Approaches

Qassrawi and Zhang (2012) used the algorithm of an alternative decision tree (Gothai & Balasubramanie, 2012) to determine whether a domain is an FF domain. Figure 2. 10 shows that only one DNS response resource record is needed to achieve fast detection in real time. Previous studies show that DNS information is insufficient to detect FF botnets (Martinez-Bea et al., 2013).

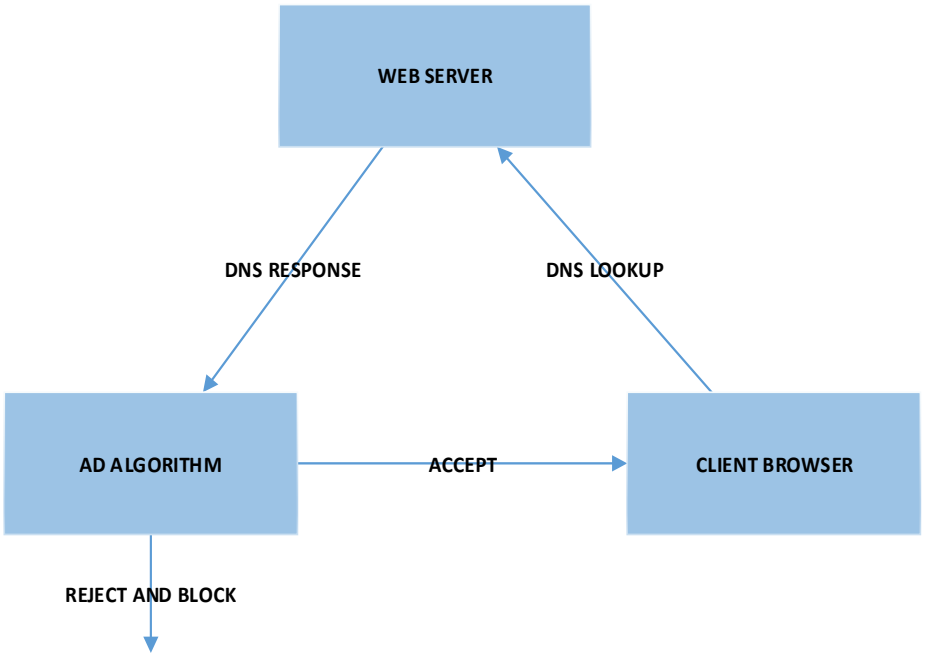


Figure 2. 10 Alternative decision tree detection approach(Qassrawi & Zhang, 2012)

Unlike Qassrawi, Hsu et al. (C.-H. Hsu et al., 2010) proposed a real-time system to measure the delay for HTTP responses by relaying user requests from an FF-agent to back-end servers. Thus, a long delay means that a host (FF-agent) relayed a request to another server. The authors proposed this real-time system to reduce detection time to a few seconds without affecting detection accuracy (96% with FP and FN rates below 5%). The authors utilized a classification based on supervised learning using SVM trained on six features. The delays in the relaying request are counted because of the limited power and bandwidth of the relaying hosts (FF-agents). However, extracting the six features from this volume is time consuming. Thus, keeping the detection time within the real-time range is difficult. The proposed system cannot effectively detect fast flux domains with long TTLs, and the detection accuracy is 67% (H.-T. Lin et al., 2013). The proposed detection system cannot detect zero-day domains.

SVM was built by McGrath to detect fluxed phishing domains (D. Kevin McGrath, 2009/09/01). The classifier was trained on the basis of the features extracted from the DNS responses, such as the number of IP addresses related to one domain, ASNs, number of different prefixes, and number of countries of an IP address. The main limitation of the classifier, based

on the nature of its features, is that it can be misled by botmasters. The previous classifier proposed in (C.-H. Hsu et al., 2010) can be misled by a benign server, such as a CDN or RRDNS. A new SVM classifier proposed by Martinez et al. (Martinez-Bea et al., 2013) was trained on real features from both domains and bots. Combining the two feature sets from the two previous approaches (D. Kevin McGrath, 2009/09/01; C.-H. Hsu et al., 2010) increased the TP and TN rates and decreased the FP and FN rates. Unfortunately, the author stated that the proposed method for detecting fast flux domains may still be evaded theoretically and that the proposed detection system cannot detect zero-day domains.

The genetic based real-time approach for FFSN detection (GRADE) was proposed by H.-T. Lin et al. (2013). The authors assumed that fast flux bots are distributed arbitrarily in a multitude geographical locations. Thus, the distances between bots (FF-agent) and users differ. The fast flux domains would result in significant differences in the round trip time between the user and the agents. The GRADE system architecture is depicted in Figure 2. 11. GRADE can more effectively detect FFSNs (within a few seconds) than flux scores and is more accurate (98%) than fast flux bot detection and SSFD. However, GRADE suffers from the single IP problem, in which only one point in the geographic coordination system may cause GRADE to malfunction. Table 2. 8 summarizes real-time machine learning approaches.

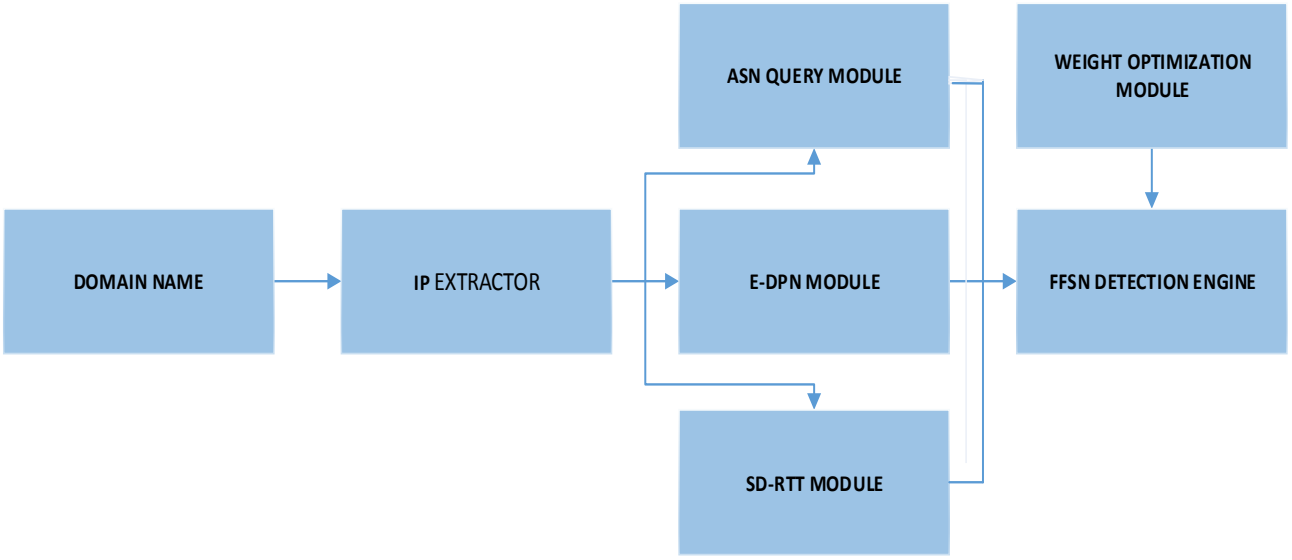


Figure 2. 11 GRADE system architecture(H.-T. Lin et al., 2013)

Table 2. 8 Summary of real-time learning approaches.

Authors	**Algorithms**	**Mechanism**	**Advantages**	**Disadvantages**
Qassrawi and Zhang (2012)	Alternative decision tree	One DNS response RR is needed to achieve FF detection in real time	One DNS response RR in needed	Insufficient features to conduct classification
C.-H. Hsu et al. (2010)	Linear SVM algorithm	Observes longer delays for HTTP responses as a result of relaying the requests via fast flux agents	-Real time -Robust -Lightweight	**-Long detection time** **-Cannot detect long TTL domains** -Unable to detect zero-day domains
Martinez-Bea et al. (2013)	Linear SVM algorithm	Builds an SVM classifier trained via real features extracted from domains and bots to differentiate malicious FFNs	-Increased TP and TN -Reduced FP and FN	**Unable to detect zero-day domains**
H.-T. Lin et al. (2013)	Genetic algorithm	The distances between clients and flux bots varies significantly	Outperforms other systems, such as flux score, FFBD, and SSFD	Single IP problem

The proposed real-time approaches can detect malicious domain names and malicious FFSNs in most cases. However, the above-mentioned techniques have certain limitations, which cast doubt on their results (accuracy, TP, TN, FP, and FN), as some methods have high percentage

of false positive and others have low detection accuracy. We still lack a stable technique that can detect malicious domains, particularly zero-day domains, in an acceptable period of time with high detection accuracy.

2.2.2 Router-based Detection Methods

Many researchers have used different sets of information from network traffic to solve several network problems generally and particularly for the fast-flux botnet problem. Network traffic comprises both DNS traffic and non-DNS traffic. Recent studies (Al-Duwairi & Al-Hammouri, 2014; Paul, Tyagi, Manoj, & Thanudas, 2014) did not rely on DNS data traffic. Al-Duwairi and Al-Hammouri (2014) compared incoming and outgoing data traffic at a leaf router of stub networks to find matches between incoming and outgoing SYN packets. This online approach efficiently detects malicious fast flux agents within stub networks. However, installing the system on all the leaf routers of stub networks is difficult to achieve (scalability problem), and the utilized data traffic traces do not have fast flux traffic (Al-Duwairi & Al-Hammouri, 2014).

A previous work (Paul et al., 2014) aimed to cluster similar packets in data traffic from both router sides assuming that the C&C servers had to change their IP addresses automatically. The approach assembles all packets, as shown in Figure 2. 12, between the C&C server and the host for analysis and obtains the malicious pattern in each cluster. The detection accuracy of this approach to malicious traffic is 95.8%, and its low FP rate is 1.6% in the worst case. However, the approach suffers from a scalability problem. Thus, when data traffic is insufficient, the malicious packet sensitivity decreases. Table 2. 9summarizes the router-based approaches.

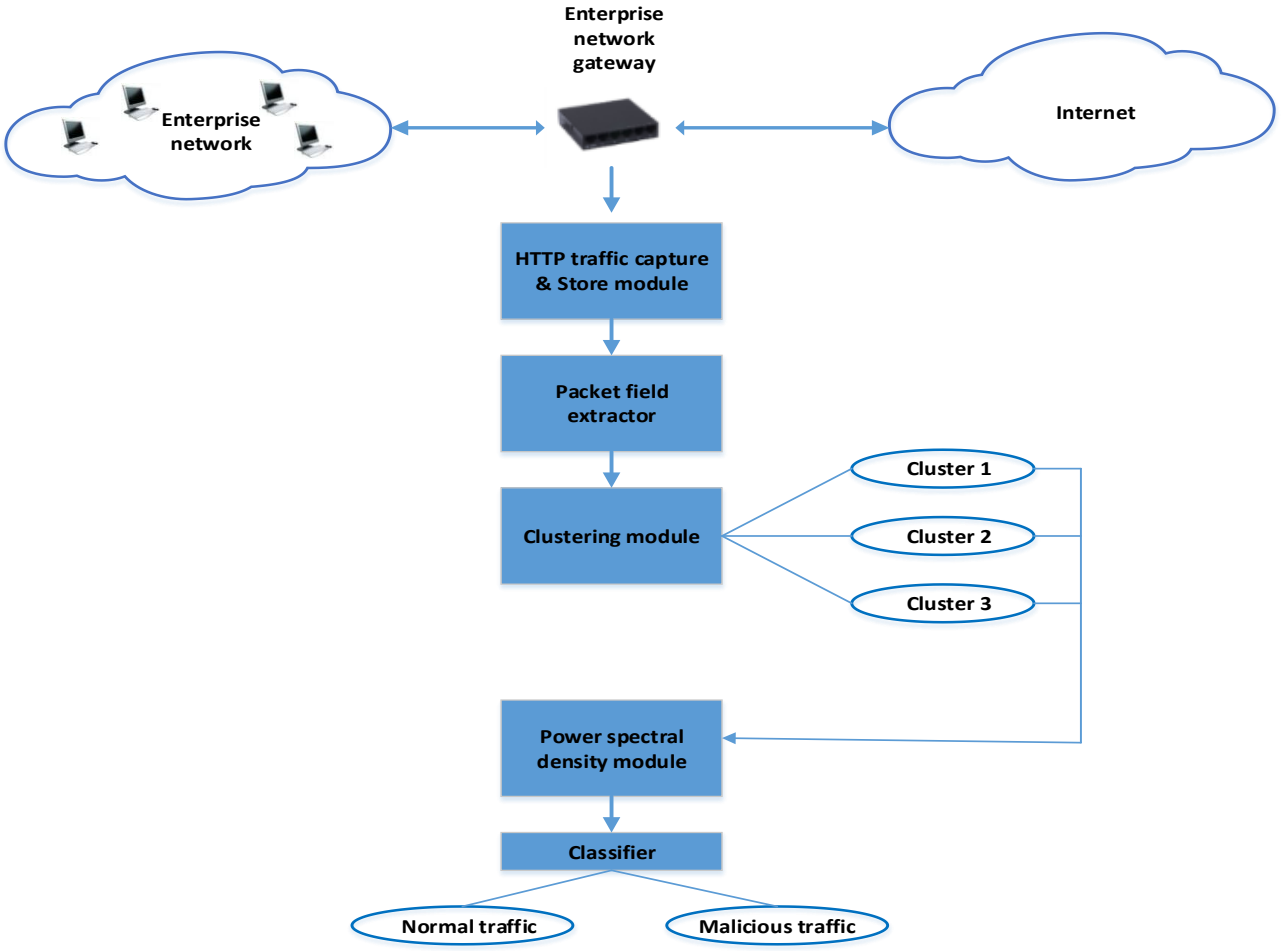


Figure 2. 12 System architecture of the clustering detection method (Paul et al., 2014)

Table 2. 9 Summary of router-based approaches.

Author-s	Algorithms	Mechanism	Advantages	Disadvantages
Al-Duwairi and Al-	FF-watch algorithm	Correlates incoming TCP connection requests to flux agents within a stub network with outgoing TCP connection	Eliminates the need for large DNS traffic	-Old dataset which may not contain FF traces -Scalability problem

Hammouri (2014)		requests from the same agents to the point-of-sale website		
Paul et al. (2014)	PSD value used as classifier	Computes the power spectral density (PSD) for each cluster and feeds it to the classifier, which examines the PSD data for significant peaks	Detects traditional HTTP and fast flux botnets	-Scalability problem -Malicious packet detection sensitivity problem

Table 2.9 clearly indicates that constructing router-based systems to detect fast flux botnets may produce acceptable results for the authors. However, the speed and the large amount of data passing through the router cause three main problems to build systems: high false rates based on the concept of a fast detection of FF botnets, memory problems (databases) due to handling large traffic data flows, and a scalability problem. Therefore, detecting fast flux botnets and particularly zero-day domains at this part of the network is ineffective.

2.2.3 DNS-based Detection Methods

Researchers studied DNS data traffic in their country of origin. Thus, their work focused on monitoring and analyzing DNS data traffic and detecting malicious activities, such as fast flux botnets. Some researchers employed passive, active, and real-time approaches, as presented in the following subsections.

2.2.3.1 Passive Approaches

Researchers monitored DNS servers and analyzed data traffic passively to detect malicious activities. Gržnić, Perhoč, Marić, Vlašić, and Kulcsar (2014) presented a detection system called CROFlux that detects fast flux domains relying on a passive DNS replication method. Their system aims to reduce FP rates and detect unknown fast flux domains with flux characteristics, which are usually used to share malware. Thus, the approach avoids the reporting of legitimate domains with similar characteristics. The proposed system suffers from a design problem because it does not utilize active DNS requests to feed the system and many IP addresses can

enhance fast flux detection (Gržnić et al., 2014). The proposed system cannot detect zero-day fast flux domains because the classification process depends on the comparison of the number of malicious domains in the candidate fast flux cluster with predefined fixed malicious domains.

A scalable and fast approach proposed by Kwon, Lee, Lee, and Perrig (2016) detects fast flux botnets on the basis of large-scale DNS traffic. This approach analyzes the collected large-scale DNS data traffic to extract malicious behaviors. A signal processing technique, namely, PSD analysis, is leveraged to determine the main frequencies from the periodic DNS queries initiated by botnets. Their system detection accuracy is 95%, given its detection of 23 unknown and 26 known botnet groups with 0.1% FP. However, the proposed method relies on the number of hosts. Thus, increasing the number of hosts should decrease speed and detection efficiency. A threshold number should be assigned according to the circumstances of DNS servers; such threshold number differs for all DNS servers (Kwon et al., 2016).

Some decision tree algorithm versions have been used for many detection techniques, such as the system proposed by Perdisci, Corona, Dagon, and Lee (2009). This system passively collects recursive DNS queries and responses by deploying multiple sensors in front of RDNS servers in two ISP networks. Perdisci analyzed the extracted features to detect malicious FFSNs using C4.5 decision tree. Their experiments showed that they accurately distinguished malicious and legitimate FFSNs. They used a statistically supervised learning approach to build a service classifier. Thus, this classifier cannot detect malicious zero-day flux services.

Similarly, Perdisci, Corona, and Giacinto (2012) proposed a novel passive DNS system called FluxBuster using C4.5 decision tree as a classifier; this system analyzes DNS traffic for malicious FFN detection and blocking. Their approach gathers DNS traffic generated from hundreds of RDNSs, which are scattered in many networks around the world. A large-scale analysis is carried out on the basis of the resultant traffic. Thus, FluxBuster can detect unknown FFNs before they are reported in a public blacklist. However, the detection system waits for a user to click on a domain name to initiate a request and detect a domain. Furthermore, more IP addresses are needed to set the threshold value of their classifier.

An anomaly-based technique using a decision tree with AdaBoost algorithm was proposed in a previous work (Vu Hong, 2012). This approach depends on the passive analyses of extracted

DNS data traffic to detect fast flux botnets. Two graphs were constructed, the lookup graph and the failure graph, from the extracted DNS traffic. The resulting graphs were distributed into clusters, as depicted in Figure 2. 13. These clusters exhibited a strong correlation between traffic elements (domain, host, and IP addresses). The related features of DNS traffic were extracted from these clusters to feed the classification module in the detection system and identify the existence of a fast flux botnet. The authors believed that they succeeded in detecting a fast flux botnet from traffic analysis. However, the system produces FP rates when the number of domain names in a malicious subgraph was small and produced FN rates when a benign subgraph included a large number of random-looking domain names. The malicious characteristics exhibited by the subgraph were not sufficiently distinctive for the technique to obtain. Table 2. 10 summarizes passive DNS-based detection approaches.

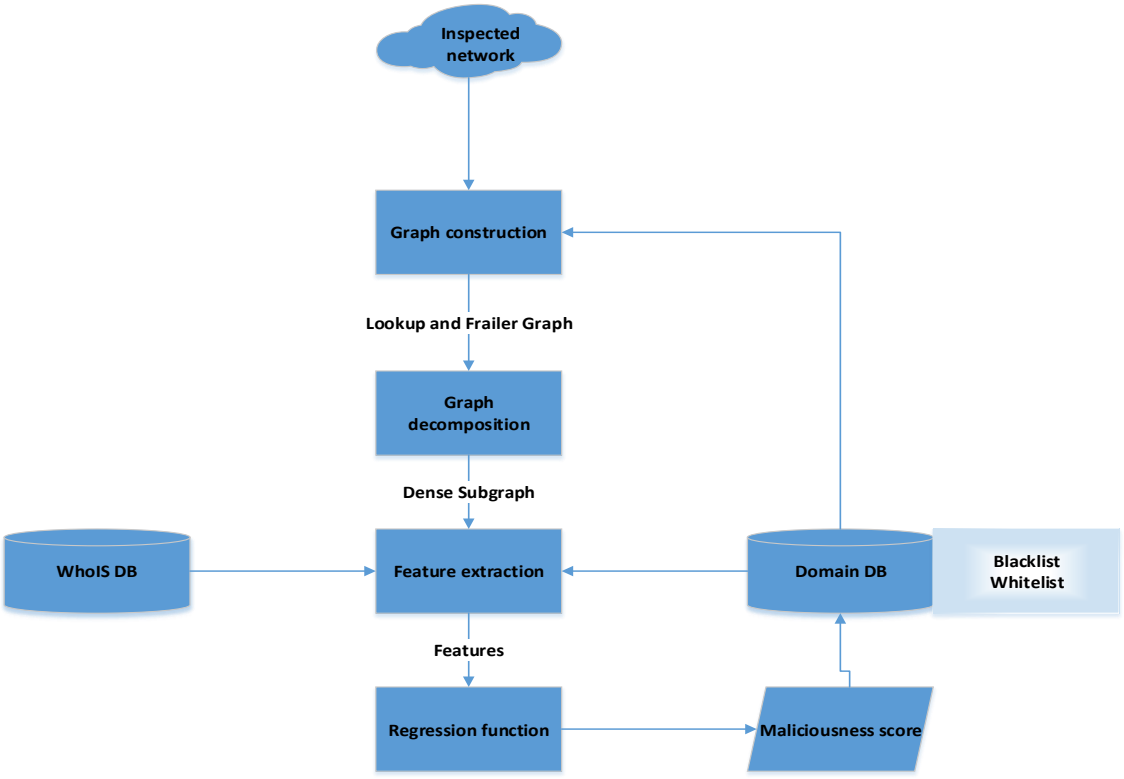


Figure 2. 13 Analysis procedure of an anomaly-based technique using a decision tree with AdaBoost algorithm (Vu Hong, 2012)

Table 2. 10 Summary of passive DNS-based detection approaches.

Authors	Algorithms	Mechanism	Advantages	Disadvantages
Gržnić et al. (2014)	Publicly available and private malware lists	Relies on the passive DNS replication method to detect suspicious fast flux domains	Reduces FP	-Design problem -Unable to detect zero-day fast flux domains
Kwon et al. (2016)	PSD	Leverages a signal processing technique to discover the major frequencies of periodic DNS queries of botnets	Detection of 23 unknown and 26 known botnet groups	-Increases in the number of hosts decreases the efficiency -Fixed threshold for all DNS servers
Roberto Perdisci et al. (2009)	C4.5 decision tree	Detects malicious flux service networks through passive analysis of recursive DNS traces	Accurate classification	Cannot detect zero-day malicious flux services
Roberto Perdisci et al. (2012)	C4.5 decision tree	A passive DNS traffic analysis system for detecting and tracking malicious flux networks	Detects unknown flux networks before blacklisting	-wait user click -The threshold needs sufficient IP addresses to be set
Vu Hong (2012)	Decision tree with AdaBoost algorithm	-Constructs a lookup graph and a failure graph from captured DNS traffic	Helps detect botnets through traffic analysis	-Produces FN and FP -Insufficient distinctive features

		-Decomposes these graphs into clusters with a strong correlation between their domains, hosts, and IP addresses		

2.2.3.2 Active Approaches

An active approach (Zhou, Leckie, & Karunasekera, 2009) adopts a collaborative detection system based on a decentralized correlation model called large-scale intrusion detection to detect fast flux phishing domains by analyzing the relationship between the number of IP addresses and DNS requests from different networks.

Figure 2. 14 shows the combination of different DNS server responses to quantify the probable time to be saved. The results indicated that combining evidence from multiple DNS servers would speed up the process of fast flux detection. No significant time was saved, which leads to fast detection of fast flux domains. Table 2. 11 summarizes the active approaches.

Table 2. 11 Summary of active approaches.

Authors	**Algorithms**	**Mechanism**	**Advantages**	**Disadvantages**
Zhou et al. (2009)	Decentralized correlation model called LarSID	Correlation of multiple responses of DNS servers to increase detection time	Reduces query time up to 30%	Detection time is long

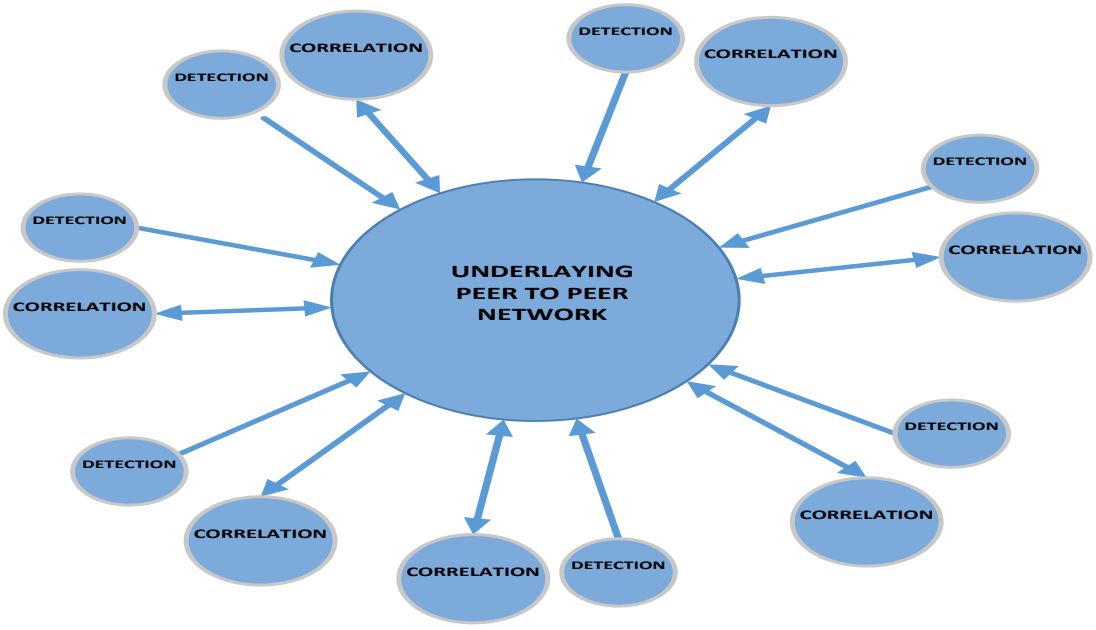


Figure 2. 14 LarSID architecture (Zhou et al., 2009)

2.2.3.3 Real-time Approaches

A real-time approach (Futai, Siyu, & Weixiong, 2013) was used to develop a fast flux botnet detection method. This approach employs the J48 decision tree algorithm as a classifier in a hybrid system, which combines real-time detection and long-term monitoring, as depicted in Figure 2. 15. Their approach can achieve a higher real-time detection rate compared with flux score-based methods. Still, the proposed approach cannot detect fast flux domains with high TTL values. Table 2. 12 summarizes the real-time approaches.

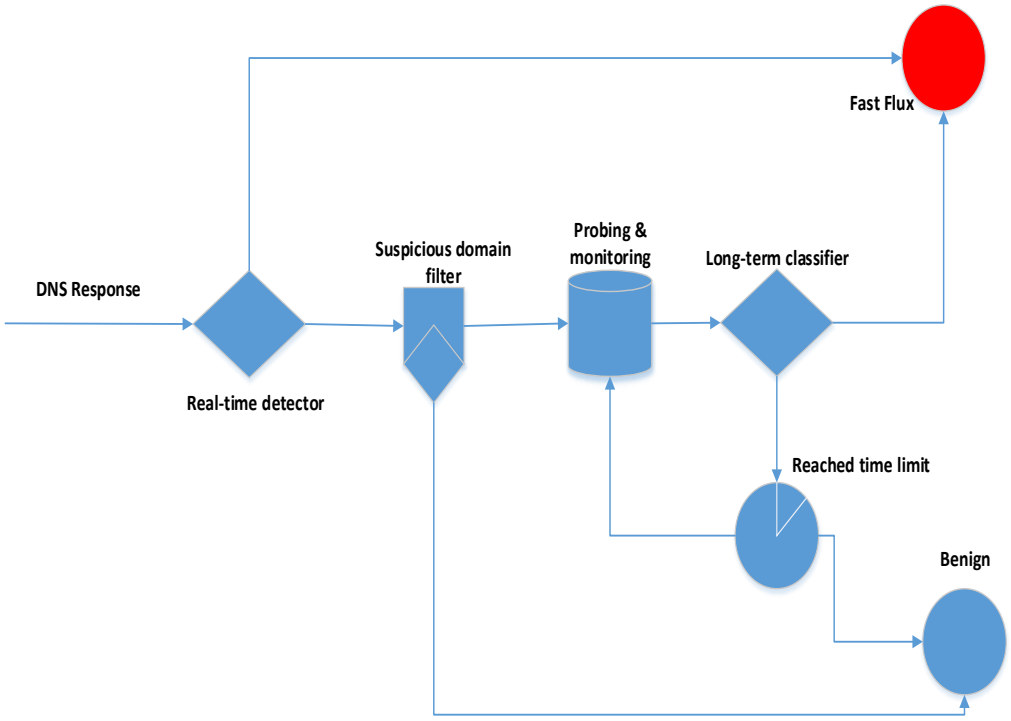


Figure 2. 15 Hybrid detection system(Futai et al., 2013)

Table 2. 12 Summary of real-time approaches.

Authors	**Algorithms**	**Mechanism**	**Advantages**	**Disadvantages**
Futai et al. (2013)	J48 decision tree	Combines real-time detection and long-term monitoring	Higher detection rate compared with flux score-based algorithms	-Cannot detect FF domain with high TTL value -Cannot detect unknown FFSNs

The detection systems initiated over a DNS server do not exhibit network time delays. Many researchers determined that systems that depend on DNS features cannot provide an accurate detection rate for fast flux domains (Martinez-Bea et al., 2013).

The main problem of fast flux botnet detection methods is detecting the evasion detection mechanism before the attack is initiated to support botnet malicious activities. This is

particularly true when detecting zero-day domains without any prior knowledge about the incoming domain name, which serves malicious websites, C2 servers, and motherships. At the same time, detection accuracy and low detection error rates are monitored. On the basis of the developing strategies of attackers, the detection system should develop new systems that are long-lasting and adaptive to allow the future modification of their functions. Detection systems should continuously learn by analyzing new system inputs as new data instead of training with old data.

2.2.4 Hydra Flux Service Network

Hydra fast flux networks and SQL injection attacks are the main advanced features of Asprox botnets. Al-Bataineh and White (2012) studied the design and structure of Asprox botnets, in which communication protocols are used to download malicious codes, propagate malicious codes, and employ hydra FFSNs. The authors mentioned that SQL injection attacks are responsible for the recruitment of new bots and social engineering ruses to spread malware binaries. Hydra FFSNs prevent the disruption of the communication channel between bots and the C&C server.

The crucial aspect of the hydra fast flux is the possibility of bots contacting other C&C servers when the original C&C server is taken down. Hydra is an advanced double fast flux that refluxes the name server and the host IP addresses, making the prevention of massive disruption impossible.

2.2.6 Dynamic evolving Spiking Neural Network (DeSNN)

Spiking Neural Network (SNN) is the third generation of neural network that adds the time element to the network. eSNN is an improvement on spiking neural network, as well as an extension of the ECOS models, employing integrate and fire neuron (IF) and Rank-Order learning (RO). A new improvement is a Dynamic evolving Spiking Neural Network (DeSNN) (Kasabov et al., 2013). The RO learning is built on the theory where the most significant information of an input pattern is enclosed in earlier incoming spikes, the priority of the inputs is comes from the incoming spikes order at the input synapses for a specific pattern. This is a

simulation of the biological system as well as an important base for some spatio-temporal hard cases.

One of the main problems facing the fast flux Botnet detection methods is how to detect such an evasion detection mechanism before the attack begins, to support botnet malicious activities. This is especially so when detecting zero-day FFSNs without any prior knowledge about the incoming domain name serving the malicious website/C2 server/mothership, at the same time as keeping track of the detection accuracy and low detection error rates. In previous related work done by Almomani et al. to detect unknown zero-day phishing emails in online mode(Deeb Al-Mo et al., 2011), they classified phishing emails based on ECoS, which gives a promising platform for phishing detection. ECOS proved its adaptability in classification of ham and phishing emails in online mode based on a one pass algorithm for increased speed (Deeb Al-Mo et al., 2011), which accesses the data only once from the memory to create rules. A limitation of the ECoS algorithm used in that paper is that it classified phishing emails as traditional connectionist algorithms which need a careful assignment of their parameters.

According to Demertzis and Iliadis (2015), eSNN is used to detect DGA domain names. The authors proposed a fast evolving Smart URL Filter in Zone-based Policy Firewall. Their work promised improvement on zone-based policy, but the inclusion of self-modified parameter values are still necessary to get more efficiency.

Dynamic evolving spiking neural network is used as an output classifier under the NeuCube platform (Alvi, Pears, & Kasabov, 2017), as the DeSNN is achieved outstanding success in spatio-temporal classification problem in many areas. Also, the DeSNN algorithm was used in (Doborjeh & Kasabov, 2016) to perform the output classifier with NeuCube platform in supervised learning mode.

(Kasabov et al., 2013) introduced the dynamic evolving spiking neural network that utilized the rank-order learning and SDSP spike-time learning in unsupervised, supervised, or semi-supervised modes. The results were performed a high level of accuracy and speed comparing with other SNN algorithms. However, the algorithm suffer from many parameters to be set before implement the algorithm.

According to (Doborjeh, Capecci, & Kasabov, 2014), a DeSNN algorithm used as classifier in NeuCube platform, where the NeuCube model was used for FMRI data learning. Nevertheless, the model is extremely influenced by its parameters values. Alauthman and Almomani (2017) used the DeSNN as a classifier over the NeuCube platform to detect the spam email.

As a result, all the three solution scopes were discussed, the advantages and disadvantages, current work tends to be implemented as a root at the end host-based part of the network, Also, this system can be implemented at the local DNS server in order to work as defender in case of threats and risks. Fast flux domains act in an online mode to keep up their duties as a response to the mothership orders, so the need for online detection system became necessary to try to shut down such threats. The proposed system is expected to detect FFDN in online mode with high detection accuracy and low false positive and negative rates respectively. It is also expected to have a high level of performance depending on using one-pass algorithm and chosen proper feature set, also the proposed system should work for a lifetime with low memory usage.

2.3 Related Work

Holz et al. (2008) were the first who proposed a system that measures and detects a FFSN on the basis of the calculated flux score. Their proposed system takes malicious domains from spam emails and then uses the Dig tool to generate DNS lookups and reverse DNS lookups and thereby obtain necessary information about a feature set, for the sake of space all the features of all the related works displayed in Table 2.13. Thus, the flux score calculation is fed for use later in distinguishing between malicious FFSNs and legitimate ones. Their results showed that the proposed system achieves a detection accuracy of 99.98%.

Some decision tree algorithm versions have been used for many detection techniques, such as the system proposed by Roberto Perdisci et al. (2009), which passively collects recursive DNS queries and responses by deploying multiple sensors in front of RDNS servers in two ISP networks. They analyzed the extracted features to detect malicious FFSNs using a C4.5 decision tree. Their experiments showed that they accurately distinguished malicious and legitimate FFSNs.

In the naïve Bayes classifier used by Passerini et al. (2008), all malicious domains are collected from spam emails. Their detection and monitoring "FluXOR" system relies on the idea of a host being a victim to such scams. The system begins to send requests and gathers the feature set information to feed the naïve Bayes classifier. FluXOR reduces the time of detection to 1–3 h, which is still relatively long.

Zhao and Traore (2012) proposed another method (REPTree) for botnet detection using a decision tree with reduced error pruning. This type of machine learning decision tree is used to classify and identify malicious FFSNs by defining and computing some of the network metrics captured from network flows.

The support vector machine (SVM) was used by X. Yu et al. (2012) to detect fast flux botnets by analyzing the patterns of DNS queries from FF botnets. They extracted six features to build the weighted SVM classifier for use in distinguishing legitimate and FF botnet domains. They noted that using SVM to identify fast flux botnets is effective and provides a satisfactory detection accuracy.

Celik and Oktug (2013) used the C4.5 decision tree algorithm to evaluate various DNS feature sets and put forward a detection approach, which is a high-dimensional feature vector with various features, including timing network, spatial, and NS and DNS response information. C4.5 evaluates each feature set of previous vectors and decides which one is the best feature vector on the basis of detection accuracy. Combining all features together provides a detection accuracy of 98.9%.

The above researches used different features as stated in Table 2.13, Holz et al. (2008); (Passerini et al., 2008; Perdisci et al., 2009; X. Yu et al., 2012; Zhao & Traore, 2012) provided different methods to detect the fast flux domains and FFSN. All except Holz used supervised learning approaches to build their classifiers. Thus, the classifiers cannot detect malicious zero-day flux domains. The coefficients used in the score calculation (Holz et al., 2008) require modification to ensure the highest possible accuracy of the detection system.

Table 2. 13 List of the features used in previous works

Related work papers	**Features**
Holz et al. (2008)	Number of A records, number of autonomous system numbers (ASNs), and number of NS.
Perdisci et al. (2009)	Number of resolved IPs, Number of domains, Avg TTL per domain, Network prefix diversity, Number of domains per network, IP Growth Ratio, Autonomous System (AS) diversity, BGP prefix diversity, Organization diversity, Country Code diversity, Dynamic IP ratio, and Average Uptime Index.
Passerini et al. (2008)	Domain age, Domain registrar, Number of distinct DNS records of type "A, Time-to-live of DNS resource records, Number of distinct networks, Number of distinct, ASN, , Number of distinct resolved qualified domain names , Number of distinct assigned network names, and Number of distinct organizations.
Chen et al. (2014)	Number of A records in answer section, number of distinct ASNs for all A records, number of A records for NSs, number of distinct ASNs for all NSs, TTL of A records in answer section, and TTL of A records for domain's NSs.
Chen et al. (2011)	Number of unique ASNs, The number of Cname, number of name server (NS), number of different IP addresses, TTL, and Rate flux.
Yu et al. (2012)	Domain age, number of IP addresses of a distinct DNS A records, TTL, IP distribution, ASN, and organizational distribution.
Celik and Oktug (2013)	Various features including: timing network, spatial, NS and DNS response information.
Zhao and Traore (2012)	TTL, Number of unique A records, IP in the same Networks, IP Geolocation, IP Geolocation, and Domain lifetime,

Zhao and Jin (2015)	Intrinsic Characteristics of Fast-flux Net, Intrinsic Features of DNS Analysis, and Intrinsic Features of Process of Visiting FFSN Domains
Vu Hong (2012)	Median domain's life time, Median IP's life time, Domain/IP ratio, Median number of distinct domains, Median number of distinct IPs, Median number of IPs per query, Median maximum TTL value, Network diversity of return IPs, Dominant domain ratio, Dominant host ratio, Lexical features on domain labels, Query pattern, IP overlap, and IP growth rate.

Chen et al. (2014) proposed Bayesian probability theory to distinguish between benign and malicious domains using dissimilar ASNs, reverse DNS lookups, and domain registration time features. They aimed to detect a fast flux website on the basis of its fluxed characteristics. The result of this proposed system presents its ability to identify possible threats.

Chen et al. (2011) used the k-nearest neighbor (KNN) and random forest (RF) as sampling techniques to solve the imbalanced problem, with respect to FFSN detection. In addition, they proposed a sampling technique that is combined with feature extraction from datasets for use in fast flux detection. The result showed that the TTL is an important feature to the classification of the proposed technique.

The classification and regression tree algorithm is used in the method proposed by Zhao and Jin (2015). This method uses a small dataset to quickly distinguish legitimate and malicious FFSNs. This method is mainly based on FFSN domains, DNS, and the process of HTTP visiting. Yoshioka and Matsumoto used distinct mapping of features (Pa et al., 2015). The classification process needs less than a few days, and the detection accuracy is 90%.

An anomaly-based technique using a decision tree with AdaBoost algorithm was proposed by Vu Hong (2012). This approach depends on the passive analyses of extracted DNS data traffic to detect fast flux botnets. Two graphs were constructed, namely, the lookup and failure graphs, from the extracted DNS traffic. The resulting graphs were distributed into clusters. These clusters exhibited a strong correlation between traffic elements (domain, host, and IP addresses).

The related features of DNS traffic were extracted from these clusters to feed the classification module in the detection system and identify the existence of a fast flux botnet.

Some of the related work showed low detection accuracy based on the feature set chosen, such as in (Celik & Oktug, 2013; Chen et al., 2014; Chen et al., 2011; Vu Hong, 2012), and others need long time to gather sufficient information for these features, such as in (Passerini et al., 2008; Zhao & Jin, 2015).

The two closest works to the current study are the works developed by Celik and Oktug (2013) and Lin et al. (2013).

According to H.-T. Lin et al. (2013) a genetic approach was proposed as a real-time detection solution of the fast flux domains problem. This method suggested a two-detection feature to classify the benign and the flux domains. Firstly, the entropy of the domain name (E-DPN) of the preceding node of the flux node (flux-agent). By using the trace route of all the returned IPs from the DNS response. Of course, if the E-DPN is high then, most probably, the domain is classified as benign, otherwise it is classified as fluxed. Secondly, the Standard Deviation of Round Trip Time (SD-RTT) between the user and all the return IPs of the flux-agents, assuming that the scatter flux-agent is going to produce a high value of the SD-RTT. This spatial feature takes the number of different ASNs and number of IPs return in single DNS response in their calculations. Unfortunately, this two detection features was evaded by the botmaster, as it controls the returned list of IPS that the user receives. The returned list could have IPs in the same ASN or adjacent to the user ASN, so the above measures can inaccurately classify the benign and flux domains. On the other hand, botmaster may return a list that contains just a single IP address, which leads to ineffective detection of the domains (F.-H. Hsu et al., 2014; Otgonbold, 2014). Although genetic algorithms provide good accuracy (as stated in their paper), their results could be affected by returning a list of IP addresses belonging to the same AS. According to current implementations the overall accuracy of the linear classifier was (95.37 %). Also, the linear decision function used as the classifier needs to estimate the categorizer of the linear function, so if the estimation is good then the linear function will work properly, otherwise the classification process will contain significant errors. (Chahal & Khurana, 2016).

On the other hand, the second compared algorithm was the C4.5 as presented in (Celik & Oktug, 2013). Several feature sets were examined to detect fast flux network. Such feature sets consist of timing based, spatial based, network based, domain based, and DNS answer-based feature sets. As mentioned in the literature the data set was small and the accuracy of the experiment was high. Also, when all features are involved in the experiment the prediction results become insensitive to two features (timing and domain based feature sets) (Otgonbold, 2014). Besides, as C4.5 algorithm is considered as a supervised learning algorithm, it could not be used to discover the unknown attacks, especially the zero-day fast flux domains. Moreover, according to our implementation the accuracy was not as high as stated in their paper; rather it was 93.38%. When this result and the previous linear results were compared to the current proposed ADeSNN, obviously the proposed approach overcame the two methods. Discussions in section in chapters four and five showed other accuracy measures indicating the results.

2.4 Conclusion

In this chapter, a wide-range of the literature on fast flux detection approaches was studied to choose the most suitable techniques and methods that were used for detecting fast flux domains. The techniques and methods were explored based on various aspects, such as the feature sets used to classify FFD and classification accuracy. The approach of this study includes scope of solution, testing criteria for classification, and mode of detection, whether online or offline. Also included is a theoretical background for evaluating the selected approaches.

Besides, the literature showed some approaches that have used the dynamic evolving neural network in different areas, most of them were implemented as a classifiers under the NeuCube platform, its been clear that the DeSNN algorithm has many parameters that has to be set before run the algorithm. Based on this the current research is going to address this issue as one of its problem space.

On the other hand, the DeSNN algorithm still need to be improved to best enhanced the classification accuracy, so the current research will focus on improving the performance of the algorithm as well. Also, the literature review proved that the gap of knowledge according to the zero-day domains problem. Where most of the work done so far do not solve this problem.

In the next chapter, the adaptive dynamic evolving spiking neural network based on the proposed initial weight of spike time, as one of the contributions of this study, will be introduced and explained in detail.

CHAPTER THREE

ADAPTIVE DYNAMIC EVOLVING SPIKING NEURAL NETWORK – ADESNN

Chapter Overview

This chapter presents the first contribution of the current work, which is the initial weight initialization based on the spike time of the incoming inputs. Then, the adaptive dynamic evolving spiking neural network algorithm and the original one are compared. Finally, the achieved results are discussed.

3.1 Introduction

This chapter presents the proposed adaptive DeSNN. DeSNN algorithm that is built based on the RO learning rules and the SDSP learning algorithm. According to previous work, the initial weight of the DeSNN is calculated based on the RO rules. As stated in (Kasabov et al., 2013), the output of the DeSNN algorithm consists of the initial and final weight matrices, as a new incoming input pattern is arrived an initial weight is computed as well as the final weight. At the recall mode the classification of the new arrival is going to be based on the Euclidean distance testing measure-. Experiments showed that the current initial weight based on the RO setting introduces a clear misclassification percentage of detecting of the incoming inputs, while the proposed approaches give a satisfactory accuracy percentage compared with the former one.

3.2 Adaptive Dynamic Evolving Spiking Neural Network

The expected approach is an online detection approach, and is dealing with real data so spiking neural network (SNN) is used. Systems based on SNN have already showed their ability to capture spatial and temporal data. Evolving Spiking Neural Network (eSNN), are based on a one-pass rank order (RO) learning rules and a scheme to evolve a new spiking neuron and connections, which lead to learn new patterns from arriving data. This chapter presents an

adaptation of the dynamic evolving spiking neural network (deSNN), that employ both RO learning and dynamic synapses to learn spatial and temporal data in a fast and on-line mode. By employing both RO learning and Spike Driven Synaptic Plasticity SDSP, deSNN -as depicted in Figure 3. 1 - could be used in unsupervised, supervised, or semi-supervised learning modes. The proposed approach is a hybrid learning approach, where the supervised learning phase works offline and the unsupervised learning phase works online to detect the zero-day domains. The SDSP learning is used to dynamically update the connection weights of the network that capture data clusters both through training and through recall.

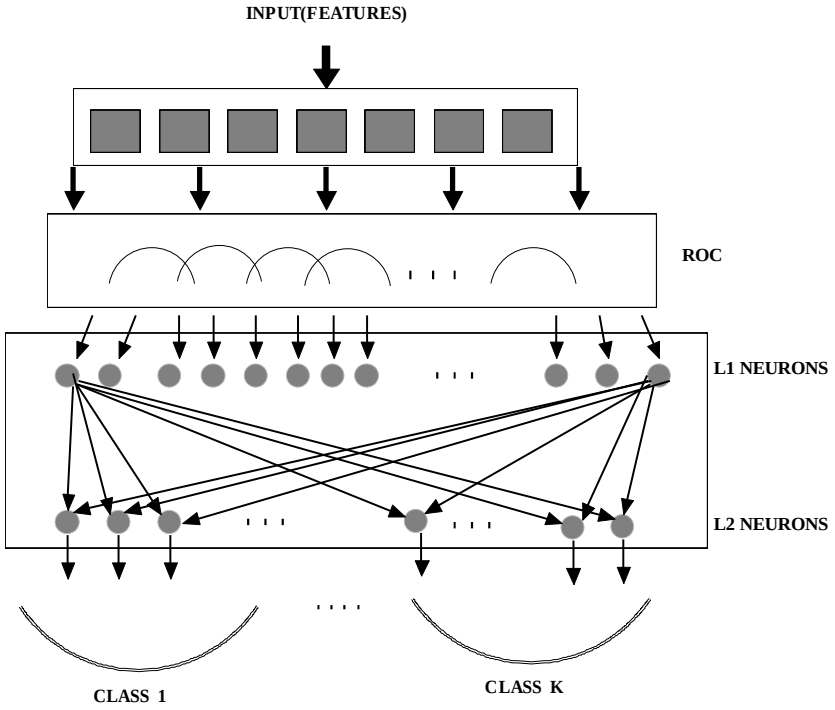


Figure 3. 1 An evolving spiking neural network (classification) (Kasabov et al., 2013)

At training phase for each training input pattern $\chi \iota$, a new output neuron $_{j}$ is created, also the connection weights w*ij* of the input neurons (feature) is initiated according to Matrix (1), the weights calculations are based on the spike times of each input:

Matrix $\chi 0$ contains the input records of all inputs.

$$\chi = \begin{Bmatrix} \chi 11 & \chi 11 & \dots & \chi 1m \\ \chi 21 & \chi 21 & \dots & \chi 2m \\ & & . & \\ & & . & \\ & & . & \\ \chi n1 & \chi n1 & \dots & \chi nm \end{Bmatrix} \qquad (\chi\ 0)$$

Here the first row of the matrix χ refers to the input record, where the first record consists of the feature set, and so on. "n" refers to the number of the inputs records, "m" is the number of features in each input record. Matrix (1) below contains the spike times of all input records after ROC encoding process exploiting Gaussian receptive field (Soltic & Kasabov, 2010). Here, the first contribution of the proposed approach is to use the spike time records as the initial weight instead of the initial weight created using RO learning rules as stated in (Kasabov et al., 2013). More about this contribution is discussed in details in this chapter.

$$\boldsymbol{ST} = \begin{Bmatrix} ST11 & ST11 & \dots & ST1m \\ ST21 & ST21 & \dots & ST2m \\ & & . & \\ & & . & \\ & & . & \\ STn1 & STn1 & \dots & STnm \end{Bmatrix} \qquad (1)$$

Where $STij$ refers to the spike time of the ιth input record. Each record comes as a result of the number of the Gaussian receptive fields multiplied by the number of the features of the original input record. Moreover, the proposed contribution is to replace the initial value that was set by the RO learning rules by the spike times of the input records after the ROC encoding. The reason behind this modification is to improve the detection accuracy. Figure 3.2 shows both the initial

weights initiation techniques for both the RO and the spiketime initial weights, it is clear that using the spike time instead of the rank of the spikes order is more related to the incoming data, and helps in classifying input records correctly as depicted in Figure 3.2:

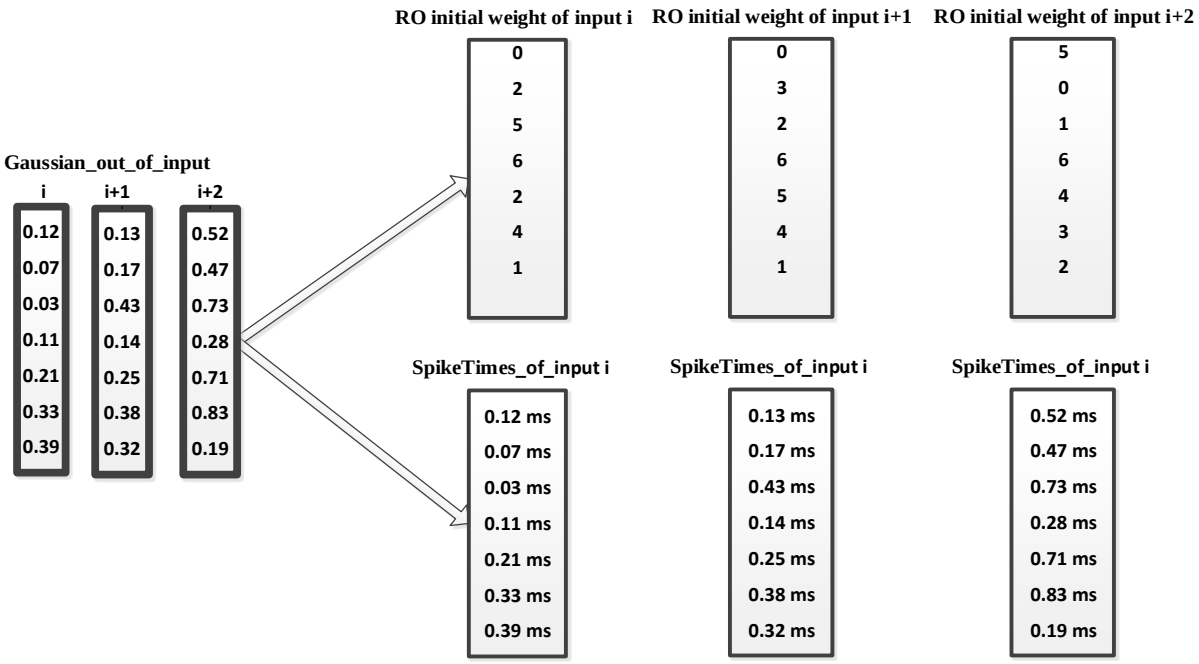


Figure 3. 2 An example of the RO and Spike Times initial values

According to the work done in (Kasabov et al., 2013), DeSNN algorithm is belt based on rank order learning rules to set the initial weight of the synapses among the input and the output neurons. Furthermore, SDSP adjusts the weight of the synapses based on the upcoming spikes, so at the end the DeSNN algorithm, the output is the weight matrices of the initial weight based on the RO rules, and the final weight is based on the SDSP learning rules.

One significant insight into the initial weight setting by the RO learning rules is that it has been noticed that all malicious and non-malicious input values are given the same initial values as depicted in Figure 3.2. This is because of the fixed mechanism of the RO initial values, the similarities between the outputs values of the final weight of the SDSP weight matrix of different inputs will be almost high, even with the updates on the synapses weights caused by the working of the evolving spiking neural network. These similarities are going to affect the classification process of both the malicious and non-malicious inputs, as the classification here

is based on the minimum Euclidian distances between the testing sample and the weight of the training samples at recall phase.

When the initial weight initialized based on the spike time matrix, so this helps in the classification process butter than assigned the same fixed weights mechanism for all different inputs. Figure 3.2 shows the idea more clearly. To show the effect of this change, this phenomenon will be discussed in the results and discussion sections of this chapter.

After the initiation of the weight on the synapses of jth neuron based on the spike time matrices of the incoming inputs, the dynamic synapses adjust their weights based on the SDSP algorithm according to equation (2). While the spike arrives at any time t, weight value increases, as there is no spikes arriving at this time, weight value decreases:

$$\Delta\omega_{j,i}(t) = ST_j(t).D \qquad (2)$$

Where $ST_j(t)$ equals to 1, if there is a sequenced spike at time t arrives at synapse j of arriving learning patterns at the output neuron j, and it equals to (-1) otherwise. D is the drift parameter, which can be changed for up or down drifts.

In parallel, all synapses change their values in every time unit t according to equation (3), while the input patterns Pi arrive at the output neuron i. Based on these values which may go up or down, the synapses of the neuron all together could capture nearly all relationships of spike timing through the learned pattern. Continuously, as the incoming training patterns arrive (input spikes on different synapses), they are encoded within the time window T. Then the threshold Th_i of the neuron is defined. Based on the value of this threshold the neuron i spikes or not. The threshold is defined in equation (4) as a fraction of the entire PSPi (PSPimax) collected through the appearance of the Entire input pattern

$$PSP_{imax} = \sum_{t=1,2,\ldots,T} \sum_{j=1,2,\ldots,M} f_j(t).\omega_{j,i}(t) \qquad (3)$$

$$Th_i = C.PSP_{imax} \qquad (4)$$

Where: T is the time window in which the input patterns arrived, M is the number of neuron I input synapses, $f_j(t)$ equals to 1 if the spike appears in the time window at the synapse j for this input pattern, if not it equals to 0. $\omega_{j,i}(t)$ is the efficacy of the dynamic synapse between the neurons j,i which is calculated in equation (2).

Figure 3.3 shows the architecture of the DeSNN algorithm, also positions the rank order encoding method based on multiple Gaussian receptive fields. In addition, the figure presents the SDSP learning rule which adjusts the synapses weights. These weights change up and down based on the drift parameter value which is discussed before.

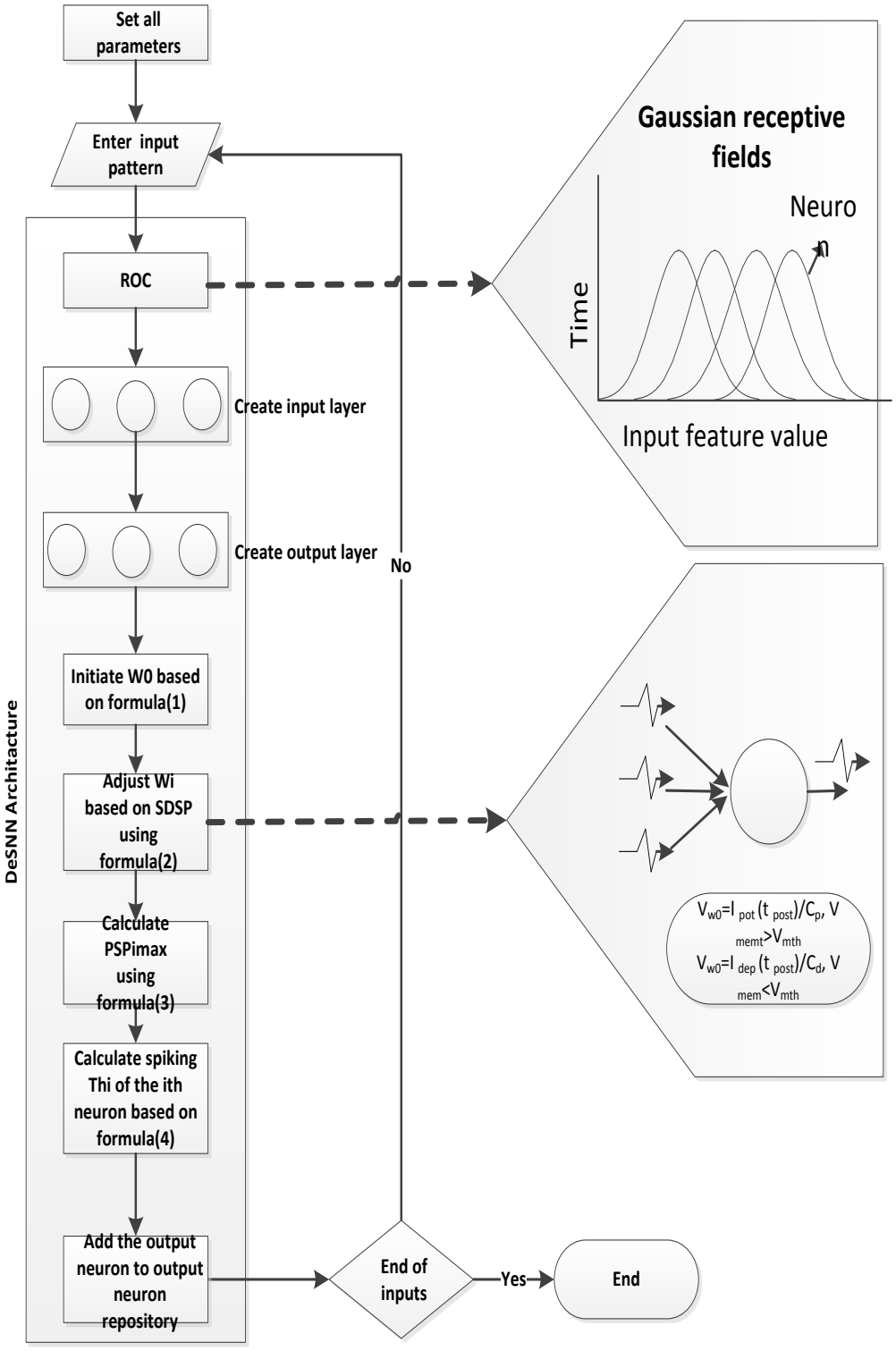


Figure 3. 3 The DeSNN algorithm architecture

The original DeSNN algorithm is mentioned in the algorithm 3.1 in this chapter.

Algorithm 3.1 The original deSNN algorithm(Kasabov et al., 2013):

- Setting deSNN parameters* (RO and the SDSP parameters too)
- For each input spatio-temporal spiking pattern Pi Do
    - Create a new output neuron i for this pattern and calculate the initial values of connection weights wi(0) using the RO learning formula (1).
    - Adjust the connection weights wi for consecutive spikes on the corresponding synapses using the SDSP learning rule formula (2).
    - Calculate PSPimax using formula (3).
    - Calculate the spiking threshold of the ith neuron using formula (4).
    - (Optional) If [The new neuron weight vector wi is to the weight vector of an already trained output neuron] then
        - merge the two neurons
    - Else
        - Add it to the output neurons repository.
    - End If
- End For

3.3 Dataset

Two public datasets were used in this chapter to evaluate the performance of both the DeSNN and the Adaptive DeSNN algorithms (Especially to evaluate the effect of using the new proposed initial weight). As the proposed adaptive algorithm classify fast flux and benign domains and the unsupervised phase of the proposed FFKA approach capture the new fast flux pattern. So, IRIS dataset (Benjamin & R.A., 2013) and Wisconsin Diagnostic Breast Cancer (WDBC) dataset (Dua, 2017) is introduced. The public IRIS dataset, consists of 3 classes each with 50 instances, this dataset is the best known dataset for pattern recognition (Barra, Casanova, Narducci, & Ricciardi, 2015; Z. Lin, Ma, Meng, & Chen, 2018). One class is linearly separable from the other two, but the latter two classes are not separable from each other. The IRIS dataset has 4 features which are (sepal length, sepal width, petal length, and petal width). WDBC public dataset is a well-known binary medical dataset and many of machine learning algorithms used it in their experiments for pattern recognition and classification purposes (Aličković & Subasi, 2017; Basu, Roy, & Savitha, 2018; Mandal, 2017; Zheng, Yoon, & Lam, 2014), the characteristics of both datasets are depicted in Table 3.1.

Table 3. 1 Characteristics of the WDBC dataset.

Dataset	Number of		
	Classes	Records	features
IRIS dataset	3	150	4
Wisconsin Diagnostic breast cancer (WDBC)	2	569	10

The ten real-valued features of WDBC are computed for each cell nucleus: (radius (mean of distances from center to points on the perimeter), texture (standard deviation of gray-scale values), perimeter, area, smoothness (local variation in radius lengths), compactness (perimeter^2 / area - 1.0), concavity (severity of concave portions of the contour), concave points (number of concave portions of the contour), symmetry, and fractal dimension).

3.4 Experiments, Result, and Comparison

This subsection shows the results of the experiments that were conducted on the DeSNN and ADeSNN algorithms. Then, both results were compared to each other to prove that the performance of the adaptations that have been introduced on DeSNN have significantly improved the results.

The idea behind modifying the DeSNN algorithm came from many of our experiments; it seemed that the result always gave the same range of certain results boundaries. While tracing the variables' values and each process outputs, it became clear that the problem was the setting of the initial weight that is the RO initial weight. The initial weight sets by the RO is based on giving the same values with different orders, and this order was changed based on the spike time of the incoming spike at the identified neuron, as depicted in Figure 3.2.

Next, two experiments were conducted to prove the efficiency of the proposed adaptive ADeSNN compared with the original DeSNN as mentioned in algorithm 3.1. Two public datasets were also used. These datasets have non-leaner separable attributes, and therefore the classification task was not straight forward.

The software and hardware used in these experiments were based on Linux mint operating system run with the following attributes, core i7 7500U CPU, 16GB RAM, the simulations of the compared method were conducted using MATLAB 8.5 and Python 2.7 environments.

Various detection accuracy methods were conducted to evaluate the proposed algorithm, and the results of these measures are presented in each experiment. Table 3.2 provides the description of the measures used in this experiment.

Table 3. 2 Accuracy measure used in all experiments

Measure	**Equation**	**Description**
True positive **TP**	$\sum_{1}^{i}(1)$, where i belongs to positive instances	Number of benign domains accurately identified as benign domains
True negative **TN**	$\sum_{1}^{i}(1)$, where i belongs to negative instances	Number of fast flux domains accurately identified as fast flux domains
False positive **FP**	$\sum_{1}^{i}(1)$, where i does not belong to positive instances	Number of fast flux domains identified as benign domains.
False negative **FN**	$\sum_{1}^{i}(1)$, where i does not belong to negative instances	Number of benign domains identified as fast flux domains.
False positive rate $\boldsymbol{FPR}$	$\frac{\boldsymbol{FP}}{(\boldsymbol{TN}+\boldsymbol{FP})}$	The percentage of positive cases misclassified as negative cases
Recall or True positive rate $\boldsymbol{TPR}$	$\frac{\boldsymbol{TP}}{(\boldsymbol{TP}+\boldsymbol{FN})}$	The percentage of positive cases that classified as positive cases.

Accuracy $\boldsymbol{ACC}$	$\frac{(\boldsymbol{TP}+\boldsymbol{TN})}{(\boldsymbol{TP}+\boldsymbol{TN}+\boldsymbol{FP}+\boldsymbol{FN})}$	The percentage of correct predictions of all instances.
Precision	$\frac{\boldsymbol{TP}}{(\boldsymbol{TP}+\boldsymbol{FP})}$	The percentage of cases correctly classified as positive cases.
Fmeasure	$\frac{(\boldsymbol{2}\times\ \boldsymbol{Precision}\times\ \boldsymbol{Recall})}{(\boldsymbol{Precision}+\boldsymbol{Recall})}$	Accuracy test measure using both the precision and the recall.
Root mean square error **RMSE**	$\sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\frac{(\boldsymbol{yi}-\boldsymbol{ti})^{\boldsymbol{2}}}{\boldsymbol{N}}}$	The differences between the target and the actual expected value
	Where: N : Number of input, yi: Actual value, ti: Target value. $RMSE$ is an vital measure of differences between the values expected from a model and the values actually detected.	
Non-Dimensional Error Index	$\boldsymbol{NDEI}=\frac{\boldsymbol{RMSE}}{\boldsymbol{std}(\boldsymbol{ti})}$	NDEI is used to estimate the prediction quality (Espinosa & Vandewalle, 2000).
The Matthews Correlation Coefficient	$\frac{(\boldsymbol{TP}\times\boldsymbol{TN})-(\boldsymbol{FP}\times\boldsymbol{FN})}{\sqrt{((\boldsymbol{TP}+\boldsymbol{FP})(\boldsymbol{TP}+\boldsymbol{FN})(\boldsymbol{TN}+\boldsymbol{FP})(\boldsymbol{TN}}}$	MCC is used to evaluate the efficiency of the classifier in imbalanced classes (Matthews, 1975).
The receiver operating characteristic ROC	Graphical plot that depicts a binary classifier's performance	ROC arcs plot the true positive rate on the vertical axis and the false positive rate on the horizontal axis

The AUC represents the classifier's performance (Swets, 2014). Likewise, the AUC is well-known to be a more robust estimator of classifier performance (Fawcett, 2006).

A question may raise why the two datasets were used in this research. This was done to prove that the initial weight initialization based on the spike times would give better results in the classification process than the old method. To the best of our knowledge, no one has pointed to this problem before. To ensure the quality of the proposed adaptation of the AdeSNN algorithm, a cross-validation method is used to estimate the error rate.

The first experiment was conducted to show the performance of the two algorithms, the proposed adaptive ADeSNN and the original DeSNN as mentioned in algorithm 3.1. Therefore, the IRIS public dataset was exploited. It consists of 3 classes each with 50 instances; this dataset is the best-known dataset for pattern recognition as this study is dealing with looking to detect new unknown patterns (zero-day domains). One class is linearly separable from the other two, but the latter two classes are not separable from each other. The dataset was randomly initiated into three groups, then the experiments of 3-fold cross-validation datasets were selected. At the end of these three experiments the average was taken to present the results shown in Table 3.3 and Figure 3.4.

Table 3. 3 The 3-fold cross-validation result of both the original DeSNN and the proposed ADeSNN of the first experiment

Evaluation measures	**DeSNN**	**ADeSNN**
FNR	0.4400	0.0625
TPR	0.5600	0.9375
ACC	0.5600	0.9167
Precision	0.5600	0.9091
Recall	0.5600	0.9375
F1-Measure	0.5600	0.9231
MCC	0.1200	0.8327
AUC	0.5600	0.9152
RMSE	0.6633	0.2887
NDEI	1.3200	0.5749
MSE	0.4400	0.0833

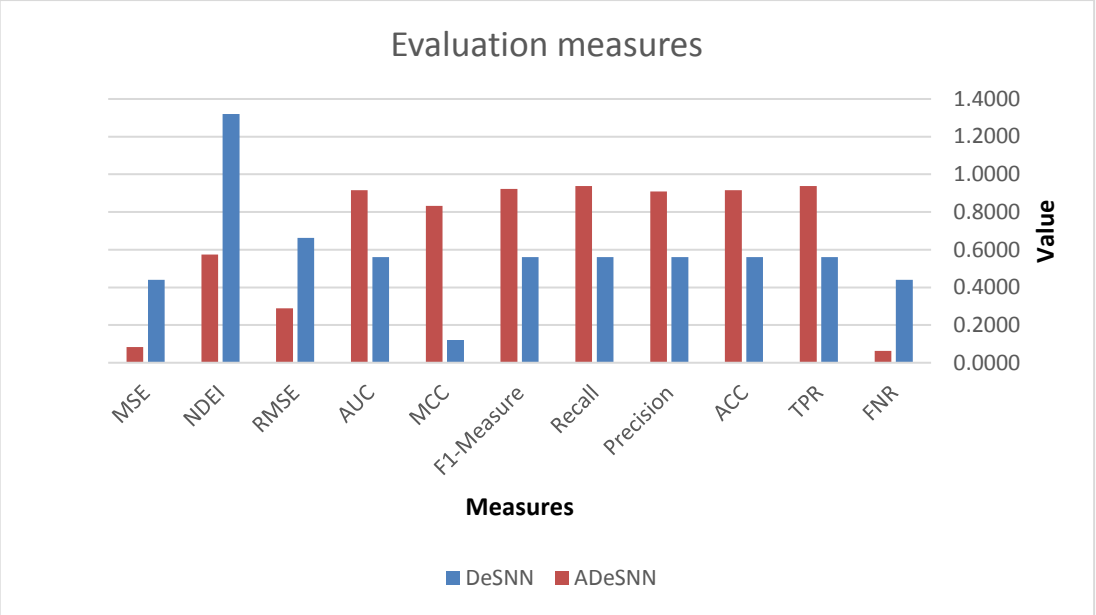


Figure 3. 4 The accuracy measures of both DeSNN and ADeSNN

Table 3.3 shows the results of the average of the cross-validations of both algorithms. The error measures (RSME, NDEI, and MSE-ERROR) of the proposed adaptive ADeSNN were less than those for the original DeSNN as mentioned in algorithm 3.1, which means that the adaptation of the DeSNN will minimize the misclassification of the input instances. For example, the root mean square error of the original algorithm was 66% while in the adaptive algorithm became 28%, also the non-dimensional error index value for the original algorithm according to this experiments was 1.32 compared with the adaptive algorithm which was 0.57. This also helps to maximize the accuracy of the detection and classification. In addition, the MCC is a performance metric which is widely used in bioinformatics. The two algorithms used this metric because it best deals with the imbalanced data, and this leads us to conclude that the adaptive algorithm has a high degree of correctness, surpassing the original one even in cases of imbalanced data. By comparing the following accuracy measures, all of F-measure, Recall, and ACC they revealed that the proposed adaptation produced more accurate results than the original DeSNN. The precision measure value enhanced to reach 90% while it was 56% in the original algorithm, as well as the recall measure reached the 93%, and the f-measure became 92% .The overall accuracy of DeSNN was (56%) while it was (91.67%) for ADeSNN. According to the IRIS dataset two classes were non-linearly separable which cause to show

almost 91% accuracy while it was tested on the first linearly separable classes and give 100% accuracy. This leads to the ability of the adaptive DeSNN to classify classes even when inputs are mutually mixed. Finally, ADeSNN exhibited higher true positive rate and less false negative rate than DeSNN as depicted in Figure 3.4.

The parameters of the ADeSNN algorithm set in the experiment are shown in Table 3.4.

Table 3. 4 The parameter values used in the 3-cross-validations of the first experiment

Neurons and synapses equations parameters	Value	Unit
Excitatory synapse time constant (tau_exc)	2	Ms
Inhibitory synapse time constant (tau_inh)	5	Ms
Neuron time constant (tau_mem)	20	Ms
Membrane leak (El)	20	mV
Spike threshold (Vthr)	800	mV
Reset value (Vrst)	0	mV
Fixed inhibitory weight (winh)	0.20	V
Fixed excitatory weight (wexc)	0.40	V
Thermal voltage (UT)	25	mV
Refractory period (refr)	4	Ms
SDSP parameters		
Up/Down weight jumps (Vthm)	0.75*Vthr	mV
Calcium variable time constant (tau_ca)	5 *tau mem	Ms
Steady-state asymptote for Calcium variable (wca)	50	mV
Stop-learning threshold 1 (stop if *V*ca < thk1)	1.7 × wca	mV
Stop-learning threshold 2 (stop LTD if *V*ca > thk2)	2.2 × wca	mV
Stop-learning threshold 2 (stop LTP if *V*ca > thk3)	8 × (wca–wca)	mV
Plastic synapse (NMDA) time constant	9	Ms
Plastic synapse high value (wp hi)	6	mV
Plastic synapse low value (wp lo)	0	mV
Bistability drift	0.25	
Delta weight	0.12 × wp_hi	mV
Input size	150 spike train	
Simulation time	40	Ms
Default clock unit	0.2	Ms

The second experiment was conducted and exploited the public WDBC dataset in order to prove the effectiveness of the proposed adaptation on the original DeSNN algorithm. WDBC public dataset is a well-known binary medical dataset and many machine learning algorithms used it in their experiments for pattern recognition and classification purposes. The WDBC dataset has ten real-valued features for each cell nucleus, where they computed from a digitized image of a fine needle aspirate of a breast mass, which are radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. Furthermore, the dataset contains 569 instances presenting two classes the (diagnosis: B = benign, M = malignant), the two classes distribution were 357 benign, 212 malignant. Finally, this dataset has no missing attribute values.

Current experiment distributed the dataset into 5-fold cross-validations groups. So, five separated experiments were done, where the instances randomly distributed on the five groups, then the results were computed and the average was taken, as depicted in Table 3.5 and Figure 3.5, the result of several measures to compare both of the original and the proposed adaptive algorithms were presented.

Table 3. 5 The 5-fold cross-validation result of both the original DeSNN and the proposed ADeSNN of the second experiment

Evaluation measures	**DeSNN**	**ADeSNN**
FNR	0.402439	0.04898
TPR	0.597561	0.95102
ACC	0.765957	0.971631
Recall	0.597561	0.95102
F_measure	0.748092	0.958824
MCC	0.619046	0.937925
RMSE	0.483779	0.172406
NDEI	0.965842	0.344209
mse_error	0.234043	0.029787

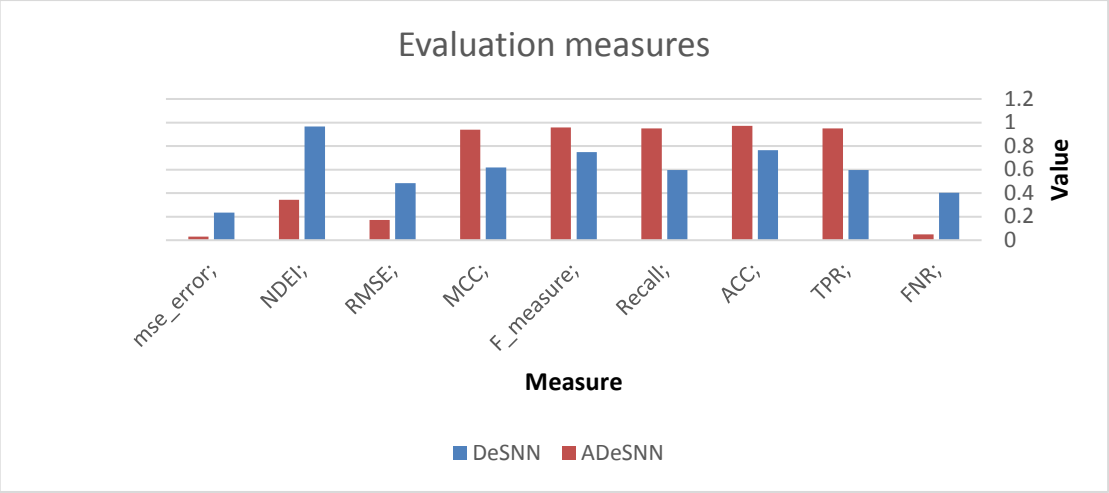


Figure 3. 5 The accuracy measures of both DeSNN and ADeSNN

Table 3.5 showed the result of average of the cross-validations of both algorithms. The error measure (RSME, NDEI, and MSE-ERROR) values of the proposed adaptive ADeSNN were less than those for the original DeSNN, which means the adaptation on the DeSNN will minimize the misclassification of the input instances, and maximize the accuracy of the detection and classification. In addition, the MCC is a performance metric which is widely used in bioinformatics, the two algorithms used this metric because they best deal with the imbalanced data, and this leads to conclude that the adaptive algorithm has higher accuracy than the original one. Coming to compare the accuracy all of F-measure, Recall, and ACC revealed that the proposed adapted algorithm produced more accurate results (97.16%) than the original DeSNN (76.59%). Finally, ADeSNN exhibited higher true positive rate and less false negative rate than DeSNN as depicted in Figure 3.5.

The parameters of the ADeSNN algorithm set in the experiment are shown in Table 3.6.

Table 3. 6 The parameter values used in the second experiment

Neurons and synapses equations parameters	Value	Unit
Excitatory synapse time constant (tau_exc)	2	Ms
Inhibitory synapse time constant (tau_inh)	5	Ms
Neuron time constant (tau_mem)	20	Ms
Membrane leak (El)	20	mV
Spike threshold (Vthr)	800	mV
Reset value (Vrst)	0	mV
Fixed inhibitory weight (winh)	0.20	V
Fixed excitatory weight (wexc)	0.40	V
Thermal voltage (UT)	25	mV
Refractory period (refr)	4	Ms
SDSP parameters		
Up/Down weight jumps (Vthm)	0.75*Vthr	mV
Calcium variable time constant (tau_ca)	5 *tau mem	Ms
Steady-state asymptote for Calcium variable (wca)	50	mV
Stop-learning threshold 1 (stop if *V*ca < thk1)	1.7 × wca	mV
Stop-learning threshold 2 (stop LTD if *V*ca > thk2)	2.2 × wca	mV
Stop-learning threshold 2 (stop LTP if *V*ca > thk3)	8 × (wca–wca)	mV
Plastic synapse (NMDA) time constant	9	Ms
Plastic synapse high value (wp hi)	6	mV
Plastic synapse low value (wp lo)	0	mV
Bistability drift	0.25	
Delta weight	0.12 × wp_hi	mV
Input size	569 spike train	
Simulation time	40	ms
Default clock unit	0.2	Ms

Based on the above two experiments, the adaptation on the algorithm gave excellent results compared to the original one. The next chapter will build on this adaptation to detect the zero-day fast flux domains.

3.5 Chapter Summary:

This chapter presented our first contribution in adapting the ADeSNN algorithm, and the comparison experiments have been conducted on both supervised FFKA algorithm ADeSNN and the original DeSNN. The proposed adaptation can be summarized as producing the initial

weight of the RO based on the spike time itself rather than the RO initial weight based on the ranks of the order of the incoming spikes. The current version has been modified based on the changing of the initial weight of the RO, where the initial weight is based on the spike time of the incoming spikes, as a results the (mod) variable became useless, which leads that the current contribution improved the parameters customization problem.

Furthermore, two public datasets were used to evaluate the two DeSNN and the adaptive ADeSNN algorithms. The first experiment used the IRIS dataset and produced 3-fold cross-validations, then the average of their result was computed. The results of the experiment showed that the adaptive algorithm has enhanced the performance of the DeSNN, where the overall accuracy was (91.67%) of the ADeSNN over DeSNN (56%).

The second experiment exploited the public WDBC dataset and this dataset was randomly distributed over 5-fold cross-validations. The average of the results was taken, and several accuracy measures were tested, all the discussed results above proved that the performance of the adaptive ADeSNN algorithm is better than DeSNN, the overall accuracy of the ADeSNN was (97.16%) while the original DeSNN was (76.59%).

As a result, the contribution presented in the adaptation on the DeSNN algorithm by modifying the initial weight mechanism, improves the performance of the algorithm, and achieves better results in several ways. The adaptation also minimizes the error rates of the classification process, as show in Tables 3.3, respectively. Finally, the exploitation of two other public datasets gave comparable results. In chapter four, the new adaptive supervised FFKA frame work will be introduced to detect the fast flux domains.

CHAPTER FOUR

SUPERVISED FAST FLUX KILLER APPROACH FFKA

Chapter Overview

This chapter presents the fast flux killer approach, then introduces the first phase, which is the supervised one. FFKA in this phase trains the adaptive dynamic evolving spiking neural network and sets the classification threshold. The evaluation of the proposed method is then compared with two fast flux detection approaches in the field. Also, this chapter presents the new feature set that helps the proposed approach to accurately classify the fast flux domains.

4.1 Introduction

According to the achieved enhancement regarding the initial weight described in the chapter three, that the proposed adaptation of the DeSNN algorithm improved its performance, this chapter introduces the FFKA which has to be exploited to detect the fast flux domains. FFKA works online and offline. In the online mode, FFKA deals with unknown new domains which the approach was not trained on in order to detect the zero-day fast flux domains. On the other hand, the offline mode deals with labelled data where the supervised learning phase of the FFKA will be trained to produce an enhancement in the classification process. Section 4.2 will present the FFKA in both phases: the supervised and the unsupervised. Finally, the rest of this chapter will discuss the supervised mode, and the unsupervised mode will be discussed in chapter five as a part of the FFKA Hybrid approach.

4.2 Fast Flux Killer Approach

ADeSNN was discussed in chapter three in details, this algorithm was designed to work online, that means it adapts its' structure and functionality based on the incoming data. FFKA as depicted in Figure 4.1, is a Hybrid learning approach that employed two parallel ADeSNN algorithms, the former works as supervised in an offline mode and the later works as unsupervised in an online mode.

To evaluate the proposed supervised FFKA, a public fast flux dataset was used in order to test the ability of the ADeSNN algorithm to detect fast flux domains with labeled dataset at the supervised phase in an off-line mode to adjust the threshold value of the classification process.

The supervised mode in FFKA is about training the ADeSNN algorithm on both fast flux domains and benign domains. Besides, a threshold of the classification process will be trained along the training process. The outputs of the supervised training mode are the final weight and the classification threshold, where the weights are stored in the weights repository. Furthermore, the threshold stored to be accessed by both the supervised and unsupervised modes later.

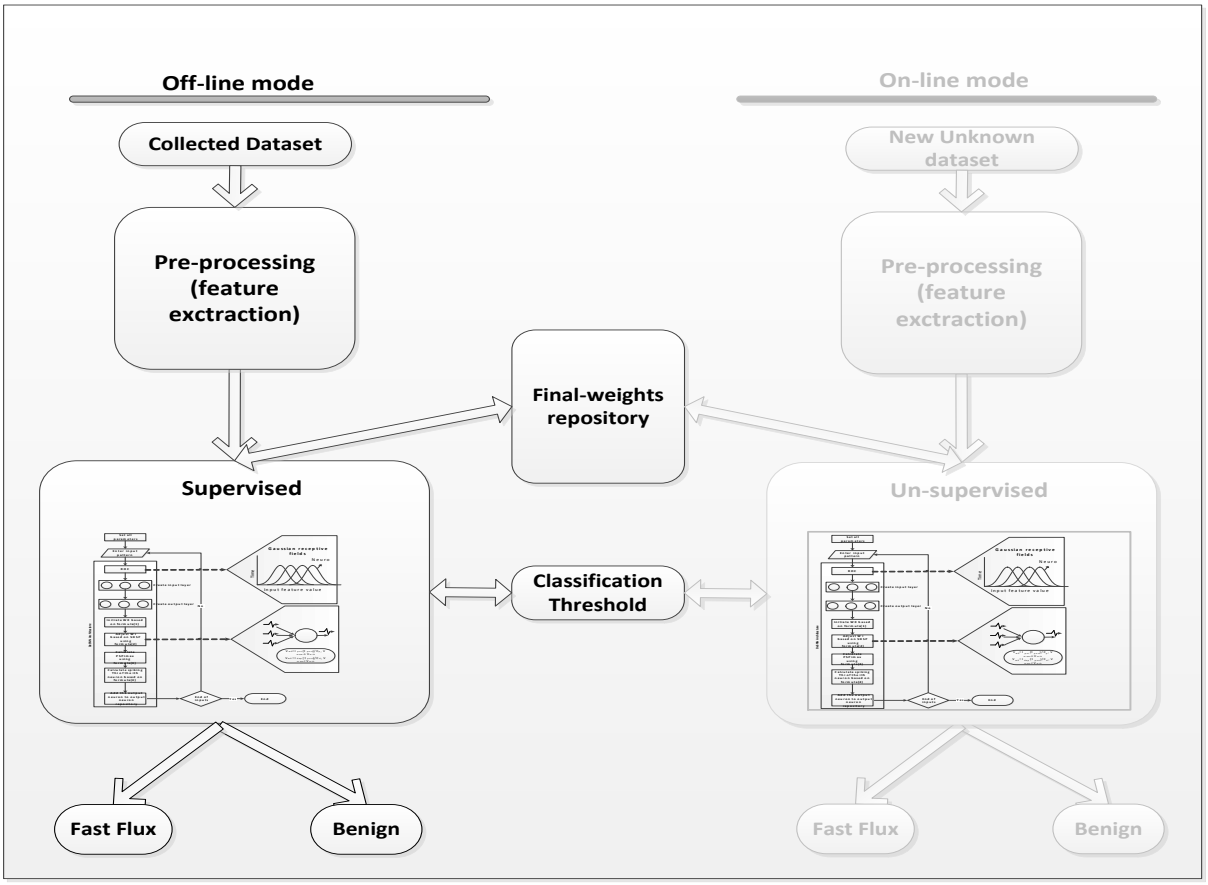


Figure 4. 1 The architecture of the FFKA

Figure 4.1 shows the two supervised and unsupervised phases of the FFKA Hybrid approach. The first step starts with letting the approach learn how to classify the benign and fast flux domains based on labelled data, then stores the output of the ADeSNN algorithm in the weights

repository and stores the calculated classification threshold as well. The second unsupervised phase that will be discussed in chapter five continues to deal with new unknown unlabeled data. The decision here is taken based on the output of phase one, specifically the classification threshold.

4.3 Supervised Learning Phase

The supervised learning phase as mentioned in section 3.2 trains the classifier used (ADeSNN) on labelled data. It also produces the classification ability based on the features of each class. Based on this, the discussion here is about the stage of preparing the dataset, the process of the feature extraction, and the learning process based on the FFKA supervised phase.

4.3.1 The Preprocessing Stage

The fast flux public dataset found as stacks of DNS responses, a script of python was written to extract information needed to build the feature set. Some feature needs to contact the ASN to get extra information about the IP addresses, and to speed up the process of building the feature an ASN repository that is located in the local dive of the approach as depicted in Figure 4.2.

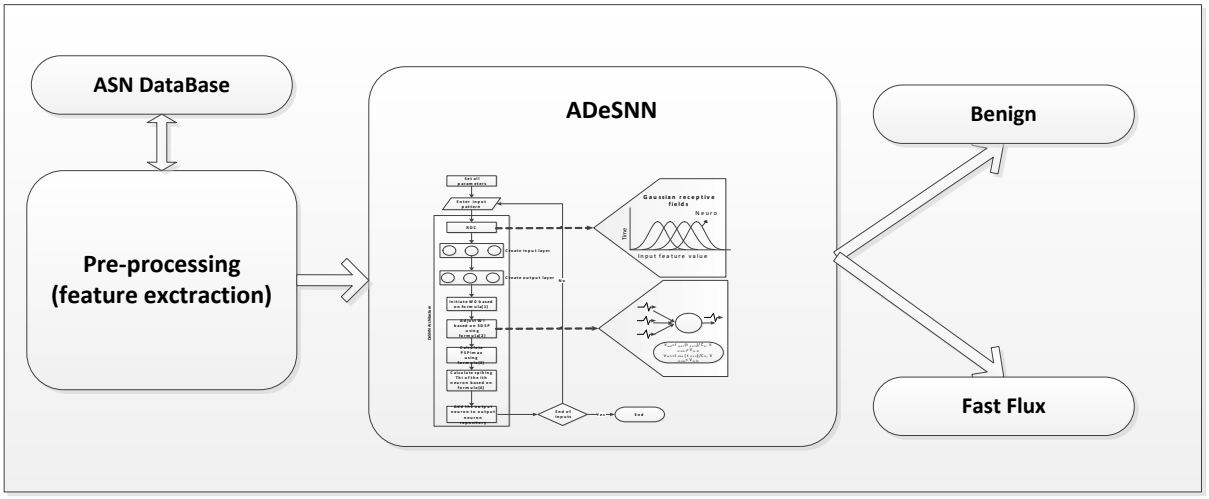


Figure 4. 2 The pre-processing phase

4.3.2 Feature Extraction

This stage is about how to build and calculate some of the features that need calculations. Building some features requires extra information from the ASN repository so one of the updated and freely downloadable ones was saved into the local drive to speed up the process of detection. For example, the number of the ASN for the answer section of the DNS response as well as the number of ASN of the additional section in the same DNS response message needs extra information from the ASN repository, in this case from the local drive. Another example, the similarity feature need to access the same repository to calculate the similarity between the autonomous system numbers of the user and the autonomous system numbers of the returned IPs. Finally some other features could be taken straight away from the DNS response message.

4.3.3 Adaptive dynamic evolving spiking neural network

The trained ADeSNN starts to classify the new upcoming inputs to the proper class benign/fast flux, one by one, so for each input the algorithm creates one output neuron, and so on. Moreover, the algorithm continues learning from the incoming inputs incrementally. The ADeSNN algorithm was discussed in details in section 3.2.

The criteria of testing and classification of the DeSNN and ADeSNN algorithms was to perform the Euclidean distance between the weight of the new input record and the weights of the trained inputs. As shown in chapter 3, the results were good so far, but in case of changing this criteria to the proposed similarity measure, it is expected that this would give better results than before. The similarity criteria is based on the calculation of the similarity between the new input weight and the already trained inputs weights according to formula 1 in chapter 4, which is one another contributions of this work. Two variables are significantly similar if the similarity between them is closest to 1, and they are not similar to each other if their similarity is closest to 0, which means that the bigger value refers to highest similarity between the two variables. As there are a classification process to classify more than one class, a testing threshold is needed to be set to each class, with suitable threshold value that separates all classes. Moreover, the threshold takes its value while the training phase is in progress, as each input is classified for one class so the threshold of this class is calculated based on the values of the inputs of the specific class.

4.4 Dataset

The proposed adaptation on ADeSNN method has to be evaluated and tested in case of detecting fast flux domains, so a public dataset was used in this study (Alvi et al., 2017),which is also used in (Huang et al., 2010).The majority of the learning machine approaches such as, the real-time, active, passive approaches used also the same sources (Castelluccia et al., 2009; C.-M. Chen et al., 2013; Holz et al., 2008; Martinez-Bea et al., 2013; Qassrawi & Zhang, 2012; Sheng et al., 2010; B. Yu et al., 2014). This dataset consists of DNS responses that were labeled domain names as benign and fast flux. The benign domains are selected from the top trusted websites like Alexa (Alexa), top blogs as Blogs On Top "BOT". While the fast flux domains are collected from the famous fast flux blacklisted websites such as ATLAS ("ATLAS URL: https://www.arbornetworks.com/atlas-portal,"), DNSBL ("DNSBL URL: https://www.zerobounce.net/,"), and FluXOR (information security expert's detection systems). Each record contains the selected feature set that helps to identify each class. Fast flux dataset contains (1710) instances, while the benign dataset contains (3420) instances.

4.5 Feature Selection

Building detection systems and other classification systems needs to identify the feature or attribute set that best describes the problem and the needed solution. Also, these features should help in minimizing the irrelevance and redundancy problems, as well as minimizing the false correlation and classification of the detection and classification systems. Based on that, feature selection methods needs to get released of the irrelevancy and redundancy of the feature set without affecting or decreasing the performance (Balepin, Maltsev, Rowe, & Levitt, 2003; Giacinto, Roli, & Didaci, 2003; Lee & Stolfo, 2000).

The embedded model, as one of the main methods that deals with feature selection, joins the training phase for particular method E.g., decision tree algorithms, such as C4.5. The algorithm selects the greatest feature which is the best for classification. At that time they divided sub-space based on the carefully chosen feature. The algorithm repeats this process until a certain threshold is reached (Boutemedjet, Bouguila, & Ziou, 2009; Jeong, Kang, Jeong, & Kong, 2012).

4.6 Feature Set

The first stage of the proposed solution is the feature extraction. A well-built fast flux botnet detection method should distinguish between a legitimate and malicious network. On the other side, a well-built Fast Flux Network (FFN) seems like a benign CDN, due to returning a DNS records that belong to the same geographic areas. This leads to a detection systems that depends on IP address features to misclassify those types of FFN domains as benign CDNs. In addition, the FFN developers are trying to change the characteristics of the fast flux Network to evade detection, even if this modification affects the performance of the FFN. Therefore, a new detection approach should rely on features belonging to the FFN itself, as these features are not prone to change quickly.

Based on the current fast flux dataset, some of the features used in the proposed solution are used before in related works. Moreover, new features are suggested to improve and enhance the performance of the classifying process of the fast flux and benign domains. Table 4.1 shows the selected features set.

Table 4. 1 The proposed feature set

Feature	**Description**	**New feature**
IPans	Number of IP addresses in the answer section	**Not**
NSadd	Number of IP addresses in the additional section	**Not**
NASN_ans	Number of ASN for the IP addresses of the answer section	**Not**
NASN_add	Number of ASN for the IP addresses of the additional section	**Not**
AVGSIM	The average of similarity of the ASN (among the answer section and the ASN of the victim himself)	**New**
Qtime	Time of the query	**New**
Msgs	Message size	**New**

Table 4.1 shows the definition of all the feature set. The first two feature are straight forward and obtained from the response directly, NASN_ans and NASN_add need to get extra information from the local ASN repository as presented in Figure 4.2. AVGSIM is a new feature that computes the average of the similarity between the ASN number of the requested IP address and the other ASN number of the returned IP addresses of the DNS response. In other words,

AVGSIM refers to the average of the similarity between the autonomous system number of the user's IP and autonomous system numbers of the proxy bots (compromised computers) returned in the answer section of the DNS response, and is computed according to equation (1) (Alkhazaleh, Salleh, & Hassan, 2011):

$$M_i(y(e) - x(e)) = 1 - \frac{\sum_{j=1}^{n} yij(e) - xij(e)}{\sum_{j=1}^{n} yij(e) - xij(e)} \quad (1)$$

Where ė is the input vector. It could say that μ(ė) and δ(ė) are significantly similar if M(μ(ė),δ(ė)) ≥ ½, which means that the bigger value refers to high similarity between the two variables.

The majority of the chosen features were found in the literature work, but the last three features are new. In addition, the other two new features are the time of the query and the DNS packet size. All these features together showed that their ability to distinguish between the fast flux network (served by domains) and the legitimate domains. In the discussion section, these new features will be tested to show their effect on the process of classification. Finally, all the feature set was put into a feature selection evaluation to show their effectiveness rank on detection process.

4.7 Experiments and Discussion

This section discusses the experiments made to prove the effects of the proposed similarity measure to be used as a classification criteria in order to detect the fast flux domains in supervised offline mode. Furthermore, it discusses the feature set proposed from the point of view of the influence they affect the process of detection fast flux domains.

4.7.1 Introduction

The first part of the FFKA is the supervised learning offline mode. Here one more contribution is going to be added to the adaptation proposed on the DeSNN algorithm. In this chapter, the FFKA approach based on the ADeSNN algorithm detects the fast flux domains in an offline mode by using the fast flux public data set. In addition, the testing criteria in this stage is based on the proposed similarity measure in formula 1 in chapter 4. The results of the proposed

supervised phase of the FFKA will be evaluated based on a comparison with two of the fast flux detection approaches in the same field.

The feature set has been tested different times via several experiments, especially the new proposed features. The results was promising for some feature, at the same time one of the feature showed negative effect on the classification process. Thus, our recommendation is to exclude this feature from the list in future work.

4.7.2 Supervised Fast Flux Killer Approach Experiment

When the feature set became ready, the experiment can start. The supervised learning takes the inputs records and feeds them to the ADeSNN algorithm one by one. As the new input record has entered a new output neuron is created, at the end of the supervised phase the output neuron weight will have the captured pattern of the input records stored as a weighted matrix. Furthermore, these output weights will be saved at the weight repository. In addition, the threshold value was learning while the supervised phase was in progress, and the final value will be stored to be in the next phase of the unsupervised learning phase in order to help in the process of classification. Of course, this current supervised phase will classify the input records at the end in one of the two classes, the benign domain or fast flux domain.

The current supervised phase of the approach continuously run in an offline mode, where its output will be used in the unsupervised learning phase in an online mode. Periodically, the supervised phase re-executed once every 1000 (in the current experiment) new incoming domains at the unsupervised online mode. Where the learning this time will be based on the new data from the stored inputs from the unsupervised phase, then the threshold tuned to be best related to new nature of the new data and helps in the classification process.

The following discussion is part of the research validation and evaluation processes, which presented the comparison between the proposed FFKA and two other approaches from the related works (Celik & Oktug, 2013; Lin et al., 2013). A public dataset has been used to test the chosen classifiers, then compare their performance among the related previous works done in same field. The experiments were conducted using the mentioned fast flux public dataset, to test the ability of the proposed approach to solve fast flux problem based on the proposed

similarity measure in classification process as well as the proposed fast flux feature set. The performance and results of the experiments were promising and indicated an increase in the detection accuracy of fast flux domains.

Three different simulations were implemented on MATLAB and Python platforms. The hardware and software used for this experiment are the same as used in the subsection 3.4, two of them were selected based on two related previous researches, which is the linear decision function in (H.-T. Lin et al., 2013) and the C4.5 in (Celik & Oktug, 2013) algorithm. The third was the proposed adaptive ADeSNN based on the new adaptation of the similarity measure and the proposed feature set.

To ensure the quality of the supervised learnning phase of the ADeSNN, a 3-folded cross-validation method is used to estimate the error rate of the proposed classifier and the other two methods as well. Based on this, the average was taken of the linear decision, C4.5, and the FFKA classifiers. All the results of the three experiments are presented in Table 4.2.

Table 4. 2 The accuracy measures of the detection algorithms

Evaluation measures	**C4.5**	**Linear**	**Supervised FFKA**
FNR	0.06987	0.03930	0.00000
FPR	0.05333	0.05333	0.02410
TPR	0.93013	0.96070	1.00000
TNR	0.94667	0.94667	0.97590
ACC	0.93833	0.95374	0.98765
Precision	0.94667	0.94828	0.97531
Recall	0.93013	0.96070	1.00000
F-measure	0.93833	0.95445	0.98750
MCC	0.87680	0.90755	0.97561
AUC	0.9383988	0.9536827	0.9879518
RMSE	0.24834	0.21507	0.11111
NDEI	0.49641	0.42990	0.22188

According to Lin et al. (2013) a genetic approach was proposed as a real-time detection solution of the fast flux domains problem. This method suggested two detection features to classify the benign and the flux domains. Firstly, entropy of the domain name (E-DPN) of the preceding

node of the flux node (flux-agent), by using the trace route of all the returned IPs from the DNS response. Of course, if the E-DPN is high then, most probably, the domain that is classified as benign is otherwise classified as fluxed. Secondly, the Standard Deviation of Round Trip Time (SD-RTT) between the user and all the return IPs of the flux-agents, so assumed that the scatter flux-agent is going to produce high value of the SD-RTT. This spatial feature takes the number of different ASNs and number of IPs return in single DNS response in their calculations. However, these two detection features were evaded by the botmaster, as botmaster is controlling the returned list of IPS that the user receives. The returned list could have IPs in the same ASN or adjacent to the user ASN, so the above measures can inaccurately be classified as the benign and flux domains. On the other hand, botmaster may return a list containing just a single IP address, which leads to ineffective detection of the domains(Hsu et al., 2014; Otgonbold, 2014). Although genetic algorithms provide good accuracy as stated in their paper, but in case the botmaster decides to return the list of IP addresses in the same AS so the genetic algorithm results based on the countermeasures will be affected (Hsu et al., 2014). According to our implementations the overall accuracy of the linear classifier is 95.37 % as shown in Figure 4.3. Similarly, the linear decision function used as a classifier needs to estimate the categorizer of the linear function, so if the estimation is good then the linear function works properly, otherwise the error will be high in the classification process (Chahal & Khurana, 2016). So, the need for a classifier that detects the zero-day domains is still unsatisfied.

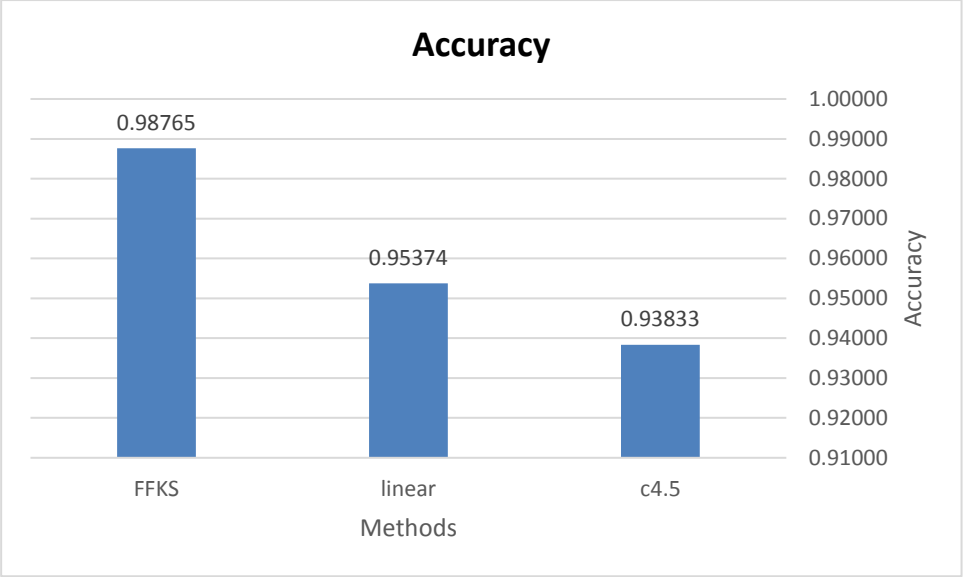


Figure 4. 3 The overall detection accuracy

On the other hand, the second compared algorithm was the C4.5 as presented in (Celik & Oktug, 2013). A number of feature sets were examined to detect fast flux network, such feature sets consist of timing based, spatial based, network based, domain based, and DNS answer based feature sets. As mentioned in the literature review, the data set was small even though the accuracy of the experiment was high; also when all features are involved in the experiment the prediction results become insensitive to two features (timing and domain based feature sets) (Otgonbold, 2014), which is the most related features to the domain resolution process.

Besides, as C4.5 algorithm is considered as a supervised learning algorithm, it could not be used to discover the unknown attacks, especially the zero-day fast flux domains, while the current Hybrid FFKA could efficiently detect this kind of domains.

Moreover, according to our implementation of the C4.5, the accuracy was not high as stated in their paper. According to the current experiment it was 93.38%. when this result and the previous linear results were compared to the current proposed FFKA, obviously the proposed approach overcome the two methods even in this part of the FFKA supervised phase with a total detection accuracy of 98.76%. Figures 4.4 and 4.5 showed other accuracy measures which are the ROC curve and F-measure, respectively.

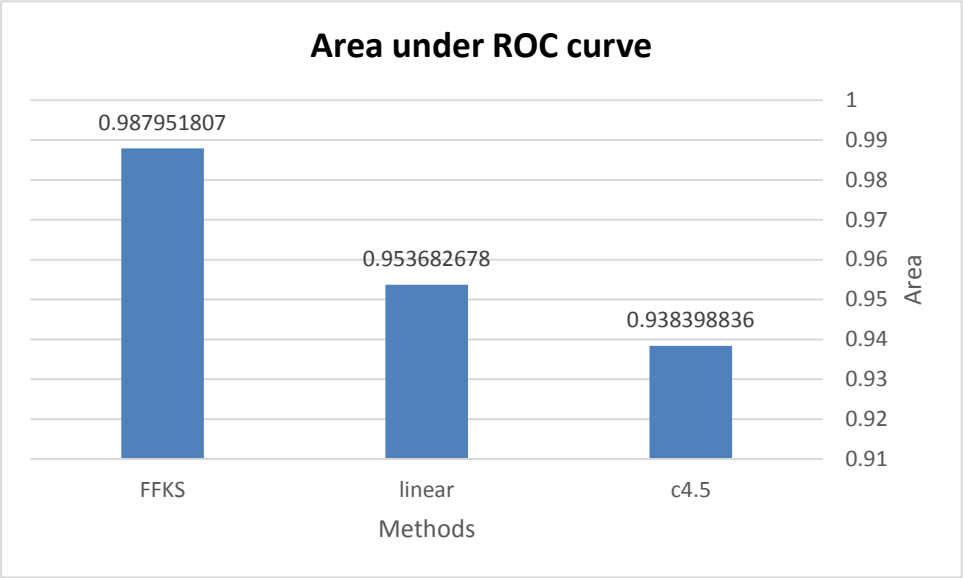


Figure 4. 4 The area under ROC curve

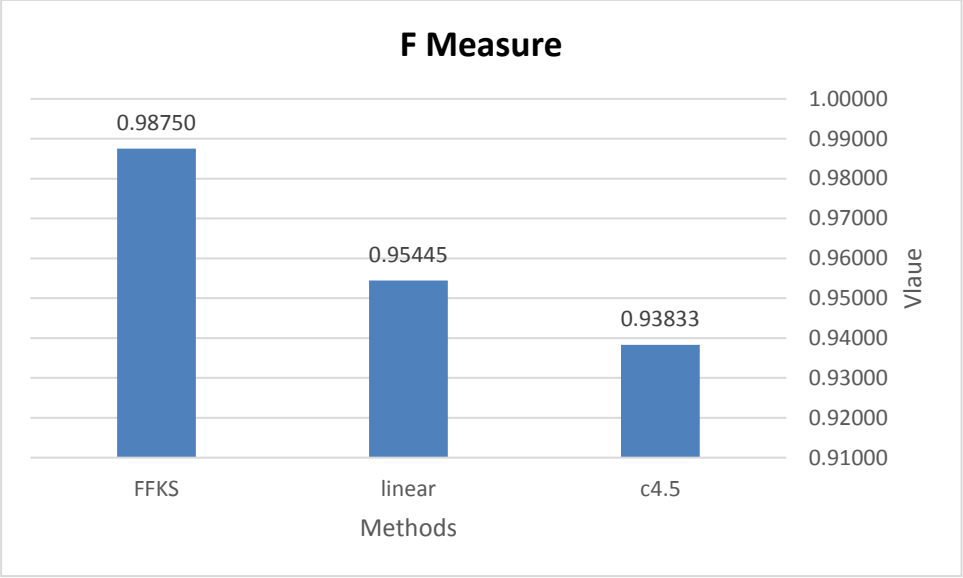


Figure 4. 5 The F measure score

Figure 4.4 exhibited the area under ROC curve of the three classifiers, as the AUC denotes the strength estimator of classifier performance. The proposed ADeSNN proved that it is the best

among the others. In addition, Figure 4.5 displayed that the results of the f-measure which overcame the other two classifiers C4.5 and the linear decision function. As a result, all those measures proved that our contribution of the adaptation revealed that the performance of the ADeSNN was enhanced, and leads us to a new version of the spiking neural network that will help solve the problem of fast flux domains .Figure 4.6 depicts the error estimation measure which is the RMSE.

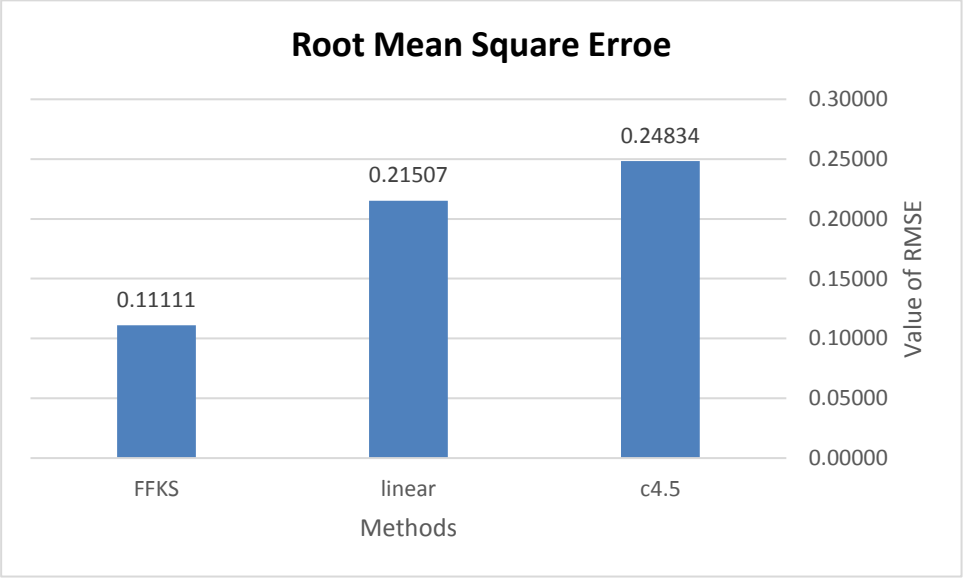


Figure 4. 6 The root mean square error

As shown in Figure 4.6 the error measure indicated that the proposed algorithm over performed the two other methods by almost 50%, which achieves an enhancement of the misclassified instances as RMSE measures the differences between the actual and the estimated targets. So, as the FFKA obtained better results, this means that it will be more accurate to deal with classification problems in an efficient way. The parameters of the ADeSNN algorithm set in the experiment are shown in Table 4.3.

Table 4. 3 The parameters of the ADeSNN algorithm used in the experiment.

***Neurons and synapses* equations parameters**	**Value**	**Unit**
Excitatory synapse time constant (tau_exc)	2	Ms
Inhibitory synapse time constant (tau_inh)	5	Ms
Neuron time constant (tau_mem)	20	Ms
Membrane leak (El)	20	mV
Spike threshold (Vthr)	800	mV
Reset value (Vrst)	0	mV
Fixed inhibitory weight (winh)	0.20	V
Fixed excitatory weight (wexc)	0.40	V
Thermal voltage (UT)	25	mV
Refractory period (refr)	4	Ms
SDSP parameters		
Up/Down weight jumps (Vthm)	0.75*Vthr	mV
Calcium variable time constant (tau_ca)	5 *tau mem	Ms
Steady-state asymptote for Calcium variable (wca)	50	mV
Stop-learning threshold 1 (stop if *V*ca < thk1)	1.7 × wca	mV
Stop-learning threshold 2 (stop LTD if *V*ca > thk2)	2.2 × wca	mV
Stop-learning threshold 2 (stop LTP if *V*ca > thk3)	8 × (wca–wca)	mV
Plastic synapse (NMDA) time constant	9	Ms
Plastic synapse high value (wp hi)	6	mV
Plastic synapse low value (wp lo)	0	mV
Bistability drift	0.25	
Delta weight	0.12 × wp_hi	mV
Input size	5130 spike train	
Simulation time	40	ms
Default clock unit	0.2	Ms

4.7.3 Feature Set Discussion

The feature set used gave excellent results with the adaptive approach. On the other hand, some experiments were performed in order to check the influence of these feature set on the whole detection process. Based on that, three experiments were conducted. Every experiment was implemented by deleting one of the three new proposed features, keeping the other used features the same for the all experiments. The results of these experiments are illustrated in Table 4.4.

Table 4. 4 The results of the three experiments by eliminating one new feature at a time.

	The three Experiments *without* feature of			
Evaluation measures	**AVGSIM**	**Qtime**	**Msgs**	**ADeSNN(ALL)**
FNR	**0**	**0**	**0**	**0**
FPR	**0.283950617**	**0**	**0.209876543**	**0.02410**
TPR	**1**	**1**	**1**	**1**
TNR	**0.716049383**	**1**	**0.790123457**	**0.97590**
ACC	**0.858024691**	**1**	**0.895061728**	**0.98765**
Precision	**0.716049383**	**1**	**0.790123457**	**0.97531**
Recall	**1**	**1**	**1**	**1**
F-Measure	**0.834532374**	**1**	**0.882758621**	**0.98750**
MCC	**0.746787994**	**1**	**0.808122036**	**0.97561**
RMSE	**0.37679611**	**0**	**0.323941772**	**0.11111**
NDEI	**0.752428371**	**0**	**0.646882951**	**0.22188**
MSE	**0.141975309**	**0**	**0.104938272**	**0.01235**

According to Table 4.4, the first experiment eliminated the feature (AVGSIM) from the feature set, then implemented the ADeSNN algorithm on the other sixth features, the accuracy was almost about 85.8%. Comparing this results with the others it is clear that implementing the algorithm while excluding this feature will give low detection rate, so this indicates that the AVGSIM played important role in classifying the input instances, as depicted in Figure 4.7.

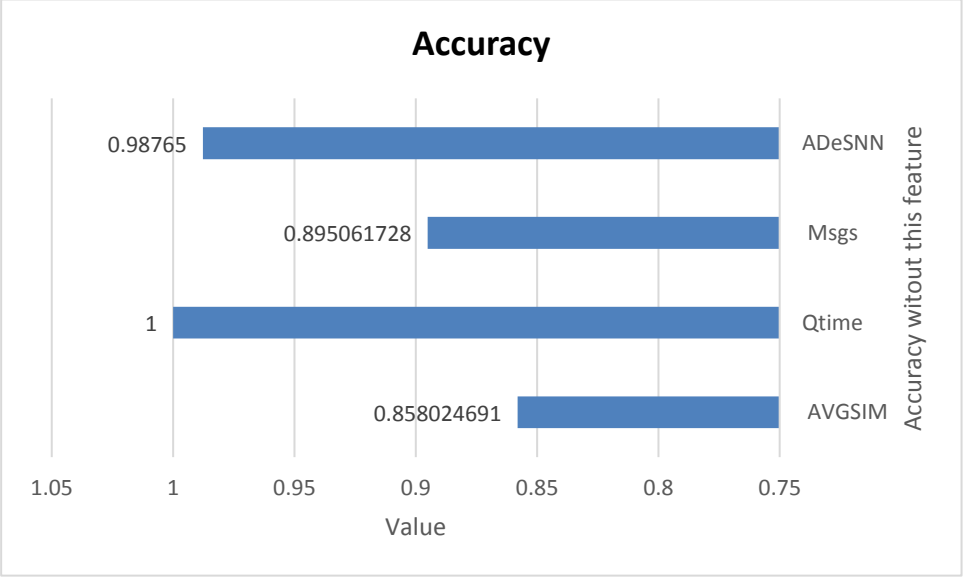


Figure 4. 7 The accuracy comparison between the feature set experiments

By looking at the results of the second experiment shown in the Figure 4.7, it is obvious that the detection rate reached 100%. Which tells us that this feature (Qtime) affects badly on the classification process. In addition, the 3rd experiment that was implemented without the message size showed an accuracy of nearly 89.5%. Actually, this is adequate as the experiment ran including Qtime feature, which seemed the worst feature among them all. The last column in Table 4.4 showed the result of the ADeSNN algorithm with all features included. Here we can say that the accuracy of 98.76% is excellent, considering Qtime was one of its features.

The root Mean Square Error was also computed for all three experiments. Figure 4.8 shows that the experiment excluded the average similarity feature was the highest RMSE value. This proves that the average similarity feature is important to the classification process, while its absence will increase the number of misclassified results. On the other hand, the query time feature showed bad results, where the absence of this feature gave 0 value for RMSE, which is excellent for the classification process as no instances will be missed.

Finally, the combination of the features together, even with including the query time feature, still produces good results, as depicted in Figure 4.8.

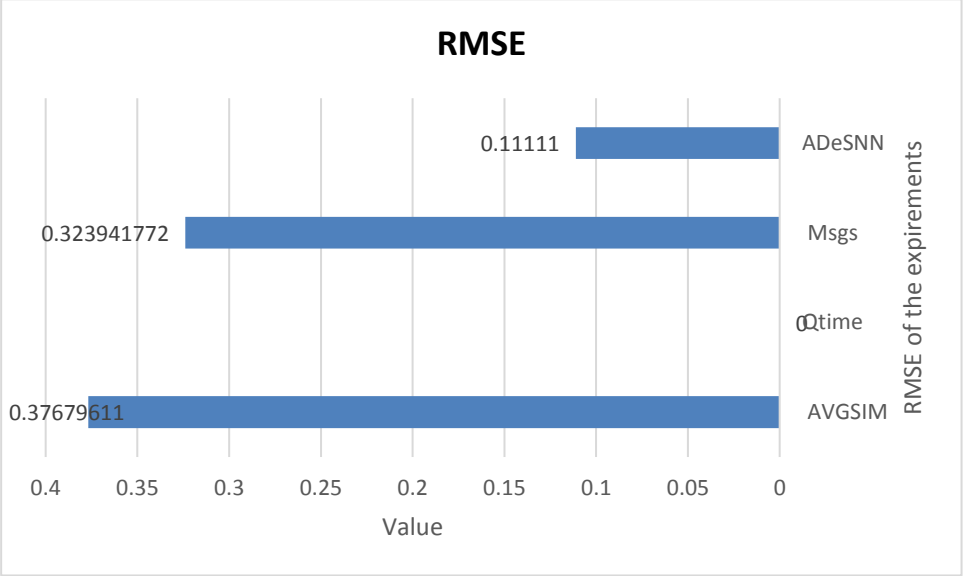


Figure 4. 8 The RMSE of the feature set experiment

Another test was implemented to prove the quality of the feature set was the feature selection method based on the decision tree (Alauthaman, Aslam, Zhang, Alasem, & Hossain, 2016; Kira & Rendell, 1992), for more details about the feature selection method see section 4.5. Table 4.5 shows the result of the current feature set.

Table 4. 5 The features set ranking importance

Feature	Important rate
Msgs	100
IPans	4.01
AVGSIM	3.45
NSadd	1.94
NASN_ans	1.75
NASN_add	1.13
Qtime	0.58

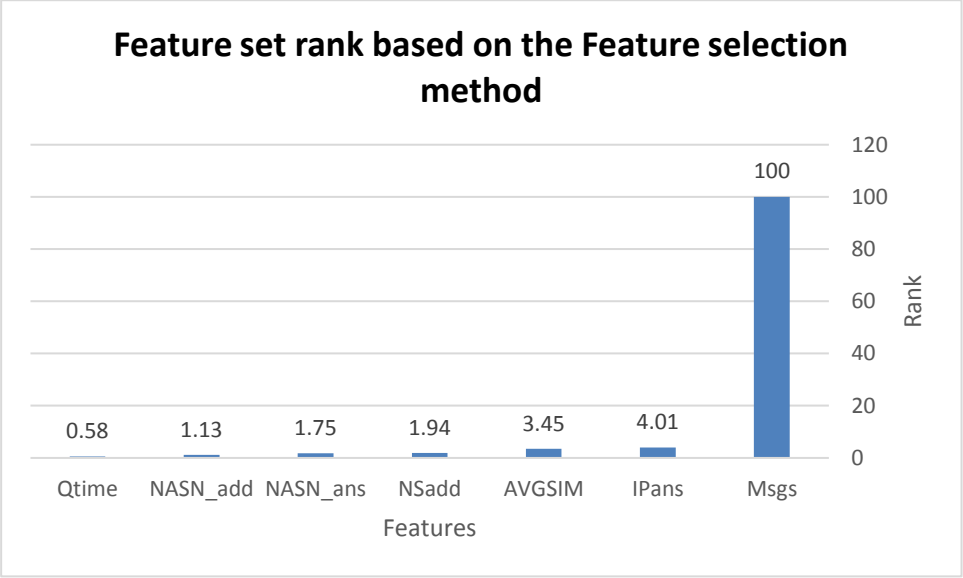


Figure 4. 9 Feature set ranking based on the feature selection method

According to the result of the features selection method in Figure 4.9, the results emphasized the previous results regarding feature quality, and indicated that Qtime is the least important feature among the selected and proposed features. Also, the AVGSIM came in the third position, and this shows its importance compared to the other features.

4.8 Conclusion

This chapter displayed the enhancement of the proposed supervised FFKA by exploiting ADeSNN to detect the fast flux domains in offline mode. The purpose of this chapter was to show the improvement in the performance of the FFKA approach based on the similarity measure used to classify the fast flux domain and the benign domains, as well as the feature set that facilitated the classification process.

In order to evaluate the proposed adaptation of the supervised FFKA phase, a public fast flux dataset was used. Three experiments were conducted, the first was the proposed supervised phase of the FFKA and the other two methods were chosen from the same field of the fast flux detection approaches, the linear decision function and C4.5 classifiers.

The discussion started with the stages of preparing the dataset and the process of the feature extraction. The fast flux public dataset found that as stacks of DNS responses, a script of python was written to extract the features and store it in a secondary database, and then began building and calculating some of the features that needed calculations. A local copy of the ASN database was also used to speed up the process of IP address information retrieval.

The feature set of the all three experiments was tested using the feature selection method, this later used a decision tree to rank the importance of the features on board. The proposed three features have been tested as well, the Qtime feature showed the worse influence on the classification and detection processes. In contrast, the average similarity feature has the best influence on the classification process. Another experiment proved the same results: the three proposed features were tested in three different experiments. Each experiment has to delete one of these three features and record the result of the classification. At the end of the experiment the results had the same indication of the feature selection method.

The supervised phase trained the ADeSNN to detect the fast flux domains. The output of the supervised phase was the final weights and the classification threshold, where the weights stored at the weight repository, as well as the classification threshold. Moreover, the weight repository and the threshold helped to save more memory storage as no need to store all inputs forever, just a particular space to store a certain number of inputs is required.

The results of the FFKA were compared with linear decision function and C4.5 classifiers from the previous related works. Overall, the performance of the FFKA over perform both of them based on previously discussed accuracy measures. Overall, the current results based on the comparisons have made were promising to move forward and added a value to the process of fast flux domains detection. Where the Overall accuracy of the FFKA to detect fast flux domains was (98.76%).

CHAPTER FIVE

HYBRID FAST FLUX KILLER APPROACH

Chapter Overview

This chapter presents the two parts of the fast flux killer approach namely the supervised and the unsupervised phases. The supervised phase trains the adaptive dynamic evolving spiking neural network at the beginning, sets the classification threshold, and stores the weights in the weights repository. The unsupervised phase detects and classifies the zero-day domains based on the output of the supervised phase. The evaluation of the proposed FFKA approach in this chapter is then compared with supervised phase results presented in chapter four. The result will be discussed at the end of this chapter.

5.1 Introduction

The improvements presented the last two chapters were substantial based on the enhanced performance obtained from the ADeSNN algorithm. The improvements enhanced the performance of the algorithm in different ways. Improvements were observed in the initial weights, the similarity classification measure, and the features set. Other slight improvements made over the ADeSNN algorithm focused on the parameter customization problem, which was discussed a part of the parameters adjustment.

There is confusion in the community about the meaning of online detection. The offline detection approaches for example trained on data once and started to detect the incoming data. If, however, some of the new data kept changing, problems may occur. This leads us to search for a new model which is trained on offline dataset and adapts itself for the new incoming data, which is called the online model. In our field, online is concerned with dealing with the new fast flux domain threats once seen.

The proposed FFKA approach deals with two phases in order to detect the fast flux domains, the first is the supervised learning phase which works offline and train the approach to detect fast flux domains, while the second is unsupervised learning phase which works online and

based on the output of the supervised phase, the online learning mode will be able to detect the zero-day fast flux domains.

5.2 The Hybrid Fast Flux Killer Approach (Supervised and unsupervised)

5.2.1 Introduction

At this chapter the FFKA will be discussed in details and this include the supervised and unsupervised phases. We will look at the full life cycle of the FFKA starting from the setting of the bases of the approach at the supervised phase, until the detection of zero-day fast flux domains at the unsupervised phase. Then, a comparison between the output of the supervised phase described in chapter four and the output of the Hybrid approach described in this chapter will be performed.

5.2.2 The FFKA Supervised Phase

In chapter four, we discussed the supervised process in detail. The Supervised phase deals with labelled data, while the process of learning is in progress and the classification threshold is being set. As the process of learning reaches the end, the output of this phase will be the final weights of the output spiking neurons and the classification threshold. The weights are stored in the weights repository and the threshold will be saved as well.

5.2.3 The FFKA Unsupervised Phase

In this section the unsupervised phase will be introduced as part of the FFKA approach. The following sections describe the whole unsupervised phase in details.

5.2.3.1 Introduction

This is the second phase of the proposed FFKA approach, called the unsupervised learning phase. This phase deals with new instances (domains), so that the unsupervised learning part of the approach will be able to deal with unknown data, in our case the zero-day fast flux domains. The classification process will be based on the output of the previous supervised phase as it will be described later on in this chapter.

5.2.3.2 The Preprocessing Stage

The preprocessing stage of the dataset from the DNS responses was introduced in detail in section 4.3.1, the dataset will be prepared, as it will be fed to the unsupervised learning phase one by one.

5.2.3.3 Feature Extraction

This stage is about how to build and calculate some of the features that need calculations. As discussed in details in section 4.3.2, some features need to be calculated based on extra information provided from a third-party database and some other features could be taken straight away from the DNS response message.

5.2.3.4 Hybrid Fast Flux Killer Approach

The two phases of the supervised and the unsupervised learning phases were combined together in order to achieve the main goal of the research, which is the detection of the zero-day fast flux domains in an online mode.

The FFKA approach started with the supervised phase to set the basic seeds of the classification process, then begins the online detection mode to detect the zero-day fast flux domains. After this phase, the supervised learning offline mode is re-executed once again to refine the classification criteria.

As shown in Figure 5.1, the supervised phase receives the labelled inputs one by one and builds the spiking neural network based on the ADeSNN algorithm. As more inputs kept coming the spiking neural network becomes bigger and its learning from the inputs produces the final weights matrix. Subsequently, for each input record there is an output neuron created to capture the input pattern along the learning process. At the end of this phase all the output neurons' weights were stored in the weights repository. Furthermore, during the learning process, the classification threshold was computed to be used in the classification process, which is based on the similarity measure.

The unsupervised mode deals with unlabeled data and the ADeSNN algorithm will capture the features of the domains, then trains the ADeSNN on the new inputs. the algorithm will then accessed the classification threshold stored from the supervised phase to classify the unknown domains. While the new un-labelled records are trained by the ADeSNN in an online mode, the final weights became ready to be stored in the weights repository.

The new weights of the new input records stored at the weights repository will be used later after certain number of records and time. In our case after 1000 records, in supervised learning again to enhance the classification threshold value as the new inputs become part of the training dataset of the supervised phase. This process is depicted in Figure 5.1.

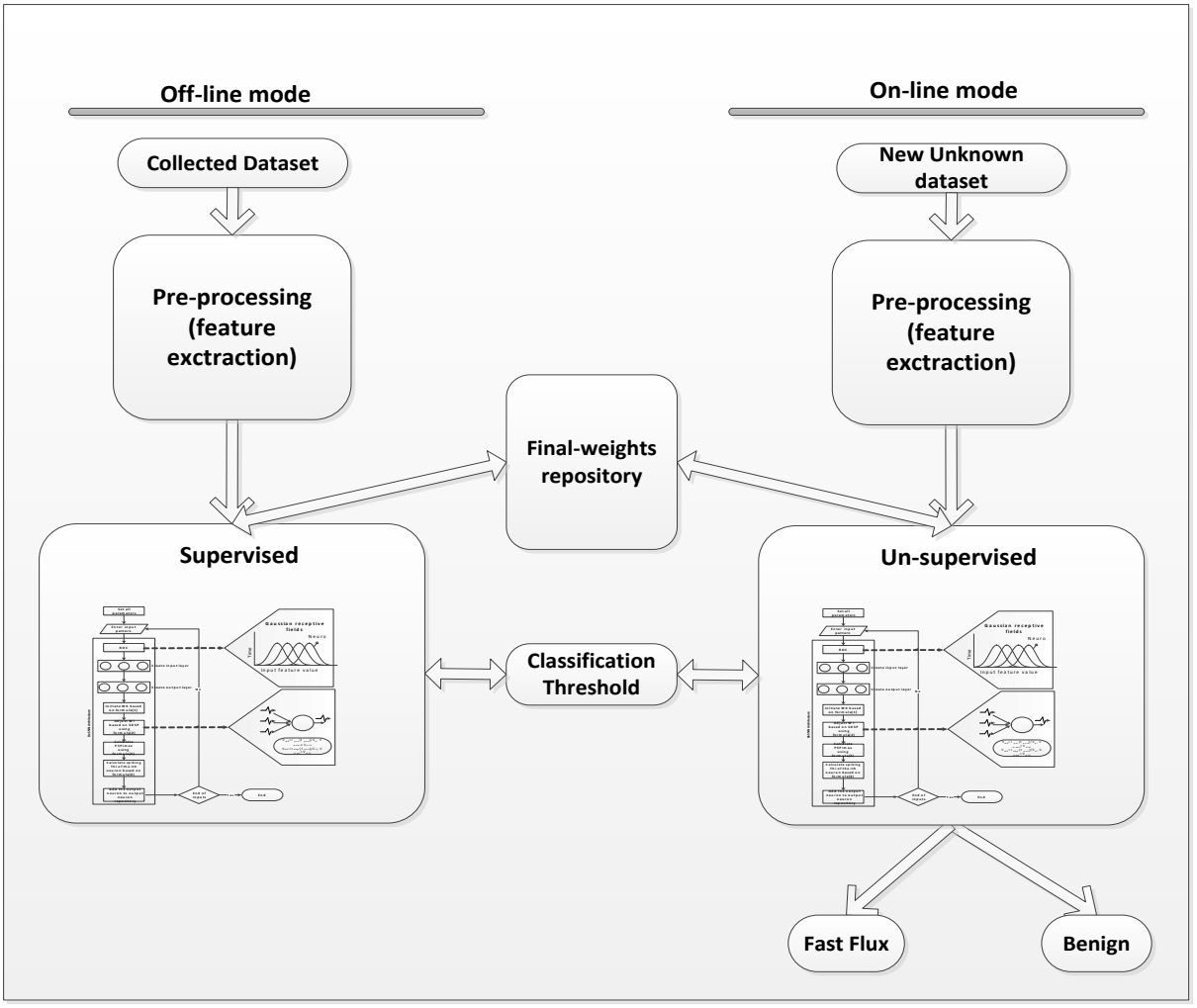


Figure 5. 1 The Hybrid FFKA

The weight repository and the classification threshold helped to save on the memory storage, as there is no need to store all incoming inputs forever but just the specified space to store certain number of inputs is required (1000 records in this case).

5.3 Dataset

The proposed modification on ADeSNN method has to be evaluated and tested by exploiting the public fast flux dataset as discussed in section 4.4.

5.4 Feature Set

A public fast flux data set was used as discussed in section 4.4. So, the feature set proposed here is the same as the one proposed in section 4.6. The seven features remain the same for comparison purposes.

5.5 Experiment and Discussion

Both supervised and unsupervised learning phases are now working at the same time, in the first step the supervised model will train the algorithm on the fast flux data then produce the seeds of the classification process to the next step. The second step, the unsupervised learning phase deals with the new input records and executes the algorithm to produce the final weights then uses the first step results to help the unsupervised phase to classify the new incoming inputs if they are fast flux domains, especially the zero-day fast flux domains.

To achieve that, the public fast flux data set was exploited in order to evaluate the Hybrid FFKA approach and compare the achieved results with the results of the supervised phase alone in chapter four. A 3-folded cross-validation is used and three experiments were conducted for this purpose. Based on that, each experiment in the first two folds is used in the supervised learning to train ADeSNN in the offline mode. While the third fold was fed into the unsupervised learning in an online mode.

At the supervised phase of the approach the first two folds are used to train the ADeSNN algorithm on the benign and fast flux domains, while the running of the spiking neural network the initial weights are updated according to the new inputs, and the final weights of the supervised phase will be stored at the weights repository. At the same time, the classification

threshold trained to classify both the benign and the fast flux domains while training, and the final value of the threshold stored will be used by the next step at the unsupervised phase.

The third fold was fed to the unsupervised phase as un-labelled data inputs, so the algorithm of ADeSNN will execute the spiking neural network and produces the final weights of those inputs, the unsupervised phase has an access to the classification threshold produced from the supervised phase and will be used to classify the new inputs. Then, the new weights are added/replaced in the weights repository.

For every 1000 new input records the approach will re-train the supervised phase on the weights stored in the weights repository to enhance the classification threshold. This leads to the fact that the proposed approach will update its classification ability based on the changing of the input upon time. Which give the approach the lifelong workability, as the functionality adapts to the new changes in the form of what the fast flux domains might do.

5.5.1 The Results of the Hybrid FFKA

Based on the three experiments mentioned in section 5.5, the hybrid FFKA approach worked offline and online in a hybrid mode to detect the fast flux domains in offline mode and trained the FFKA approach to the zero-day fast flux domains in online mode. The average of the three experiments' results are displayed in Table 5.1.

Table 5. 1 Results of the hybrid FFKA

Evaluation measures	**Hybrid FFKA**
FNR	0.00%
FPR	0.59%
TPR	100.00%
TNR	99.41%
ACC	99.54%
Precision	99.41%
Recall	100.00%
F-measure	99.71%
MCC	98.67%
RMSE	6.78%
NDEI	13.55%

According to the results displayed in Table 5.1 the approach accurately classifies the benign domains with a true positive rate of 100%. Additionally, the approach classified the fast flux domains with false positive rate of (0.59%). All the accuracy measures of the detection approach results were displayed in Table 5.1. The error results are displayed in Figure 5.2.

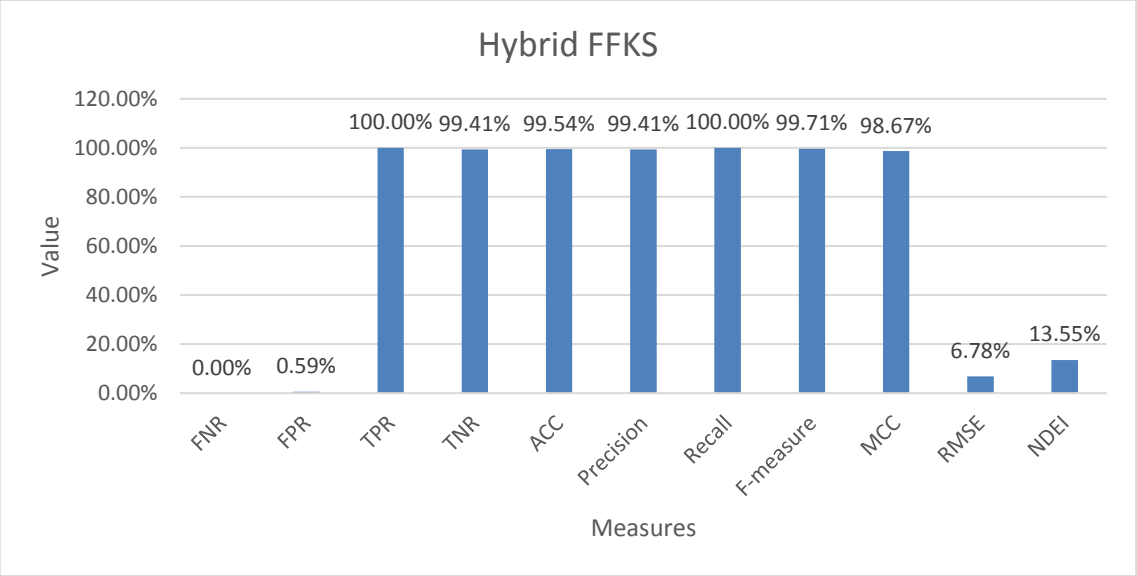


Figure 5. 2 The results of the hybrid FFKA

Figure 5.2 displays the results of the hybrid FFKA approach where the error estimators shows that the approach was able to minimize the number of misclassified instances based on the new contributed enhancements.

The set on parameters used in this experiment is shown in Table 5.2.

Table 5. 2 Parameters used in the hybrid experiment

Neurons and synapses equations parameters	Value	Unit
Excitatory synapse time constant (tau_exc)	2	Ms
Inhibitory synapse time constant (tau_inh)	5	Ms
Neuron time constant (tau_mem)	20	Ms
Membrane leak (El)	20	mV
Spike threshold (Vthr)	800	mV
Reset value (Vrst)	0	mV
Fixed inhibitory weight (winh)	0.20	V
Fixed excitatory weight (wexc)	0.40	V
Thermal voltage (UT)	25	mV
Refractory period (refr)	4	Ms
SDSP parameters		
Up/Down weight jumps (Vthm)	0.75*Vthr	mV
Calcium variable time constant (tau_ca)	5 *tau mem	Ms
Steady-state asymptote for Calcium variable (wca)	50	mV
Stop-learning threshold 1 (stop if *V*ca < thk1)	1.7 × wca	mV
Stop-learning threshold 2 (stop LTD if *V*ca > thk2)	2.2 × wca	mV
Stop-learning threshold 2 (stop LTP if *V*ca > thk3)	8 × (wca–wca)	mV
Plastic synapse (NMDA) time constant	9	Ms
Plastic synapse high value (wp hi)	6	mV
Plastic synapse low value (wp lo)	0	mV
Bistability drift	0.02	
Delta weight	0.12 × wp_hi	mV
Input size	5130 spike train	
Simulation time	40	ms
Default clock unit	0.2	Ms

Overall, the Hybrid FFKA approach proved its ability to detect the zero-day fast flux domains in the online mode where the total accuracy achieved was 99.54%, and enhanced the classification accuracy in offline mode periodically.

5.5.2 Comparison of Supervised and Hybrid approach

In chapter 4, the supervised FFKA approach was introduce to train the approach in detecting the fast flux domain and produced and enhanced (later) the classification threshold of the unsupervised phase.

The following discussion is about the comparison between the supervised FFKA approach phase from chapter 5 and the hybrid (supervised and unsupervised) FFKA approach. A public dataset was used to test the chosen classifiers in section 4.4. Three experiments were conducted using the mentioned fast flux public dataset to test the ability of the proposed FFKA approach to solve fast flux domains problem. The performance and the results of the experiments were promising and indicated an increase in the detection accuracy of fast flux domains. To ensure the quality of the learnning phase of the ADeSNN, a 3-folded cross-validation method is used to estimate the error rate of the algorithm. The three experiments were implemented then the average has been taken. Table 5.3 summarizes the results obtained in this experiments those obtained from the experiments in chapter 4.

Table 5. 3 The comparison results of supervised and hybrid FFKA approach

Evaluation measures	**Supervised FFKA**	**Hybrid FFKA**
FNR	0.00%	0.00%
FPR	2.41%	0.59%
TPR	100.00%	100.00%
TNR	97.59%	99.41%
ACC	98.77%	99.54%
Precision	97.53%	99.41%
Recall	100.00%	100.00%
F-measure	98.75%	99.71%
MCC	97.56%	98.67%
RMSE	11.11%	6.78%
NDEI	22.19%	13.55%

Table 5.3 shows the result of the comparison between the supervised phase experiment from chapter 4 and the hybrid FFKA experiment discussed in section 5.5.1.

By looking at the results, both sides shared the same achievement in detecting the benign domains where the detection rate of the benign domains was 100%. But, in the case of detecting the fast flux domains the hybrid approach performs better than the supervised phase result to achieve a detection rate of 99.41% while the supervised achieved 97.59%. Other measures were

also used as shown in Table 5.3 and include Precision, Recall, F-measure, and MCC. All the measures show that the hybrid approach outperforms the supervised one.

Figure 5.3 displays the compared graph that shows all the measures in both the supervised experiment and the hybrid FFKA approach experiment.

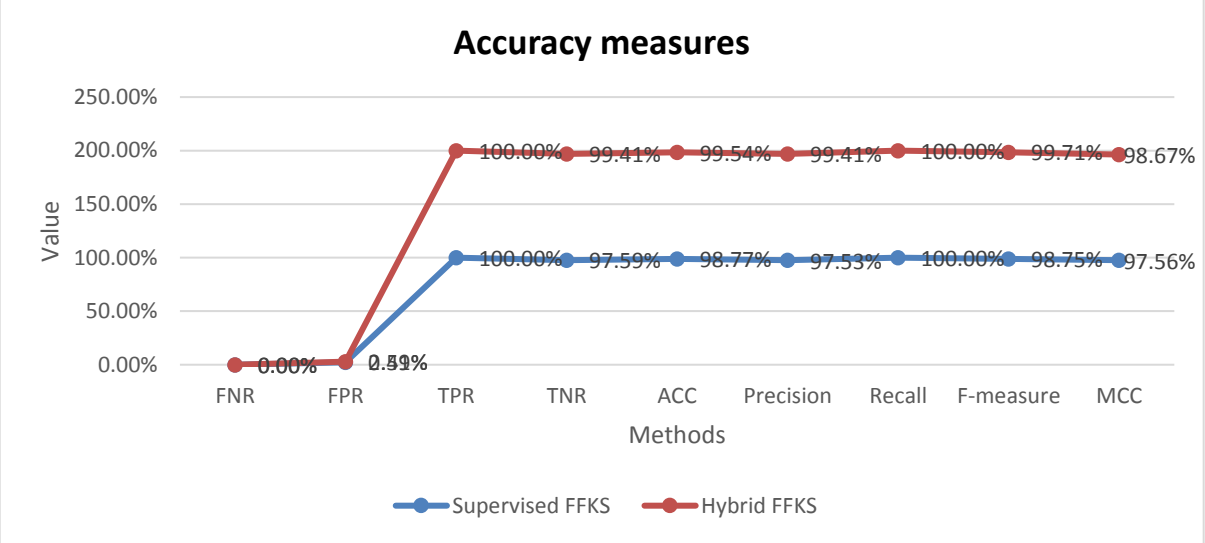


Figure 5. 3 Comparison of the supervised and hybrid FFKA approaches

In the case of the error estimator's performance, RMSE and NDEI exhibited an improvement in their values compared to the same in the supervised phase, which leads to conclude that the percentage of an error in misclassifying the normal and fast flux domains decreased in the hybrid approach, as depicted in Figure 5.4.

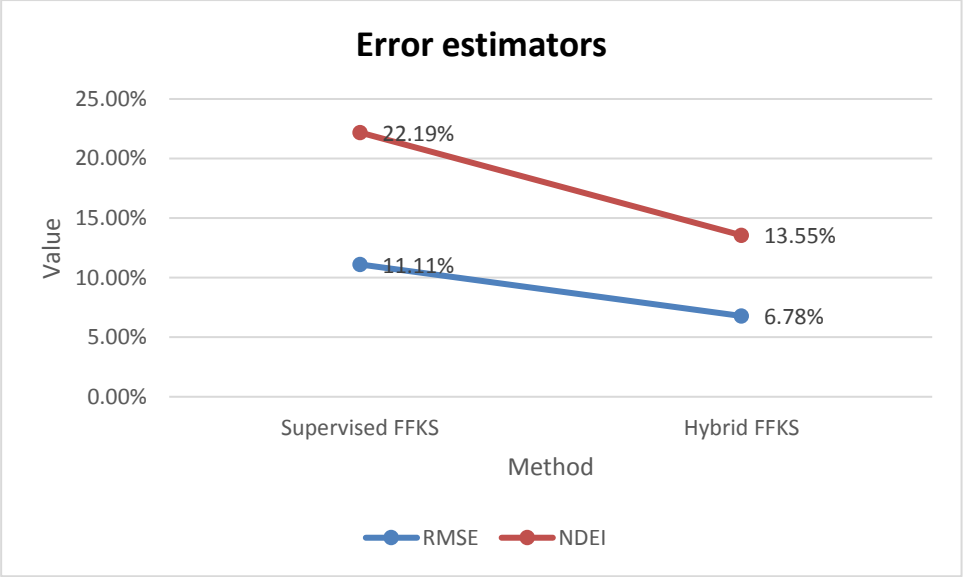


Figure 5. 4 The error comparison of the supervised and hybrid FFKA approach

Overall, the proposed contribution of the hybrid FFKA approach improved the detection accuracy and ability to detect the fast flux domains and especially the zero-day fast flux domains in online mode. Furthermore, the Precision and F-measure showed an enhancement in their values compared to the last experiment results. We conclude that the detection accuracy of the proposed hybrid FFKA approach has improved the ability of the ADeSNN algorithm to classify the incoming inputs more correctly than before.

5.5.3 Parameter Adjustment and Customization

The algorithm of the DeSNN suffers from the many parameters needed to be set before running the algorithm. Our adaptive ADeSNN modifies the process of the initial weight setting, so this adaptation added a value in the parameters customization problem by excluding the (Mod) parameter as shown in section 3.4. According to (Kasabov et al., 2013) the best values of the parameters were given in their research is the same as those shown in Table 5.4 and the same parameters are used in the current research for the sake of consistency and comparison purposes. However, the best results obtained were based on the parameters' values of the current research as displayed in Table 5.4.

Table 5. 4 The parameters values of the ADeSNN algorithm in FFKA approach

Neurons and synapses equations parameters	Value	Unit
Excitatory synapse time constant (tau_exc)	2	Ms
Inhibitory synapse time constant (tau_inh)	5	Ms
Neuron time constant (tau_mem)	20	Ms
Membrane leak (El)	20	mV
Spike threshold (Vthr)	800	mV
Reset value (Vrst)	0	mV
Fixed inhibitory weight (winh)	0.20	V
Fixed excitatory weight (wexc)	0.40	V
Thermal voltage (UT)	25	mV
Refractory period (refr)	4	Ms
SDSP parameters		
Up/Down weight jumps (Vthm)	0.75*Vthr	mV
Calcium variable time constant (tau_ca)	5 *tau mem	Ms
Steady-state asymptote for Calcium variable (wca)	50	mV
Stop-learning threshold 1 (stop if *V*ca < thk1)	1.7 × wca	mV
Stop-learning threshold 2 (stop LTD if *V*ca > thk2)	2.2 × wca	mV
Stop-learning threshold 2 (stop LTP if *V*ca > thk3)	8 × (wca–wca)	mV
Plastic synapse (NMDA) time constant	9	Ms
Plastic synapse high value (wp hi)	6	mV
Plastic synapse low value (wp lo)	0	mV
Bistability drift	0.02	
Delta weight	0.12 × wp_hi	mV
Input size	5130 spike train	
Simulation time	40	ms
Default clock unit	0.2	Ms

5.6 chapter summary

This chapter discussed the main proposed contribution, which is the Fast Flux Killer Approach or FFKA. The improvements made in chapter 3 and chapter 4 were substantial according to the enhancements in the performance of the ADeSNN algorithm. This chapter introduced the hybrid FFKA approach that worked offline to train the approach and initialize the classification threshold, and online to detect the zero-day fast flux domains based on the threshold value that was set by the supervised phase.

The same fast flux public dataset used in chapter 4 was used in the current chapter. Furthermore, a 3 fold cross-validation was used, as the experiments were conducted in the two chapters were then compared according to various accuracy measures and error estimators. The result proved that the enhancement of the hybrid FFKA approach over the supervised phase alone in both the accuracy measures where the total accuracy achieved was 99.54, and the error estimators showed that the hybrid approach had lower error values in misclassifying the domains.

This chapter also discussed the parameter customization problem, by reducing the number of parameters used in the algorithm, for example the (Mod) parameter used in RO initial weight calculations, but it became useless as the new proposed approach used the spike time as initial weight.

Finally, the parameters adjustment was discussed. A comparison between the adaptive version of the algorithm introduced in chapter 4 and the current modification in this chapter was implemented, then we discussed the improvements of the performance of the algorithm based on the same parameters values, as introduced in the original DeSNN as mentioned in algorithm 3.1(Kasabov et al., 2013).

CHAPTER SIX

DISCUSSION, CONCLUSION AND RECOMMENDATION FOR FUTURE WORK

Chapter Overview

This chapter summaries the whole research conducted in this study and give some directions for future researchers to guide them in detecting the fast flux domains as well as improving the performance of the proposed methodology.

6.1 Discussion

This thesis discussed the process of detecting fast flux botnet based on a novel proposed FFKA. Starting from the title the thesis treated two tracks, the fast flux botnet detection and the adaptation of the DeSNN algorithm. Hence, the structure of the thesis was built to develop the adaptation process first, then use the proposed adaptive algorithm to detect the fast flux problem. Based on that, chapter one discussed the fundamentals of the thesis (where the research identified the gap in the knowledge), the motivation of the research, the main aim and the objectives, the methodology of the solution, and the contribution of the proposed research were discussed as well. Chapter two revealed a solid background that covers the subject of the thesis, discussed the literature review, and examined the related work done so far in the same field.

In chapter three, two public datasets were used to evaluate the proposed adaptation on the DeSNN and compared the results with original DeSNN itself. DeSNN algorithm is built based on both the RO learning rules and the SDSP learning rules. According to previous work, the initial weight of the DeSNN is calculated based on the RO rules. As stated in (Kasabov et al., 2013), the output of the DeSNN algorithm consists of the initial and final weight matrices, as a new incoming input pattern arrives, an initial weight and final weight are computed. Then, the updates happened while running the algorithm on the initial weight. At the recall mode the classification of the new arrival is going to be based on testing the similarity measure, which is Euclidean distance in this research. Experiments showed that the current initial weight based on the RO setting introduces a clear

misclassification percentage of detecting the incoming inputs. So, the contribution here was to use the spike time as initial weight and the results obtained were satisfactory. The overall accuracy of DeSNN was (56%) while it was (91.67%) for ADeSNN using on the IRIS dataset. According to the IRIS dataset two classes were non-linearly separable which cause to show almost 91% accuracy while it was tested on the first linearly separable classes and give 100% accuracy. This leads to the ability of the adaptive DeSNN to classify classes even when inputs are mutually mixed. Finally, ADeSNN exhibited higher true positive rate and less false negative rate than DeSNN.

For the second public WDBC dataset, the experiment distributed the dataset into 5-fold cross-validations groups. So, five separated experiments were done, where the instances randomly distributed on the five groups, then the results were computed and the average was taken. It is noted that the error measures (RSME, NDEI, and MSE-ERROR) values of the proposed adaptive ADeSNN were less than those for the original DeSNN, which means the adaptation on the DeSNN will minimize the misclassification of the input instances, and maximize the accuracy of the detection and classification. In addition, the MCC is a performance metric which is widely used in bioinformatics. The two algorithms used this metric as it best deals with imbalanced data, and this leads the researcher to conclude that the adaptive algorithm is more accurate than the original one. Coming to compare the accuracy, the F-measure, Recall, and ACC revealed that the proposed adaptation produced more accurate results (97.16%) than the original DeSNN (76.59%). Finally, ADeSNN exhibited higher true positive rate and less false negative rate than DeSNN. Overall, all the measures used proved the improvement of the performance of the proposed adaptive algorithm.

According to the achieved enhancement in chapter three, which showed that the proposed adaptation on the DeSNN algorithm improved its performance. In chapter four, FFKA was tested in order to detect the fast flux domains in an online mode. The proposed FFKA consists of two parts, the supervised and unsupervised learning modes. The supervised ADeSNN in FFKA is about training the ADeSNN algorithm on both fast flux domains and benign domains. Besides, a threshold of the classification process will be trained along the training process. The outputs of the supervised training mode are the final weight and the classification threshold,

where the weights are stored in the weights repository. Furthermore, the threshold stored to be accessed by both the supervised and un-supervised modes later. The un-supervised mode deals with new unknown data, so the ADeSNN algorithm will capture the features of the domains, then access the value of the classification threshold to classify the inputs domains, then the final weights of the new inputs are calculated. The weights of the new input records then will be stored in the weights repository to be used later after certain (number of records/ time) in supervised learning mode again to enhance the classification threshold value as the new inputs become part of the training dataset. The weight repository and the threshold helped to save memory storage as there is no need to store all inputs forever, but just the specified space to store certain number if inputs is required.

Also, the contribution made over the ADeSNN algorithm focused on the testing criteria where the current research implemented the similarity measure defined by formula 1 in chapter 4, which according to the best of my knowledge, it is the first time this formula has been used in this field. Furthermore, chapter discussed the parameter customization problem by reducing the number of parameters for example the (Mod) parameter used before with the initial weight calculations.

To achieve that, a comparison between the proposed algorithm and two other approaches from the works developed in (Celik & Oktug, 2013; H.-T. Lin et al., 2013) were implemented. A public dataset has been used to test the chosen classifiers, then compares their performance among the related previous works developed in same field. Three experiments were conducted using the mentioned fast flux public dataset, to test the ability of the proposed algorithm to solve the fast flux problem. Two of them were selected based on two related previous researches, the linear decision function in (H.-T. Lin et al., 2013) and the C4.5 in (Celik & Oktug, 2013) algorithm. The third was the supervised FFKA. To ensure the quality of the learnning phase of the ADeSNN, a 3-folded cross-validation method is used to estimate the error rate of the three classifiers. The three experiments were implemented then the average has been taken. According to the implementations the overall accuracy of the linear classifier was (95.37 %). In addition, the linear decision function used as a classifier needs to estimate the categorizer of the linear function, to see if the estimation is good and the linear function work properly, otherwise the error will be high in the classification process. Besides, as C4.5 algorithm is considered as

a supervised learning algorithm, it could not be used to discover the unknown attacks especially the zero-day fast flux domains. Moreover, according to the implementation, the accuracy was not high as stated in their paper at 93.38%. When this result and the previous linear results were compared to the current proposed ADeSNN, it shows that the proposed approach outperforms the two methods with a total detection accuracy of 98.76%. Other accuracy measures have also been implemented.

The feature set proposed by the current thesis consists of several features, where three of them are for the best of our knowledge the first time to be used in the field. AVGSIM is a new feature that computes the average of the similarity between the ASN number of the requested IP address and the other ASN number of the returned IP addresses of the DNS response. In other words, AVGSIM refers to the average of the similarity between the autonomous system number of the user's IP and autonomous system numbers of the proxy bots returned in the answer section of the DNS response, and is computed using formula 1 in chapter 4. The other two new features are the time of the query and the DNS packet size. All these features together showed their ability to distinguish between the fast flux network (served by domains) and the legitimate domains. The feature set used gave excellent results with the adaptive approach. Further experiments were conducted to check the influence of these feature set on the whole detection process. Based on that, three experiments were conducted, every experiment is implemented by deleting one of the new three features with keeping the other used features the same for the all experiments. According to feature evaluation results, the first experiment eliminated the feature (AVGSIM) from the feature set, then implemented the ADeSNN algorithm on the other six features, the accuracy was about 85.8%. Comparing this result with the others it is clearly that implementing the algorithm with excluding this feature will give low detection rate, so this indicate that the AVGSIM played an important role in classifying the input instances. By reading the results of the second feature experiment, it was noted that the detection rate reached 100%. This tells us that the feature (Qtime) badly affects the classification process. In addition, the 3rd experiment showed an accuracy of almost 89.5% which is actually good enough as the experiment ran includs the Qtime feature. The last column in Table 4.4 showed the result of the ADeSNN algorithm with all features included. Here we can say that the accuracy of 98.77 is excellent, considering it had Qtime as one of its features.

Another test was implemented to prove the quality of the feature set was the feature selection method based on decision trees (Alauthaman et al., 2016; Kira & Rendell, 1992), the result of the features selection method emphasized the previous results regarding the Qtime feature. In short, Qtime is the least important feature among the selected features. The AVGSIM came in the third position during this test, and this shows its importance among the other features.

Chapter five discussed the main proposed contribution which is the Fast Flux Killer Approach FFKA. In chapter 3 and chapter 4 many improvements were proposed and proved their enhancements in the performance of the ADeSNN algorithm. This chapter introduced the hybrid FFKA approach that worked offline to train the approach and initialize the classification threshold, and worked online to detect the zero-day fast flux domains based on the threshold value that was set by the supervised phase. These two supervised and unsupervised phases play important roles in enhancing the detection performance of the proposed approach to detect the fast flux domains and especially the zero-day domains, as the approach re-trained the algorithm based on the old and the new data. This gives the approach the ability to adapt itself to whatever new changes the fast flux domains will implement to evade detection.

The same fast flux public dataset used in chapter 4 was used in the chapter 5. A 3 fold cross-validation was used as the experiments that were conducted in the two chapters and had their results compared according to various accuracy measures and error estimators. The result proved the the enhancement of the hybrid FFKA approach over the supervised phase alone in both the accuracy measures where the total accuracy achieved was 99.54, and the error estimators showed that the hybrid approach had lower error values in misclassified the domains.

Overall, the proposed adaptation and modification have improved the performance of the original algorithm and obtained better classification results.

6.2 Limitations and Future Work

The DeSNN algorithm suffers from the fact that many parameters have to be set before running it. The proposed contribution of this work has partially solved this problem, but still there are many parameters that need to be set. It is clear that the problem of fast flux is not solved and it needs several efforts to be gathered together at different levels. Governmental, private sector,

and research efforts have to be implemented and coordinated. This is because of the need to acquire a real-time dataset for a long period of time, and ensure that the dataset is controlled and tested correctly.

The suggested future work for saving more memory usage could be the use of fuzzy rules to be saved instead of the weight matrices, as each current cluster is dedicated for one weight matrix.

6.3 Conclusion

Botnets have expanded radically and is an interesting research field that concerns expertise based on the threats that it provided, fast flux botnets offer a bridge to carry other malicious threats such as DDoS, internet fraud, and identity thief. Although several methods have been suggested for detecting fast flux domains, they still have low detection accuracy, especially with the zero-day domain, quite a long detection time, and consume high memory storage.

The main contribution of this study is to come out with a approach for the detection and classification of fast flux domains. So, we proposed a new approach called Fast Flux Killer Approach (FFKA) that has the ability to detect FF-Domains, especially the zero-day domains in online mode, with an implementation constructed on Adaptive Dynamic evolving Spiking Neural Network (ADeSNN). The proposed approach proved its ability to detect fast flux domains with high detection accuracy according to the experiments have implemented.

The aspects were considered and addressed in this study has contributed scientifically to the field in many ways. Most of previous studies in the fast flux domains detection field were based on machine learning algorithms, stand-alone approaches, and network monitoring. The research was conducted through its contributions as presented in section 1.6 as follows.

- The first and second objectives were developed as described in chapters 3. The contribution focused on increasing the detection performance using adaptive fast one-pass algorithm (ADeSNN). By employing the spike time as initial weight, then the achieved performance evaluated according to true positive, true negative, recall, precision, f-Measure and overall accuracy.

- The third and fourth objective was also achieved with the proposed supervised FFKA approach phase as described in chapter 4. This contribution represents the most important part of this research that aimed to improve the detection accuracy, especially the classification criteria by conducting the similarity measure between the new and already trained inputs. Also, design of a new feature set which can be used with suggested algorithm to accurately classify fast flux domains.
- The fifth and sixth contributions were developed in chapter 4. Where a proposed Hybrid FFKA method was proposed based on an adaptive life-long learning approach able to detect dynamically the unknown zero-day fast flux domains in online mode and enhance the classification process in offline mode. Also, a new adaptive dynamic classification threshold was introduced in order to classify new incoming inputs, as well as minimize the memory storage used.

This comparisons stated that the proposed approach outperformed the other recently developed approaches. Three public datasets are exploited in the experiments to show the effects of the adaptation of the DeSNN algorithm, a high detection accuracy achieved of detecting fast flux domains especially the zero-day domains was about (99.54%) in an online mode.

References

Al-Bataineh, A., & White, G. (2012). *Analysis and detection of malicious data exfiltration in web traffic.* Paper presented at the Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on.

Al-Duwairi, B. N., & Al-Hammouri, A. T. (2014). Fast Flux Watch: A mechanism for online detection of fast flux networks. *Journal of Advanced Research, 5*(4), 473-479. doi:http://dx.doi.org/10.1016/j.jare.2014.01.002

Alauthaman, M., Aslam, N., Zhang, L., Alasem, R., & Hossain, M. (2016). A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks. *Neural Computing and Applications*, 1-14.

Alauthman, M., & Almomani, O. (2017). A proposed framework for Botnet Spam-email Filtering using Neucube.

Alexa. Retrieved from: https://www.kaggle.com/cheedcheed/top1m

Aličković, E., & Subasi, A. (2017). Breast cancer diagnosis using GA feature selection and Rotation Forest. *Neural Computing and Applications, 28*(4), 753-763.

Alieyan, K., ALmomani, A., Manasrah, A., & Kadhum, M. M. (2015). A survey of botnet detection based on DNS. *Neural Computing and Applications*, 1-18.

Alkhazaleh, S., Salleh, A. R., & Hassan, N. (2011). Possibility fuzzy soft set. *Advances in Decision Sciences, 2011*.

Almomani, A., Gupta, B., Atawneh, S., Meulenberg, A., & Almomani, E. (2013). A survey of phishing email filtering techniques. *Communications Surveys & Tutorials, IEEE, 15*(4), 2070-2090.

Almomani, A., Obeidat, A., Alsaedi, K., Obaida, M. A.-H., & Al-Betar, M. (2015). Spam E-mail Filtering using ECOS Algorithms. *Indian Journal of Science and Technology, 8*(S9), 260-272.

Almomani, A., Wan, T.-C., Manasrah, A., Altaher, A., Baklizi, M., & Ramadass, S. (2013). An enhanced online phishing e-mail detection framework based on evolving connectionist system. *International Journal of Innovative Computing, Information and Control (IJICIC), 9*(3), 169-175.

Alvi, F. B., Pears, R., & Kasabov, N. (2017). An evolving spatio-temporal approach for gender and age group classification with Spiking Neural Networks. *Evolving Systems*, 1-12.

ATLAS URL: https://www.arbornetworks.com/atlas-portal.

Balepin, I., Maltsev, S., Rowe, J., & Levitt, K. (2003). *Using specification-based intrusion detection for automated response.* Paper presented at the International Workshop on Recent Advances in Intrusion Detection.

Barford, P., & Yegneswaran, V. (2007). An inside look at botnets *Malware Detection* (pp. 171-191): Springer.

Barra, S., Casanova, A., Narducci, F., & Ricciardi, S. (2015). Ubiquitous iris recognition by means of mobile devices. *Pattern Recognition Letters, 57*, 66-73.

Basu, A., Roy, R., & Savitha, N. (2018). Performance Analysis of Regression and Classification Models in the Prediction of Breast Cancer. *Indian Journal of Science and Technology, 11*(3).

Benjamin, Z., & R.A., F. (2013). *Iris DataSet*.

Boutemedjet, S., Bouguila, N., & Ziou, D. (2009). A hybrid feature extraction selection approach for high-dimensional non-Gaussian data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(8), 1429-1443.

Buhariwala, K. (2011). Geo-locating Hidden Servers Behind Fast-Flux Proxies.

Caglayan, A., Toothaker, M., Drapaeau, D., Burke, D., & Eaton, G. (2010, 5-8 Jan. 2010). *Behavioral Patterns of Fast Flux Service Networks.* Paper presented at the System Sciences (HICSS), 2010 43rd Hawaii International Conference on.

Caglayan, A., Toothaker, M., Drapeau, D., Burke, D., & Eaton, G. (2009, 3-4 March 2009). *Real-Time Detection of Fast Flux Service Networks.* Paper presented at the Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications & Technology.

Castelluccia, C., Kaafar, M. A., Manils, P., & Perito, D. (2009). *Geolocalization of proxied services and its application to fast-flux hidden servers.* Paper presented at the Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference.

Celik, Z. B., & Oktug, S. (2013). *Detection of fast-flux networks using various dns feature sets.* Paper presented at the Computers and Communications (ISCC), 2013 IEEE Symposium on.

Chahal, P. S., & Khurana, S. S. (2016). TempR: Application of Stricture Dependent Intelligent Classifier for Fast Flux Domain Detection. *International Journal of Computer Network & Information Security, 8*(10).

Chen, C.-M., Cheng, S.-T., & Chou, J.-H. (2013). Detection of Fast-Flux Domains. *Journal of Advances in Computer Networks, 1*(2).

Chen, C.-M., Huang, M.-Z., & Ou, Y.-H. (2014). Detecting Hybrid Botnets with Web Command and Control Servers or Fast Flux Domain.

Chen, Z., Wang, J., Zhou, Y., & Li, C. (2011). An improvement for fast-flux service networks detection based on data mining techniques *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing* (pp. 302-309): Springer.

D. Kevin McGrath, A. K., Minaxi Gupta. (2009/09/01). Phishing Infrastructure Fluxes All the Way. *IEEE Security & Privacy, 7*(5), 8.

Dagon, D., Gu, G., & Lee, C. (2008). A Taxonomy of Botnet Structures *Botnet Detection: Countering the Largest Security Threa* (Vol. 36, pp. 143-164). Boston: Springer US.

Dave Marcus, R. S. (2012). Dissecting operation high roller. *McAfee and Guardian Analytics, white paper, June*.

Deeb Al-Mo, A. A., Wan, T.-C., Al-Saedi, K., Altaher, A., Ramadass, S., Manasrah, A., . . . Anbar, M. (2011). An online model on evolving phishing e-mail detection and classification method. *Journal of Applied Sciences, 11*, 3301-3307.

Demertzis, K., & Iliadis, L. (2015). *Evolving Smart URL Filter in a Zone-Based Policy Firewall for Detecting Algorithmically Generated Malicious Domains.* Paper presented at the International Symposium on Statistical Learning and Data Sciences.

DNSBL URL: https://www.zerobounce.net/.

Doborjeh, M. G., Capecci, E., & Kasabov, N. (2014). *Classification and segmentation of fMRI spatio-temporal brain data with a NeuCube evolving spiking neural network model.* Paper presented at the Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on.

Doborjeh, M. G., & Kasabov, N. (2016). *Personalised modelling on integrated clinical and EEG spatio-temporal brain data in the NeuCube spiking neural network system.* Paper presented at the Neural Networks (IJCNN), 2016 International Joint Conference on.

Dua, D. a. K. T., E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Enterprise, H.-P. (2015). 2015 Cost of Cyber Crime Study: Global. Retrieved from http://engage.hpe.com/LP_510004609_HPSW-ESP_WW_EN-US_PonemonGate

Espinosa, J., & Vandewalle, J. (2000). Constructing fuzzy models with linguistic integrity from numerical data-AFRELI algorithm. *IEEE Transactions on Fuzzy Systems, 8*(5). doi:10.1109/91.873582

Fabian, M., & Terzis, M. A. (2007). *My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging.* Paper presented at the Proceedings of the 1st USENIX Workshop on Hot Topics in Understanding Botnets, Cambridge, USA.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861-874. doi:10.1016/j.patrec.2005.10.010

Futai, Z., Siyu, Z., & Weixiong, R. (2013). Hybrid detection and tracking of fast-flux botnet on domain name system traffic. *Communications, China, 10*(11), 81-94.

Gasster, L. (2008). GNSO issues report on fast flux hosting. *Issue Report on Fast Flux Hosting*.

Giacinto, G., Roli, F., & Didaci, L. (2003). Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern Recognition Letters, 24*(12), 1795-1803.

GNSO Fast Flux Hosting Working Group Publishes Final Report. (7 August 2009).

Gothai, E., & Balasubramanie, P. (2012). An Efficient Way for Clustering Using Alternative Decision Tree. *American Journal of Applied Sciences, 9*(4), 531.

Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B. B., & Dagon, D. (2007). *Peer-to-peer botnets: Overview and case study.* Paper presented at the Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets.

Gržnić, T., Perhoč, D., Marić, M., Vlašić, F., & Kulcsar, T. (2014). *CROFlux—Passive DNS method for detecting fast-flux domains.* Paper presented at the Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on.

Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). *BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection.* Paper presented at the USENIX Security Symposium.

Hagras, H., Pounds-Cornish, A., Colley, M., Callaghan, V., & Clarke, G. (2004). *Evolving spiking neural network controllers for autonomous robots.* Paper presented at the Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on.

Holz, T., Gorecki, C., Rieck, K., & Freiling, F. C. (2008). *Measuring and Detecting Fast-Flux Service Networks.* Paper presented at the NDSS.

Horng-Tzer, W., Ching-Hao, M., Kuo-Ping, W., & Hahn-Ming, L. (2012, 16-20 July 2012). *Real-Time Fast-Flux Identification via Localized Spatial Geolocation Detection.* Paper presented at the Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual.

Hsu, C.-H., Huang, C.-Y., & Chen, K.-T. (2010). Fast-Flux Bot Detection in Real Time. In S. Jha, R. Sommer, & C. Kreibich (Eds.), *Recent Advances in Intrusion Detection* (Vol. 6307, pp. 464-483): Springer Berlin Heidelberg.

Hsu, F.-H., Wang, C.-S., Hsu, C.-H., Tso, C.-K., Chen, L.-H., & Lin, S.-H. (2014). Detect fast-flux domains through response time differences. *Selected Areas in Communications, IEEE Journal on, 32*(10), 1947-1956.

Huang, S.-Y., Mao, C.-H., & Lee, H.-M. (2010). *Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection*. Paper presented at the Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China.

ICANN Security and Stability Advisory Committee (SSAC). (March 2008). SSAC Advisory on Fast Flux Hosting and DNS, Fast and Double Flux Attacks.

Inc, T. E.-T. M. (Online). New Zeus Gameover Employs DGA and Fast Flux Techniques.

Jeong, Y.-S., Kang, I.-H., Jeong, M.-K., & Kong, D. (2012). A new feature selection method for one-class classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(6), 1500-1509.

Jiang, N., Cao, J., Jin, Y., Li, L. E., & Zhang, Z.-L. (2010). *Identifying suspicious activities through dns failure graph analysis.* Paper presented at the Network Protocols (ICNP), 2010 18th IEEE International Conference on.

Kalige, E., Burkey, D., & Director, I. (2012). A case study of eurograbber: How 36 million euros was stolen via malware. *Versafe (White paper)*.

Karasaridis, A., Rexroad, B., & Hoeflin, D. (2007). *Wide-scale botnet detection and characterization.* Paper presented at the Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets.

Kasabov, N., Dhoble, K., Nuntalid, N., & Indiveri, G. (2013). Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks, 41*, 188-201.

Kira, K., & Rendell, L. A. (1992). *The feature selection problem: Traditional methods and a new algorithm.* Paper presented at the Aaai.

Koo, T.-M., Chang, H.-C., & Chuang, C.-C. (2012). Detecting and Analyzing Fast-Flux Service Networks. *Advances in Information Sciences & Service Sciences, 4*(10).

Kwon, J., Lee, J., Lee, H., & Perrig, A. (2016). PsyBoG: A scalable botnet detection method for large-scale DNS traffic. *Computer Networks, 97*, 48-73. doi:http://dx.doi.org/10.1016/j.comnet.2015.12.008

Lee, W., & Stolfo, S. J. (2000). A framework for constructing features and models for intrusion detection systems. *ACM transactions on Information and system security (TiSSEC), 3*(4), 227-261.

Levy, E., & Arce, I. (2006). A Short Visit to the Bot Zoo. *IEEE Security & Privacy, vol*, 76-79.

Lin, H.-T., Lin, Y.-Y., & Chiang, J.-W. (2013). Genetic-based real-time fast-flux service networks detection. *Computer Networks, 57*(2), 501-513. doi:http://dx.doi.org/10.1016/j.comnet.2012.07.017

Lin, Z., Ma, D., Meng, J., & Chen, L. (2018). Relative ordering learning in spiking neural network for pattern recognition. *Neurocomputing, 275*, 94-106.

Mandal, S. K. (2017). Performance Analysis Of Data Mining Algorithms For Breast Cancer Cell Detection Using Naïve Bayes, Logistic Regression and Decision Tree. *International Journal Of Engineering And Computer Science, 6*(2).

Martinez-Bea, S., Castillo-Perez, S., & Garcia-Alfaro, J. (2013). *Real-time malicious fast-flux detection using DNS and bot related features.* Paper presented at the PST.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure, 405*(2), 442-451. doi:http://dx.doi.org/10.1016/0005-2795(75)90109-9

Nuntalid, N., Dhoble, K., & Kasabov, N. (2011). *EEG classification with BSA spike encoding algorithm and evolving probabilistic spiking neural network.* Paper presented at the Neural information processing.

Otgonbold, T. (2014). ADAPT: An anonymous, distributed, and active probing-based technique for detecting malicious fast-flux domains.

Pa, Y. M. P., Yoshioka, K., & Matsumoto, T. (2015). Detecting malicious domains and authoritative name servers based on their distinct mappings to IP addresses. *Journal of information processing, 23*(5), 623-632.

Pappas, V., Wessels, D., Massey, D., Lu, S., Terzis, A., & Zhang, L. (2009). Impact of configuration errors on DNS robustness. *Selected Areas in Communications, IEEE Journal on, 27*(3), 275-290.

Passerini, E., Paleari, R., Martignoni, L., & Bruschi, D. (2008). Fluxor: Detecting and monitoring fast-flux service networks *Detection of intrusions and malware, and vulnerability assessment* (pp. 186-206): Springer.

Paul, T., Tyagi, R., Manoj, B., & Thanudas, B. (2014). *Fast-flux botnet detection from network traffic.* Paper presented at the India Conference (INDICON), 2014 Annual IEEE.

Perdisci, R., Corona, I., Dagon, D., & Lee, W. (2009). *Detecting malicious flux service networks through passive analysis of recursive dns traces.* Paper presented at the Computer Security Applications Conference, 2009. ACSAC'09. Annual.

Perdisci, R., Corona, I., Dagon, D., & Wenke, L. (2009, 7-11 Dec. 2009). *Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces.* Paper presented at the Computer Security Applications Conference, 2009. ACSAC '09. Annual.

Perdisci, R., Corona, I., & Giacinto, G. (2012). Early detection of malicious flux networks via large-scale passive DNS traffic analysis. *Dependable and Secure Computing, IEEE Transactions on, 9*(5), 714-726.

Pharmacy, F. c. C. (Online).

Qassrawi, M. T., & Zhang, H. L. (2012). *Detecting Malicious Fast Flux Domains.* Paper presented at the Applied Mechanics and Materials.

Rajab, M. A., Zarfoss, J., Monrose, F., & Terzis, A. (2006). *A multifaceted approach to understanding the botnet phenomenon*. Paper presented at the Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, Rio de Janeriro, Brazil.

Scharrenberg, P. (2008). Analyzing Fast-Flux Service Networks. *Diploma Dissertation, RWTH Aachen University, Aachen, North Rhine Westphalia, Germany*.

Schliebs, S., & Kasabov, N. (2013). Evolving spiking neural network—a survey. *Evolving Systems, 4*(2), 87-98.

Semantec. (2018). Internet security threat report

Shaikh, A., Tewari, R., & Agrawal, M. (2001). *On the effectiveness of DNS-based server selection.* Paper presented at the INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE.

Sheng, Y., Shijie, Z., & Sha, W. (2010, 25-27 June 2010). *Fast-flux attack network identification based on agent lifespan.* Paper presented at the Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on.

Soltic, S., & Kasabov, N. (2010). Knowledge extraction from evolving spiking neural networks with rank order population coding. *International Journal of Neural Systems, 20*(06), 437-445.

Spamwiki., C. P.-. (Online). Retrieved from http://spamtrackers.eu/wiki/index.php/Canadian_Pharmacy

Stalmans, E., Hunter, S. O., & Irwin, B. (2012, 15-17 Aug. 2012). *Geo-spatial autocorrelation as a metric for the detection of Fast-Flux botnet domains.* Paper presented at the Information Security for South Africa (ISSA), 2012.

Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., . . . Vigna, G. (2009). *Your botnet is my botnet: analysis of a botnet takeover.* Paper presented at the Proceedings of the 16th ACM conference on Computer and communications security.

Stornig, F. (2013). Detection of Botnet Fast-Flux Domains by the aid of spatial analysis methods.

Su, M.-Y., & Tsai, C.-H. (2012). A prevention system for spam over Internet telephony. *Appl. Math, 6*(2S), 579S-585S.

Swets, J. A. (2014). *Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers*: Psychology Press.

Villamarín-Salomón, R., & Brustoloni, J. C. (2009). *Bayesian bot detection based on DNS traffic similarity.* Paper presented at the Proceedings of the 2009 ACM symposium on Applied Computing.

Vu Hong, L. (2012). DNS Traffic Analysis for Network-based Malware Detection.

Xu, W., Wang, X., & Xie, H. (2013). New Trends in FastFlux Networks. *media. blackhat. com*.

Yu, B., Smith, L., & Threefoot, M. (2014). Semi-supervised Time Series Modeling for Real-Time Flux Domain Detection on Passive DNS Traffic. In P. Perner (Ed.), *Machine Learning and Data Mining in Pattern Recognition* (Vol. 8556, pp. 258-271): Springer International Publishing.

Yu, S. (2014). Malicious Networks for DDoS Attacks *Distributed Denial of Service Attack and Defense* (pp. 15-29): Springer.

Yu, X., Zhang, B., Kang, L., & Chen, J. (2012). Fast-Flux Botnet Detection Based on Weighted SVM. *Information Technology Journal, 11*(8), 1048.

Zhao, D., & Traore, I. (2012). *P2P botnet detection through malicious fast flux network identification.* Paper presented at the P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on.

Zhao, Y., & Jin, Z. (2015). Quickly Identifying FFSN Domain and CDN Domain with Little Dataset.

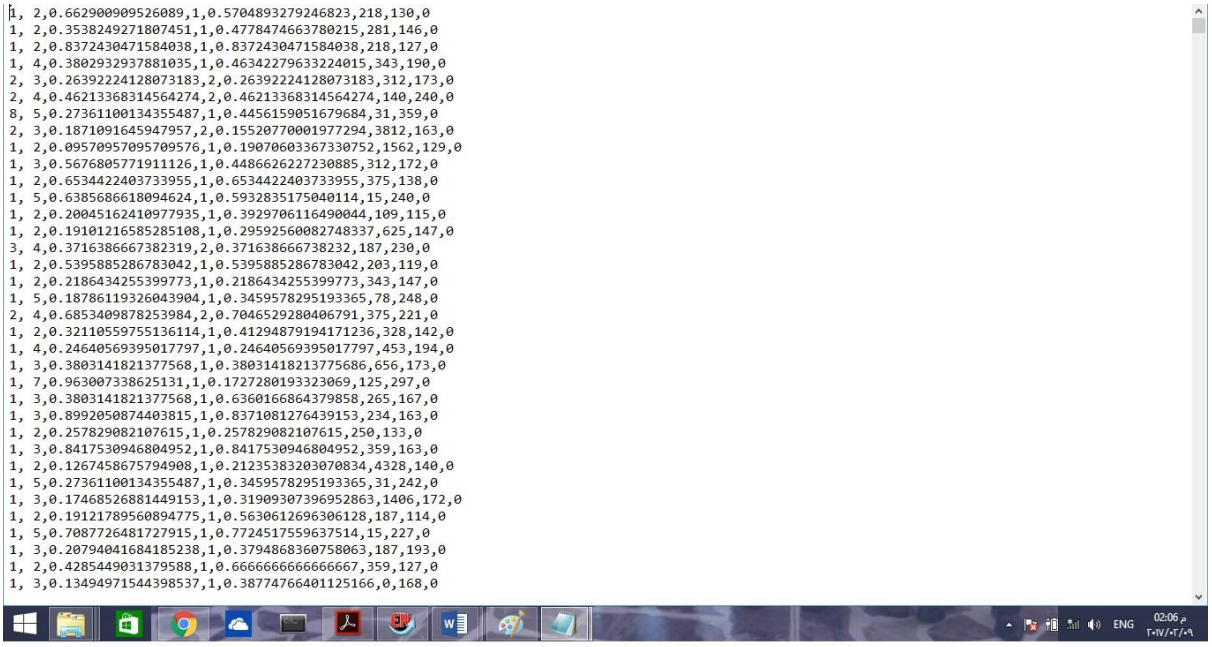
Zheng, B., Yoon, S. W., & Lam, S. S. (2014). Breast cancer diagnosis based on feature extraction using a hybrid of K-means and support vector machine algorithms. *Expert Systems with Applications, 41*(4), 1476-1482.

Zhou, C. V., Leckie, C., & Karunasekera, S. (2009). Collaborative detection of fast flux phishing domains. *Journal of Networks, 4*(1), 75-84.

APPENDICES

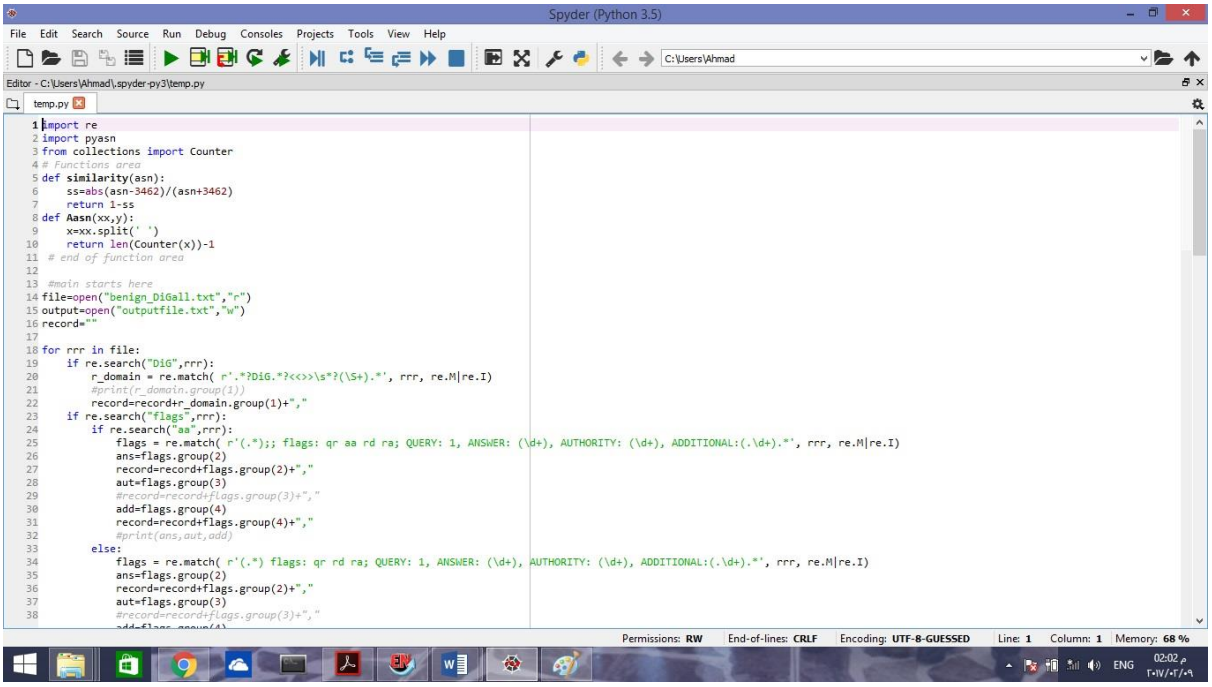
Dataset and code samples

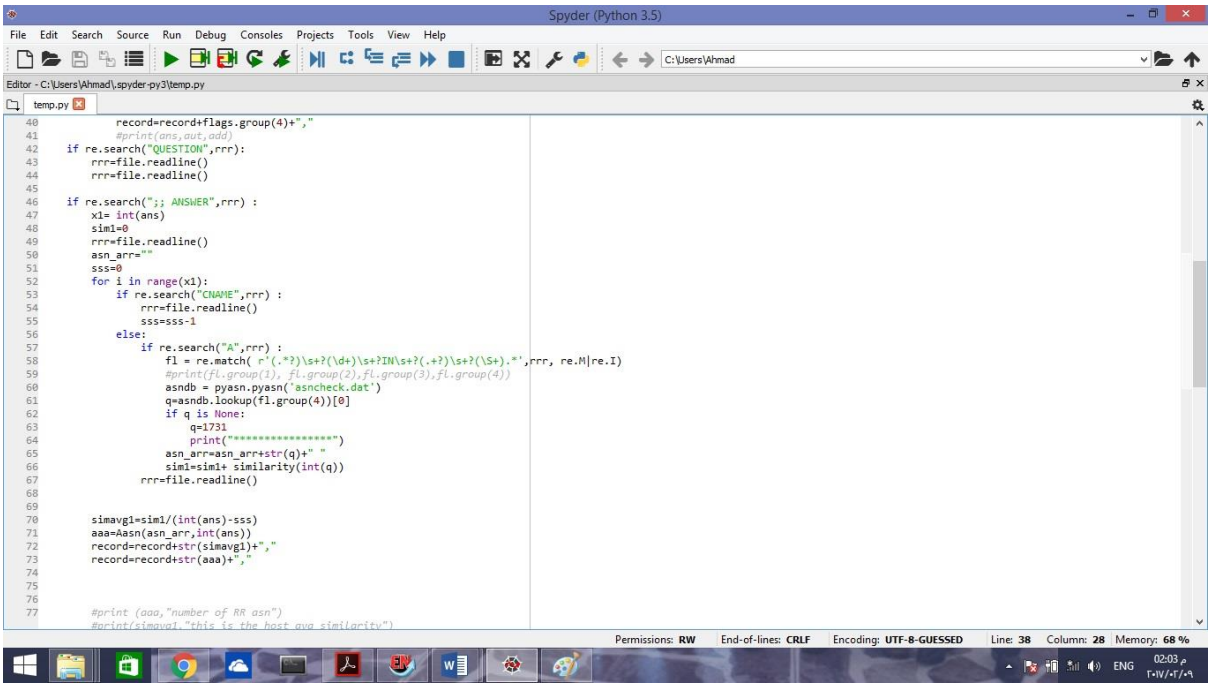
Sample of the feature file:

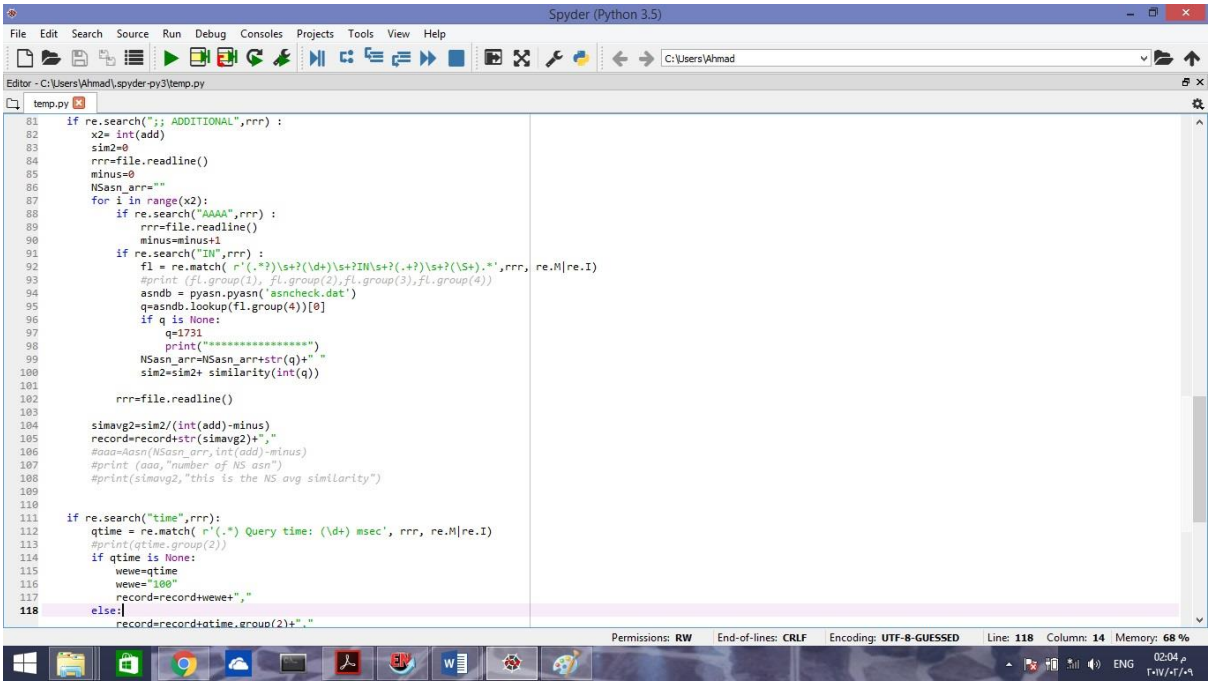


```
1, 2,0.662900909526089,1,0.5704893279246823,218,130,0
1, 2,0.3538249271807451,1,0.4778474663780215,281,146,0
1, 2,0.8372430471584038,1,0.8372430471584038,218,127,0
1, 4,0.3802932937881035,1,0.46342279633224015,343,190,0
2, 3,0.26392224128073183,2,0.26392224128073183,312,173,0
2, 4,0.46213368314564274,2,0.46213368314564274,140,240,0
8, 5,0.27361100134355487,1,0.4456159051679684,31,359,0
2, 3,0.1871091645947957,2,0.15520770001977294,3812,163,0
1, 2,0.09570957095709576,1,0.19070603367330752,1562,129,0
1, 3,0.5676805771911126,1,0.4486626227230885,312,172,0
1, 2,0.6534422403733955,1,0.6534422403733955,375,138,0
1, 5,0.6385686618094624,1,0.5932835175040114,15,240,0
1, 2,0.20045162410977935,1,0.3929706116490044,109,115,0
1, 2,0.19101216585285108,1,0.29592560082748337,625,147,0
3, 4,0.3716386667382319,2,0.3716386667382319,187,230,0
1, 2,0.5395885286783042,1,0.5395885286783042,203,119,0
1, 2,0.2186434255399773,1,0.2186434255399773,343,147,0
1, 5,0.18786119326043904,1,0.3459578295193365,78,248,0
2, 4,0.6853409878253984,2,0.7046529280406791,375,221,0
1, 2,0.32110559755136114,1,0.41294879194171236,328,142,0
1, 4,0.24640569395017797,1,0.24640569395017797,453,194,0
1, 3,0.3803141821377568,1,0.38031418213775686,656,173,0
1, 7,0.963007338625131,1,0.1727280193323069,125,297,0
1, 3,0.3803141821377568,1,0.6360166864379858,265,167,0
1, 3,0.8992050874403815,1,0.8371081276439153,234,163,0
1, 2,0.257829082107615,1,0.257829082107615,250,133,0
1, 3,0.8417530946804952,1,0.8417530946804952,359,163,0
1, 2,0.1267458675794908,1,0.21235383203070834,4328,140,0
1, 5,0.27361100134355487,1,0.3459578295193365,31,242,0
1, 3,0.17468526881449153,1,0.31909307396952863,1406,172,0
1, 2,0.19121789560894775,1,0.5630612696306128,187,114,0
1, 5,0.7087726481727915,1,0.7724517559637514,15,227,0
1, 3,0.20794041684185238,1,0.3794868360758063,187,193,0
1, 2,0.4285449031379588,1,0.6666666666666667,359,127,0
1, 3,0.13494971544398537,1,0.387747664011251 66,0,168,0
```

Code in python for extract feature from the dataset:

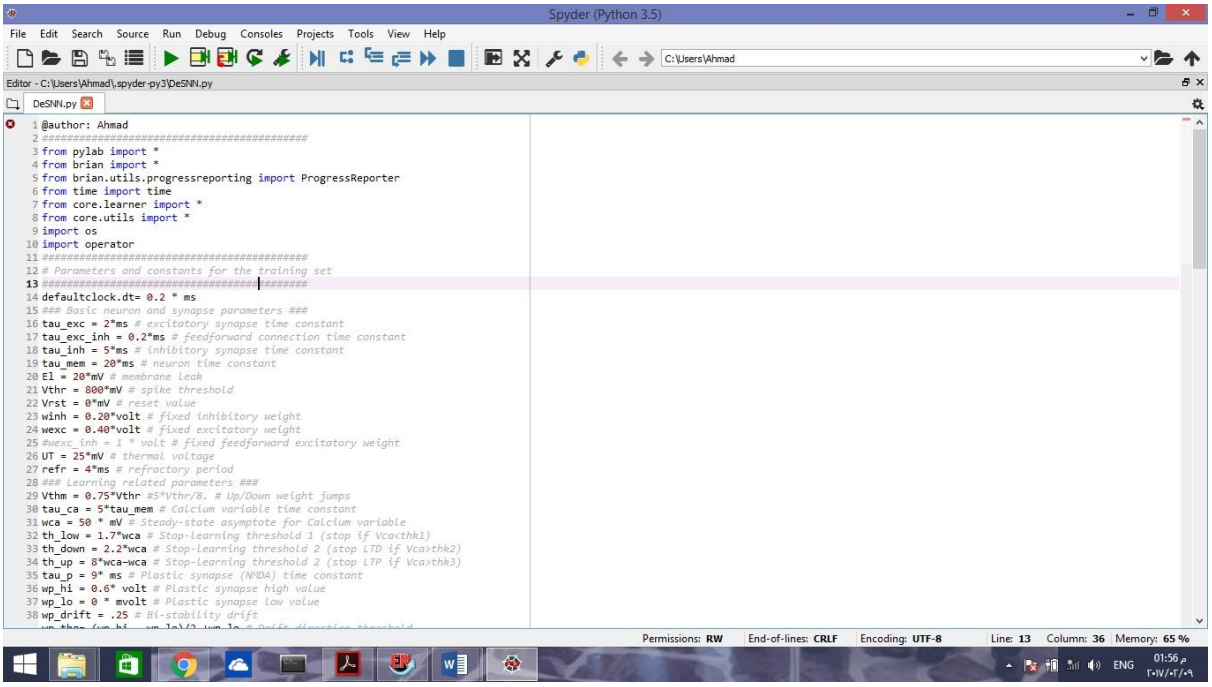


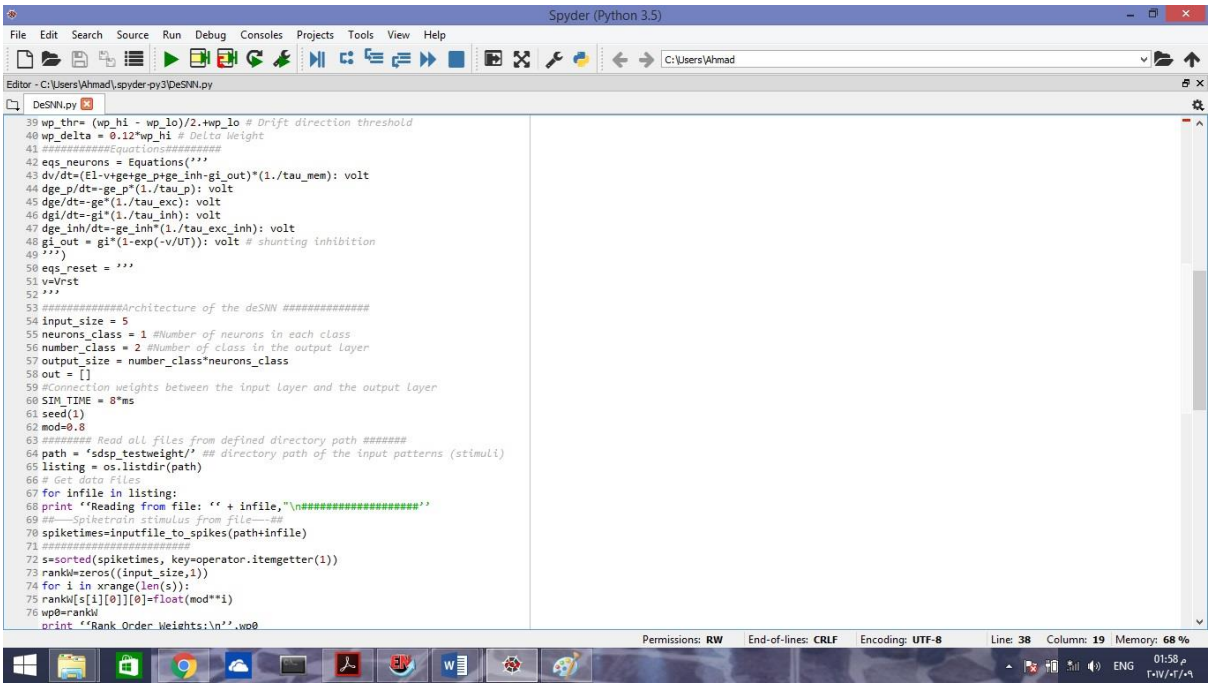


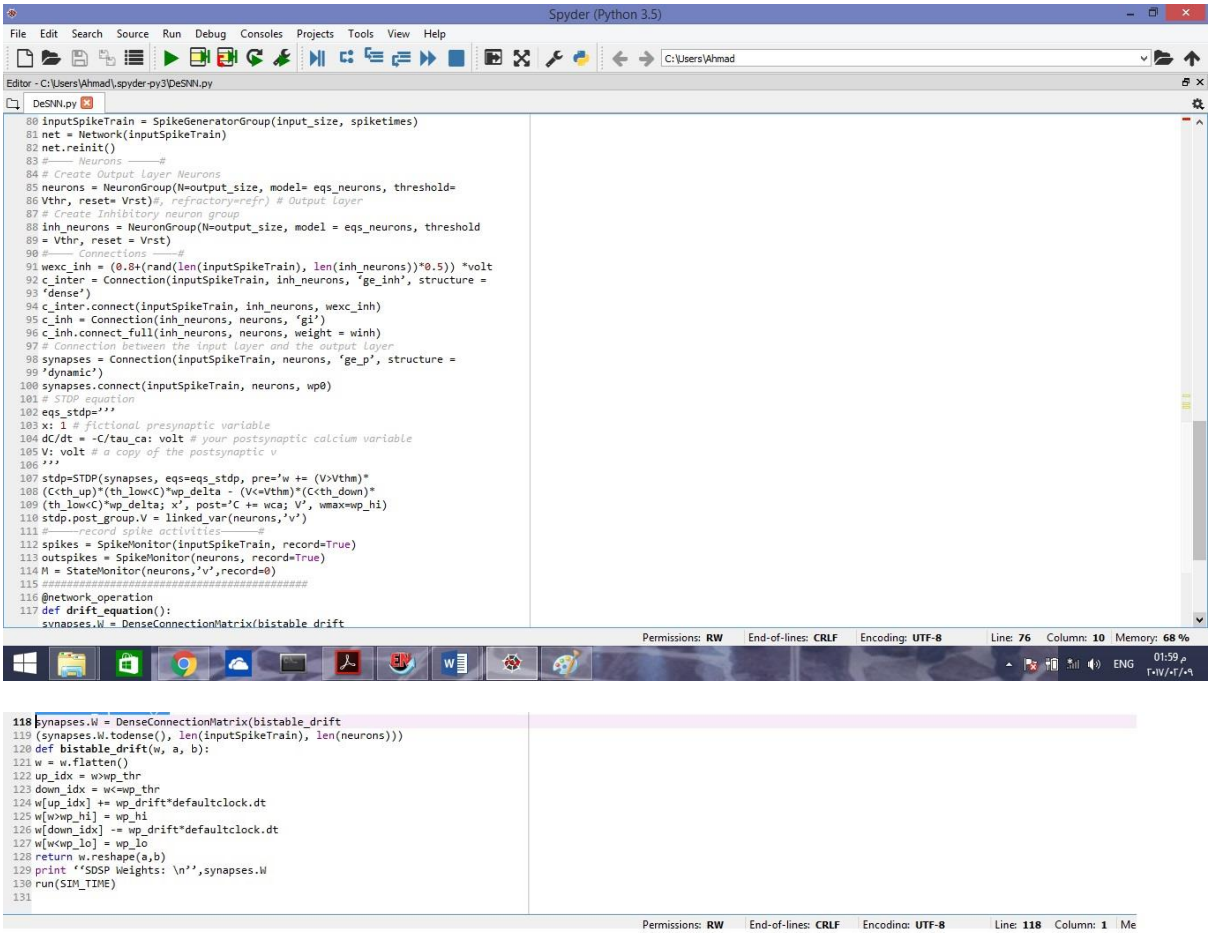



```
    if re.search(";; ADDITIONAL",rrr) :
        x2= int(add)
        sim2=0
        rrr=file.readline()
        minus=0
        NSasn_arr=""
        for i in range(x2):
            if re.search("AAAA",rrr) :
                rrr=file.readline()
                minus=minus+1
            if re.search("IN",rrr) :
                fl = re.match( r'(.*?)\s+?(\d+)\s+?IN\s+?(.+?)\s+?(\S+).*',rrr, re.M|re.I)
                #print (fl.group(1), fl.group(2),fl.group(3),fl.group(4))
                asndb = pyasn.pyasn('asncheck.dat')
                q=asndb.lookup(fl.group(4))[0]
                if q is None:
                    q=1731
                    print("***************")
                NSasn_arr=NSasn_arr+str(q)+" "
                sim2=sim2+ similarity(int(q))

            rrr=file.readline()

        simavg2=sim2/(int(add)-minus)
        record=record+str(simavg2)+","
        #aaa=Aasn(NSasn_arr,int(add)-minus)
        #print (aaa,"number of NS asn")
        #print(simavg2,"this is the NS avg similarity")


    if re.search("time",rrr):
        qtime = re.match( r'(.*) Query time: (\d+) msec', rrr, re.M|re.I)
        #print(qtime.group(2))
        if qtime is None:
            wewe=qtime
            wewe="100"
            record=record+wewe+","
        else:
            record=record+qtime.group(2)+","
    if re.search("SIZE",rrr):
        msg = re.match( r'(.*) MSG SIZE  rcvd: (\d+)', rrr, re.M|re.I)
        if msg is None:
            wewe=msg
            wewe="100"
            record=record+wewe+","
        else:
            record=record+msg.group(2)+"\n"
        output.writelines(record)
        record=""

        #print(msg.group(2))
        #record=r_domain+ the other variables values

file.close()
output.close()
# End of main
```

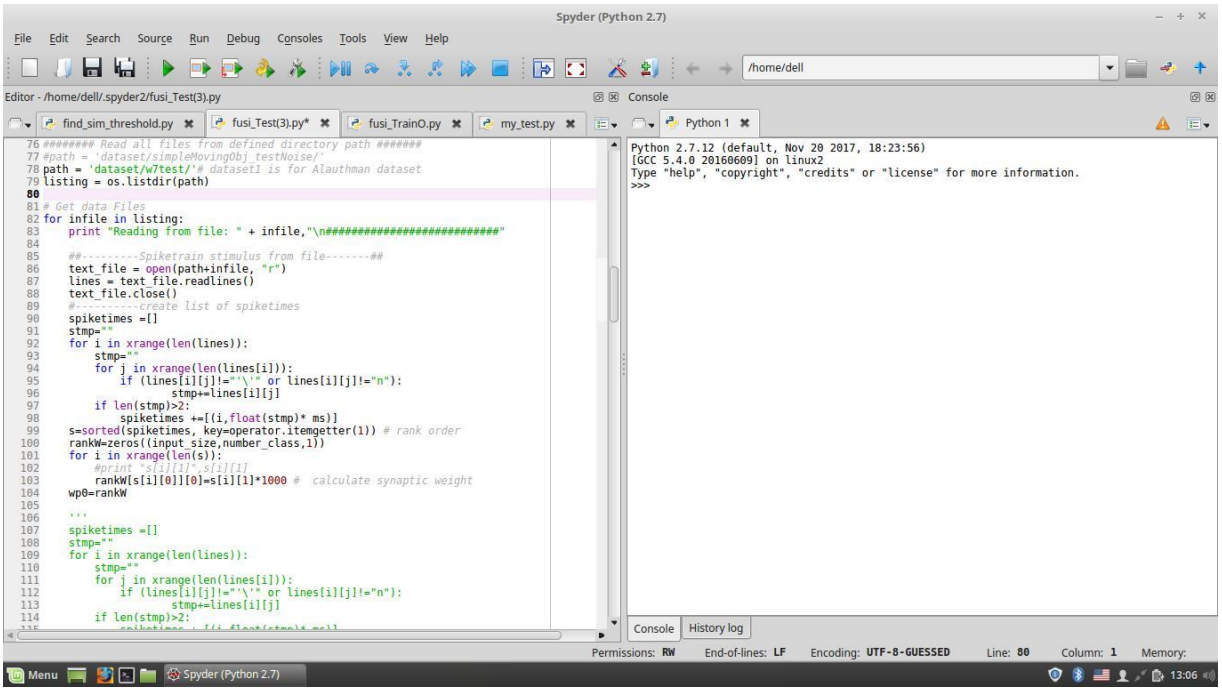
Classifier code in python (DeSNN):







Initial weight code based on Spiketime:


Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Tools View Help
/home/dell
Editor - /home/dell/.spyder2/fusi_Test(3).py
Console
find_sim_threshold.py
fusi_Test(3).py*
fusi_TrainO.py
my_test.py
Python 1
######## Read all files from defined directory path #######
#path = 'dataset/simpleMovingObj_testNoise/'
path = 'dataset/w7test/'# dataset1 is for Alauthman dataset
listing = os.listdir(path)
# Get data Files
for infile in listing:
print "Reading from file: " + infile,"\n############################"
##---------Spiketrain stimulus from file-------##
text_file = open(path+infile, "r")
lines = text_file.readlines()
text_file.close()
#----------create list of spiketimes
spiketimes =[]
stmp=""
for i in xrange(len(lines)):
stmp=""
for j in xrange(len(lines[i])):
if (lines[i][j]!="\'" or lines[i][j]!="n"):
stmp+=lines[i][j]
if len(stmp)>2:
spiketimes +=[(i,float(stmp)* ms)]
s=sorted(spiketimes, key=operator.itemgetter(1)) # rank order
rankW=zeros((input_size,number_class,1))
for i in xrange(len(s)):
#print "s[i][1] ",s[i][1]
rankW[s[i][0]][0]=s[i][1]*1000 # calculate synaptic weight
wp0=rankW
'''
spiketimes =[]
stmp=""
for i in xrange(len(lines)):
stmp=""
for j in xrange(len(lines[i])):
if (lines[i][j]!="\'" or lines[i][j]!="n"):
stmp+=lines[i][j]
if len(stmp)>2:
Python 2.7.12 (default, Nov 20 2017, 18:23:56)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
Console
History log
Permissions: RW
End-of-lines: LF
Encoding: UTF-8-GUESSED
Line: 80
Column: 1
Memory:
Menu
Spyder (Python 2.7)
13:06