Anomaly based Botnet Detection using DNS Traffic Analysis

A thesis submitted

in fulfillment of the requirements for the award of degree of

Doctor of Philosophy

by

Manmeet Singh

(951503006)

under the guidance of

Dr. Maninder Singh
Professor

Dr. Sanmeet Kaur
Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

PATIALA – 147004 (INDIA)

August 2019

Certificate

I hereby certify that the work which is being presented in this thesis titled, "**Anomaly based Botnet Detection using DNS Traffic Analysis**", for the award of degree of ***Doctor of Philosophy*** submitted in **Computer Science and Engineering Department** of Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Maninder Singh, Dr. Sanmeet Kaur and refers other researcher's work which is duly listed in the reference section.

The matter presented in the thesis has not been submitted for the award of any other degree of this or any other University.

(**Manmeet Singh**)
951503006

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Dr. Maninder Singh)
Professor
Computer Science & Engineering Department
Thapar Institute of Engineering and Technology
Patiala – 147004

(Dr. Sanmeet Kaur)
Associate Professor
Computer Science & Engineering Department
Thapar Institute of Engineering and Technology
Patiala – 147004

Acknowledgments

I am grateful to the Almighty God for all His blessings without which nothing fruitful can be envisioned or attained.

I would like to express my sincere thanks and indebtedness to Dr. Maninder Singh for his continuous support during my doctorate's research journey. His uncompromising professional ethics has been the guiding force in following the right path. His in-depth technical expertise is a source of inspiration to achieve my goal. His unstinted support has always helped me to overcome large obstacles during my research work.

I would like to express my honest thanks and gratitude to Dr. Sanmeet Kaur for her positive attitude and guidance during my doctorate's research journey. Her ability to motivate and communicate helped me in moving in the right direction. Her ability to foresee potential problems and possible solutions aided me to a greater extent. Her positive attitude as a mentor is beyond belief.

I wish to express my gratitude to Dr. Neeraj Kumar, Dr. Parteek Bhatia and Dr. Kulbir Singh for being my Doctoral committee members and advising me at all the stages of my research studies. I would also like to thank Dr. Prashant Singh Rana for his valuable inputs.

I am also thankful to Mr. Harcharan Jeet Singh for his valuable support in the traffic capture stage. I am also thankful to Dr. Sukhpal Singh for guiding me in the early stages of my research studies. I am also thankful to Mr. Gurpal Singh for his valuable contribution. I am also thankful to Mr. Abhishek Khanna for his help and support.

I would also like to thank Dr. Rafat Siddique, Dean of Research and Sponsored Projects for academic support. My sincere thanks to Dr. Parkash Gopalan, Director, Thapar Institute of Engineering and Technology for all the facilities that have been instrumental in the creation of a healthy research environment in the Institute.

Finally, I acknowledge the people who mean a lot to me, my beloved parents, lovely wife, little daughter, brother and sisters for being the continuous source of motivation and unbelievable support.

Manmeet Singh

Abstract

Cybercrimes are evolving on a regular basis and these crimes are becoming a greater threat day by day. Earlier these threats were very general and unorganized. In the last decade, these attacks have become highly sophisticated in nature. This higher level of coordination is possible mainly due to Botnet which is a cluster of infected hosts controlled remotely by an attacker (Botmaster). The number of infected machines is continuously rising thereby resulting in Botnets with many of these having even over a million infected machines. This innumerous set of machines with varied computational and storage capabilities give the botmaster a lethal weapon to launch various security attacks. This never-ending menace of the botnet is causing many serious problems on the Internet.

Domain Name System is a large-scale distributed database on the Internet, which is being abused as a Botnet communication channel. Significant efforts have been made in detecting botnet at the global level which relies heavily on finding failed queries and domain flux information for botnet detection, there are very few efforts being made to detect bot infection at an enterprise level. Detecting bot-infected machines are vital for any organization in combating various security threats.

This research work proposes a novel anomaly-based detection technique which considers captured DNS traffic from LAN hosts on hourly basis to generate DNS fingerprint and attempts to find anomalous behavior which is quite different from normal machine behavior. This research work successfully demonstrates the DNS Anomaly Detection (named BotDAD) technique for detecting bot-infected machine in a network using DNS fingerprinting. It uses a feature extractor module to extract DNS attributes and build a host profile for all hosts in the network. The host profile is then parsed to generate DNS fingerprint. BotDAD creates DNS fingerprint of each host in the network and uses anomaly detection engine to label them as bot or clean. BotDAD uses a machine learning classifier to develop a trained model for future predictions. The system is evaluated against DNS network traffic captured from TIET Patiala

campus on an hourly basis. The system was able to detect Bot infected machines in the network. The domains used for C&C by these bots were validated against online DGA domains database.

Results from BotDAD gives an accuracy of 0.9978. To improve the accuracy of the BotDAD, a multi-layer neural network named DeepDAD was implemented. DeepDAD is a Deep Learning based DNS Anomaly Detection tool created as part of this research which considers multipoint anomaly detection and uses deep learning algorithms. Instead of relying on a single point anomaly for labeling DNS fingerprint as malicious, an improved labeling technique which uses multipoint anomaly detection is implemented. Two machine learning frameworks namely Scikit-learn and TensorFlow were used to train and test the model showing significant improvement over the results obtained using BotDAD. Finally, a graphical user interface for easy testing and comparison is presented.

Table of Contents

List of Figures

List of Tables

Abbreviations

Notation	Description
AAAA	IP Version 6 Address records
ACM	Association for Computing Machinery
AHBL	Abusive Hosts Blocking List
AM	Ante Meridiem
ASN	Autonomous System Number
AUC	Area Under the Curve
BRBL	Barracuda Reputation Block List
C&C	Command and Control
CDN	Content Delivery Network
CD-ROM	Compact Disc, Read-Only Memory
CNAME	Canonical Name Record
DDoS	Distributed Denial-of-Service
DGA	Domain Generation Algorithms
DHCP	Dynamic Host Configuration Protocol
DNSBL	DNS-based Block List

FN	False Negative
FP	False Positive
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GB	GigaByte
HTTP	HyperText Transfer Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IRC	Internet Relay Chat
ISP	Internet Service Provider
MB	MegaByte
MLP	Multi-Layer Perceptron
MX	Mail Exchange
NAT	Network Address Translation
NXDOMAIN	Non-eXistent Domain
P2P	Peer-to-Peer
PM	Post Meridiem
PTR	Pointer Record (DNS)

QR	Query/Response (DNS)
RR	Response Records (DNS)
SLD	Second-Level Domain
SMOTE	Synthetic Minority Over-sampling Technique
SSH	Secure Shell
SVM	Support Vector Machine
T&F	Taylor & Francis
TCP	Transmission Control Protocol
TLD	Top-Level Domain
TN	True Negative
TP	True Positive
TTL	Time To Live
USB	Universal Serial Bus
VPN	Virtual Private Network

1. Introduction

The Cambridge Dictionary defines Bot [1] as “a computer program that works automatically especially one that searches for and finds information on the Internet”. It also defines Botnet [2] “as a group of computers that are by controlled by software containing harmful programs, without their user’s knowledge”. A botnet is a cluster of geographically diverse hosts controlled remotely via Command and Control server. Botnets are used for various wicked activities like Spamming [3]–[5], DDoS attacks [6]–[9], Click Fraud [10]–[12], Ad Fraud[13], Identity theft, etc. Vulnerable systems on the Internet are exploited using various techniques like zero-day attacks, email attachments, etc. Once exploited, it installs a backdoor that downloads the bot binary and connects to the C&C center. C&C Server is then used to launch various wicked activities. C&C Servers usually make use of various evasive techniques like DDNS, IP Flux, and Domain Flux to prevent a single point of failure. These evasive techniques are continuously evolving and presenting a greater threat each passing day. Botnet detection has become a highly focused research area in computer and network security. Figure 1.1 represents an overview of the botnet.

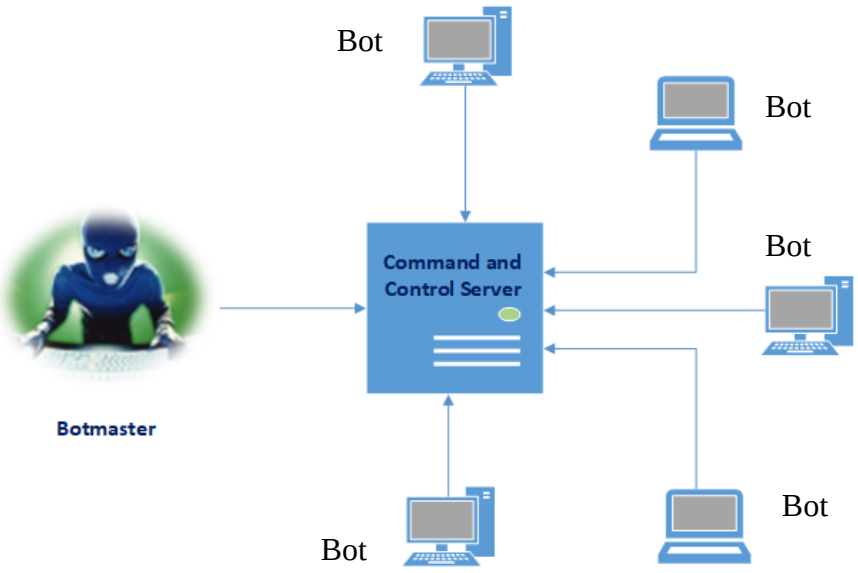


Figure 1.1 Botnet overview

Internet Relay Chat (IRC) protocol was initially used by Botnets which had a centralized architecture. It was an efficient mechanism as there was no intermediary. There was no delay in command execution and status reporting as there was direct communication. One of the inherent drawbacks of the model is a solo point of a letdown. Recent botnets are using Peer-to-Peer and HyperText Transfer Protocol (HTTP) and sometimes even both for communication. P2P Botnets offer a high level of security and robustness as there is no single point of failure. However, due to indirect communication, the command communication and reporting is slow and takes more time.

Botnet is the backbone for Cybercrimes [14]–[18] on the Internet. Economic losses due to these coordinated attacks are not only limited to advertisers and businesses but reach up to national level [19]. A botnet is used for various nefarious activities like Spam [3], DDoS [20], Click Fraud [11], Malvertising/Ad Fraud [21], Identity Theft and corporate espionage.

A botnet can be classified based on Botnet architecture which basically classifies a botnet based on the technique used to connect to the rendezvous point. Determining the right architecture is essential for Botnet detection as different architectures present a different level of resilience. Botnet architecture defines how the bots communicate with the rendezvous point. A detailed discussion on botnet architecture is presented in [22][23].

1.1. Brief History

Botnets have evolved over the years to evade the detection techniques. The evolution is based mainly only on the communication technique used to connect to the rendezvous point. A year-wise depiction of the various botnets along with their architecture/protocol and estimated botnet size is presented in [22]. EggDrop (1993) is considered the first popular botnet having a centralized architecture. It was developed by Robey Pointer using C programming language. It used IRC protocol for communication between Bot and rendezvous point.

Global Threat Bot (GTBot: 1998) used mIRC[24] along with a backdoor which hides the mIRC window and waits for a command from the botmaster. It spreads when the user is lured into installing a software utility. Upon installation, it runs silently in the background and can be used

to launch DDoS Attacks. Netbus (1998) was a software utility used for remote controlling a Windows system. It contained a wide range of commands for keylogging, screen capture, program invocation, reboot, eject CD-ROM, etc. It used a client-server architecture in which an attacker can remotely perform any task on the target system. Sinit botnet (2003) used a P2P protocol for communicating. Each infected machine is added to the P2P network where new Trojans are installed on the infected machines. Digital signatures were used to encrypt the Trojans in the P2P list.

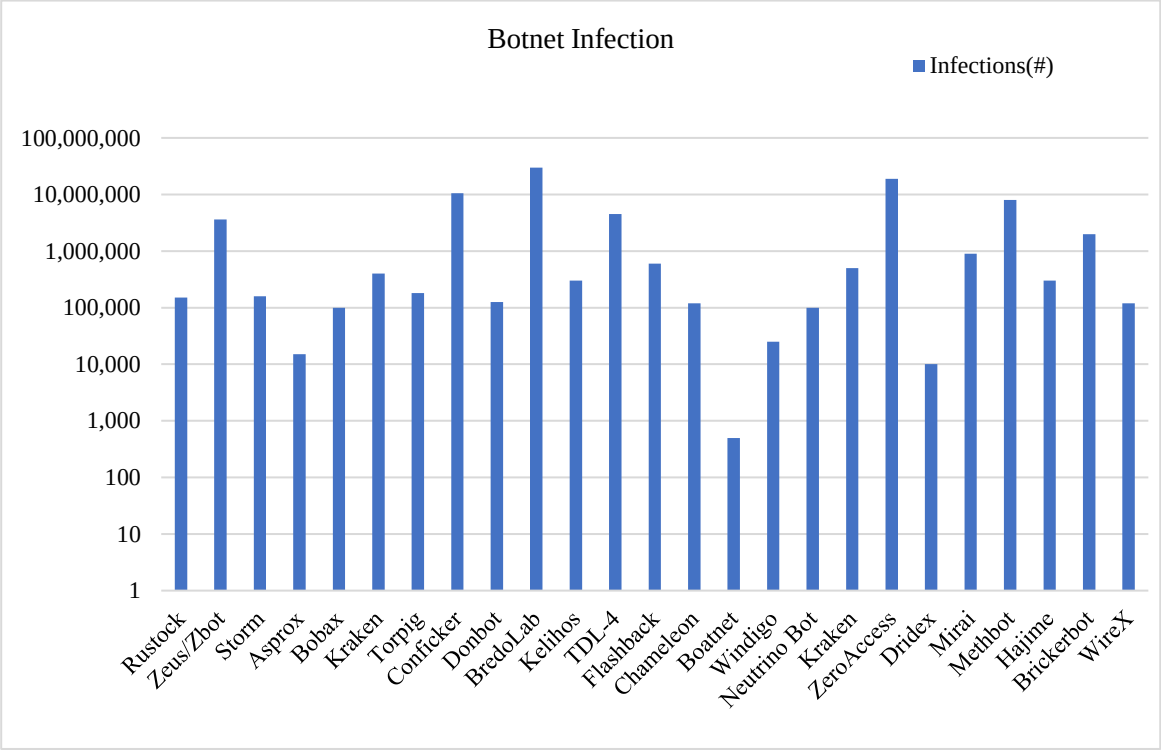


Figure 1.2 Number of infections of popular botnets

Gameover Zeus botnet (2007) [25] was a P2P botnet responsible for large financial frauds. It was dismantled in May 2014 by the law enforcement agencies from the US, UK, and Europe in Operation Tovar. U.S. government has offered an award of $3 Million [26] for the arrest of Russian hacker behind the Zeus Botnet[27][28]. New Botnets are continuously being discovered on a regular basis by the security research groups. Recently [20], [29], [30], attacks by Botnet employing Internet of Things (IoT) devices are being investigated by security agencies. Social Networking sites like Twitter are being used as a rendezvous point to launch

security attacks [31]–[39]. New botnets employ various stealth techniques to avoid detection. Figure 1.2 presents a bar chart of some of the popular botnets based on infection size.

1.2. Lifecycle

The lifecycle of botnet consists of various phases a bot undergoes from the initial infection stage to final maintenance stage [40]. An in-depth life cycle based botnet research is presented in [41]. Botmaster creates a rendezvous point where all the bots are programmed to connect on activation. Typically, this requires registering a Dynamic Domain Name Server (DDNS) and a static Internet Protocol (IP) address. This is the first step in the botnet lifecycle. After infrastructure setup, botmaster makes use of zero-day attacks to infect vulnerable systems on the Internet through different means like email attachments, malicious downloads, USB drives, etc. Once the basic infection is done, the system is then converted into bot by downloading bot binaries which contain an instruction to connect to the C&C server. In rallying phase, the bots connect to the C&C Server as set by botmaster and wait for instructions. Botmaster may use these bots to launch various attacks like DDoS, identity theft, etc. In order to avoid detection, botnet needs to continuously update its binaries to defeat various signature-based detection techniques. Maintenance and upgradation as such are an important part of the botnet lifecycle. Post upgradation, bots move back to the rallying stage and wait for a command from the C&C server. Figure 1.3 represents various stages in botnet lifecycle.

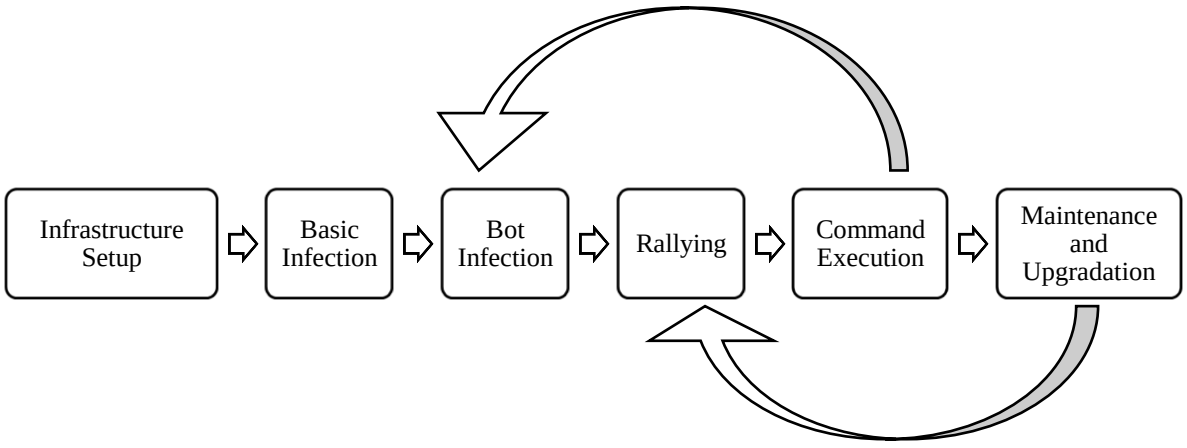


Figure 1.3 Life cycle of a botnet

1.3. Botnet Prevalence

A recent report published by McAfee Labs [42] presented a strong Botnet prevalence in the world as shown in Figure 1.4. While Wapomi botnet controls nearly half of the bot-infected machines, GoScanSSH controls 29% of the infected machines. China chopper webshell control nearly 10% of the infected machines as published in the report. Sality, Maazben, Ramnit Botnet, Sdbot & Muieblackcat have a share of 1% of the infected machines. Still, there is a significant number of infected machines 6% which are unknown. These are those machines which are infected with botnets which are still under investigation.

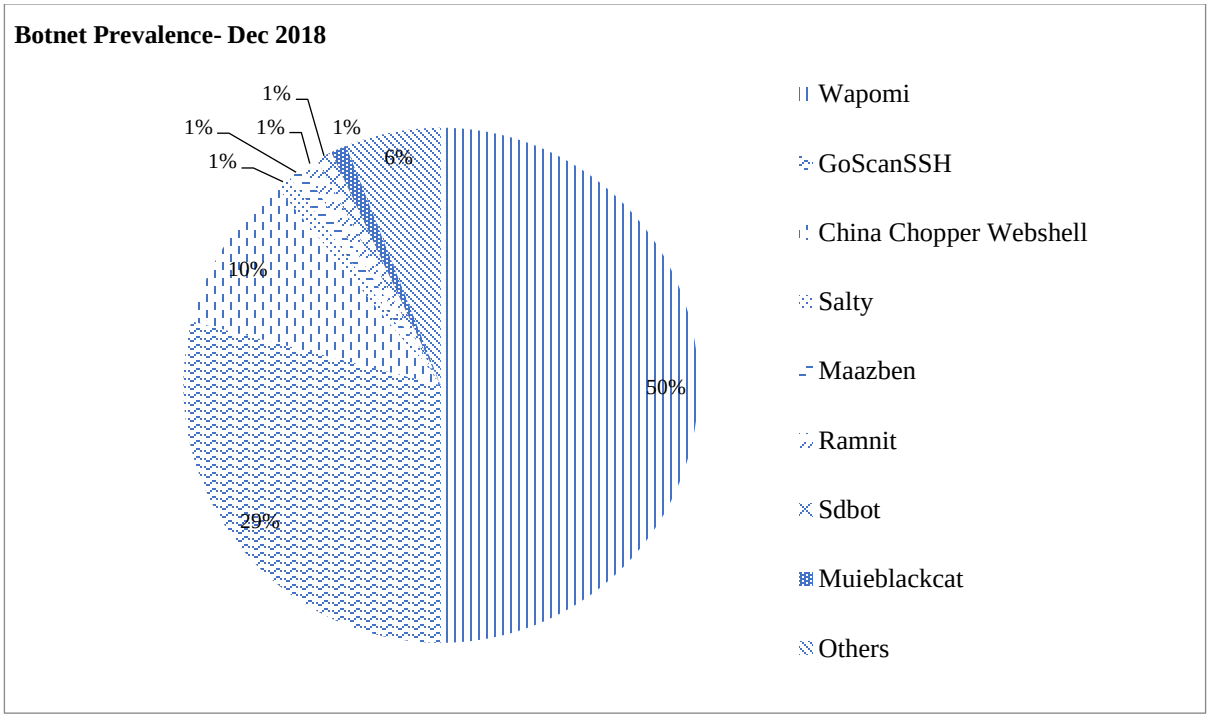


Figure 1.4 Worldwide botnet prevalence

Spamhaus project [43] maintains a country-wise list of a number of machines infected with the botnet. List of top 7 countries along with the number of infect machines is presented in Figure 1.5. India tops the list of infected machines, followed by China and Vietnam.

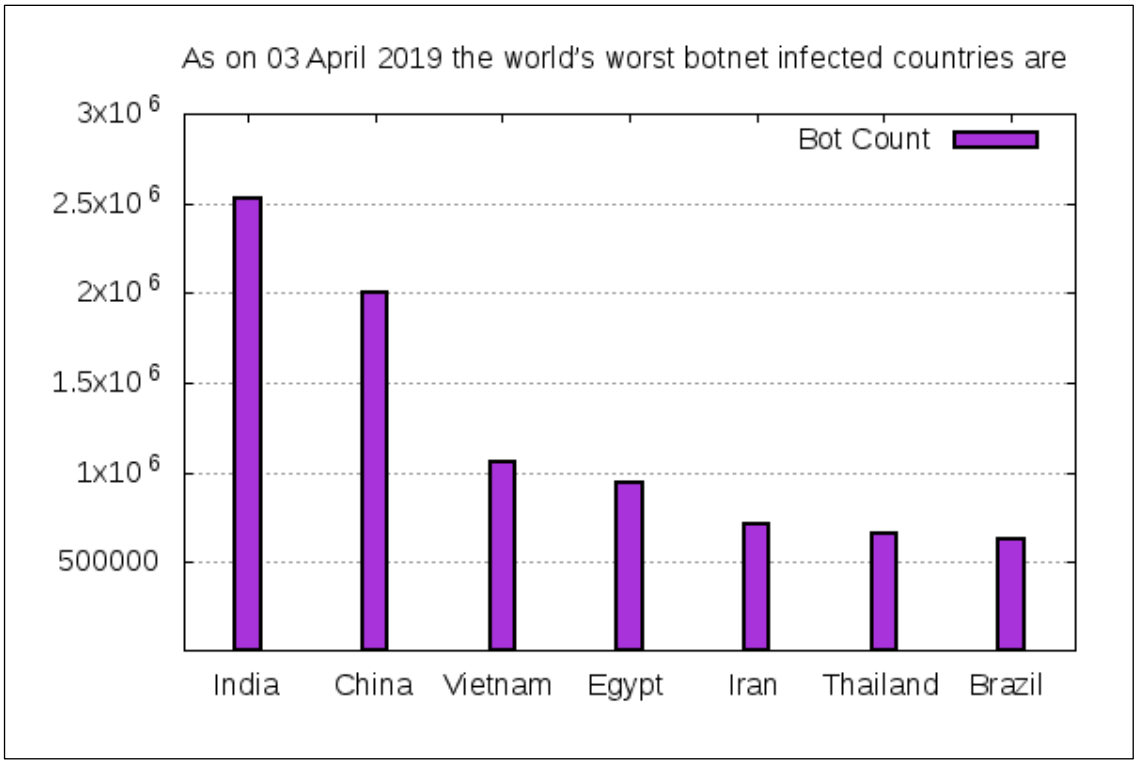


Figure 1.5 Top seven botnet infected countries

1.4. Botnet Architectures

Botnet architecture defines how the bots communicate with the C&C server [22][23]. It is classified into Centralized, Decentralized and Hybrid Architectures.

1.4.1. Centralized Command and Control Architecture

This architecture is the simplest one being used extensively in a client-server architecture wherein each client initiates a request to the server and wait for the response as shown in Figure 1.6. One of the advantages of this kind of architecture is the quick response time and easy coordination as all the clients are directly able to communicate with the server. Monitoring is very easy as the server can easily know the status of several active bots. Internet Relay Chat and HyperText Transport Protocol are the protocols extensively used for centralized architecture. IRC provides a fast and easy way for communication that requires a little setup. However, the IRC protocol has limited use and is mostly blocked in the corporate network.

HTTP thus becomes the prime means of communication in centralized architecture as it is the backbone of the Internet and cannot be blocked by the administrator. While this architecture presents the best scenario for bot monitoring and coordination, it lacks stealth and robustness capability as C&C Server disruption can cause a single form of failure.

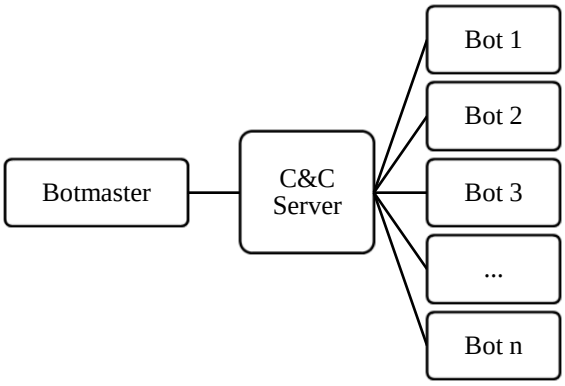


Figure 1.6 Centralized botnet architecture

1.4.2. Decentralized Command and Control Architecture

This architecture makes use of P2P framework as shown in Figure 1.7 in which greater flexibility and throughput is achieved as there is no central authority to manage or monitor the botnet. Moreover, it is difficult to take down due to the absence of a single point of failure. The command is injected into one of the peers by the botmaster which is then circulated to other peers in the network. Although P2P Botnets are very difficult to take down, bot monitoring and coordination is very slow since there is no direct communication between the botmaster and the bots.

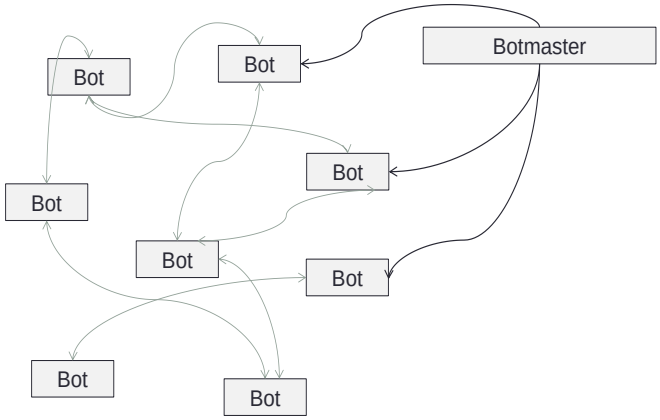


Figure 1.7 Decentralized architecture

Wang et al. [44] presented a design for an advanced P2P Botnet as shown in Figure 1.8. The proposed design requires no bootstrap routine. A bot peer list is maintained in each bot which is the single point of communication. This list is limited and reveals a little information about other peers. A report command is issued by the botmaster to monitor bots. All the bots reply to the sensor host which is controlled by the botmaster. Peer list can also be updated using the update command from the sensor host. The update allows the bot to evade many flow detection techniques.

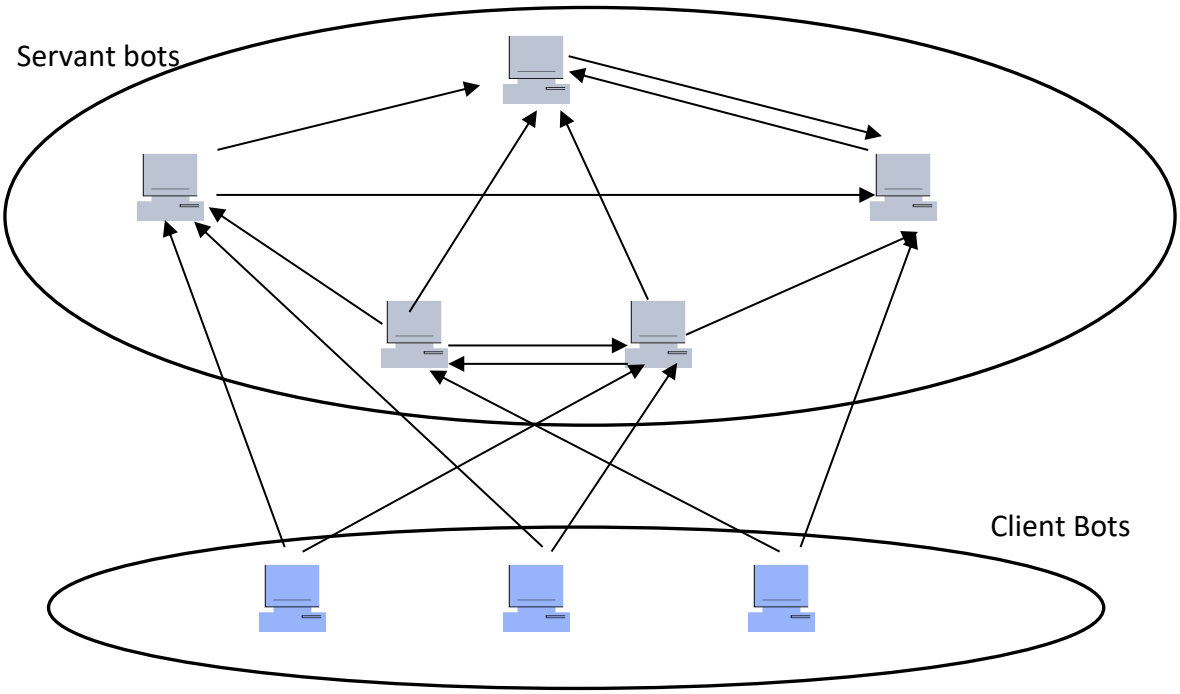


Figure 1.8 Architecture of P2P botnet

1.4.3. Hybrid Command and Control architecture

This architecture inherits the best features of both the centralized and P2P Architecture. Bots in this architecture are classified into servant bots and client bots. While servant bots are the ones which act as both client and the server having static Internet Protocol (Public IP) and routable addresses. Client bots are the ones which are placed behind the firewall or dynamically assigned

IP addresses and as such don't accept connections. All client bots connect after some period to servant bots and obtain further instructions. Bot monitoring and management are relatively easier as compared to P2P architecture. Moreover, there is no single point of failure as in Centralized architecture.

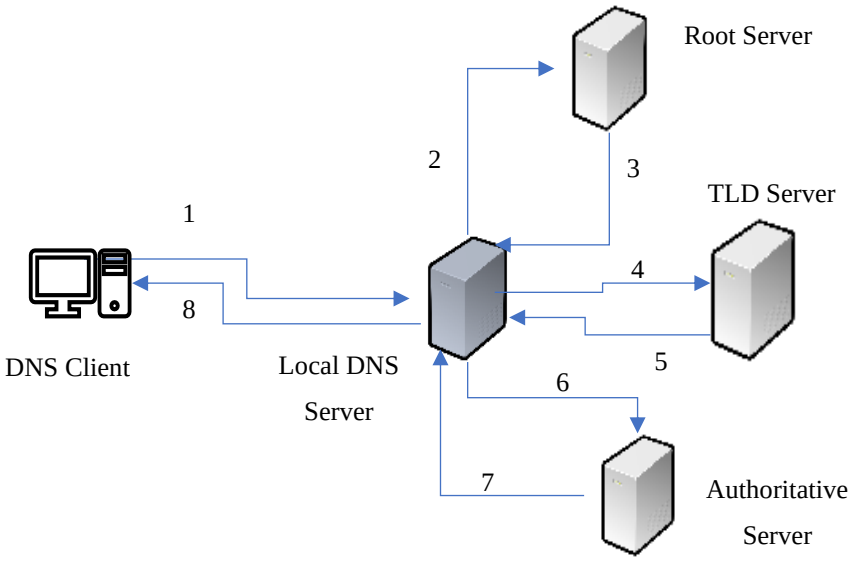


Figure 1.9 Recursive DNS query

1.5. Domain Name System

DNS service is an essential facility used by all the applications to connect to the remote server [45] [46]. DNS follows a client-server model wherein the client which wishes to connect to a Fully Qualified Domain name (FQDN e.g. www.thapar.edu) sends a DNS request to local DNS Server. The Local DNS server on receiving the request; parses and send the IP address mapped to the domain name. DNS uses a recursive query model in the sense that the Local DNS Server check the presence of Domain in its local cache. If the domain exists in the cache, the reply is sent to the client. Otherwise, its queries the Domain with other DNS servers as shown in Figure 1.9. in a recursive manner until the domain is resolved. The root DNS server is queried first for the nameserver of the top-level domain (TLD). If the request is resolved successfully, the TLD name server is queried for the name of the authoritative nameserver for the domain. Finally, the authoritative nameserver is queried for the IP address of the domain.

Packet capture for DNS request and response is shown in Figure 1.10 and Figure 1.11 using *wireshark*. It also displays the values of fields in the DNS format. A client sends a DNS request to Local DNS Server asking the IP address (Host address) corresponding to "www.thapar.edu". The request contains one question and carries a transaction ID of "0x5f2e". The Server sends a response stating that "www.thapar.edu" maps to "14.139.242.109" and "117.203.246.71" with Time to Live (TTL) value of 4245 seconds. The response consists of 1 question and two answers with transaction Id ("0x5f2e") equal to the ID send in the request. The transaction ID is the key value which maps the request to the response. TTL is an expiration date of the DNS record in seconds which indicates the number of seconds the record is valid. DNS query format is represented in Table 1.1.

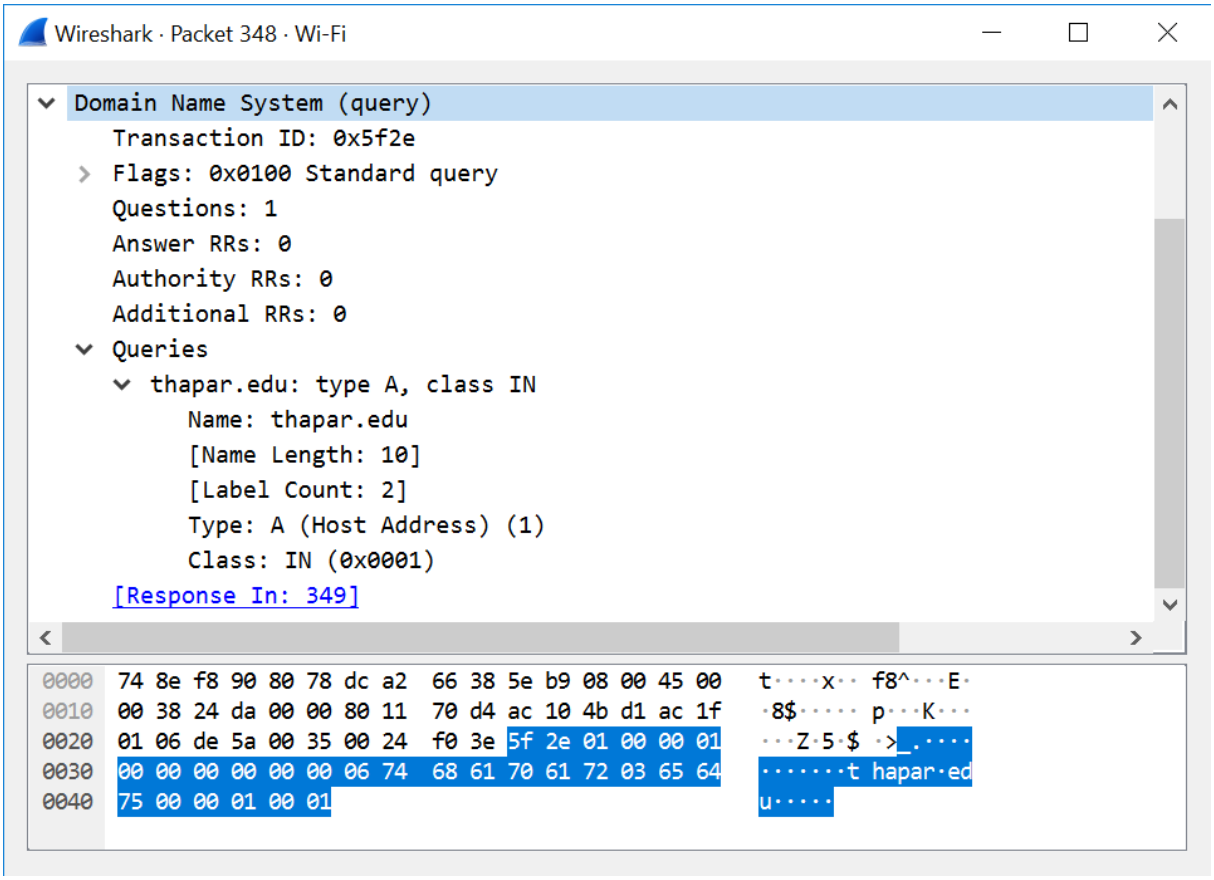


Figure 1.10 DNS query fields

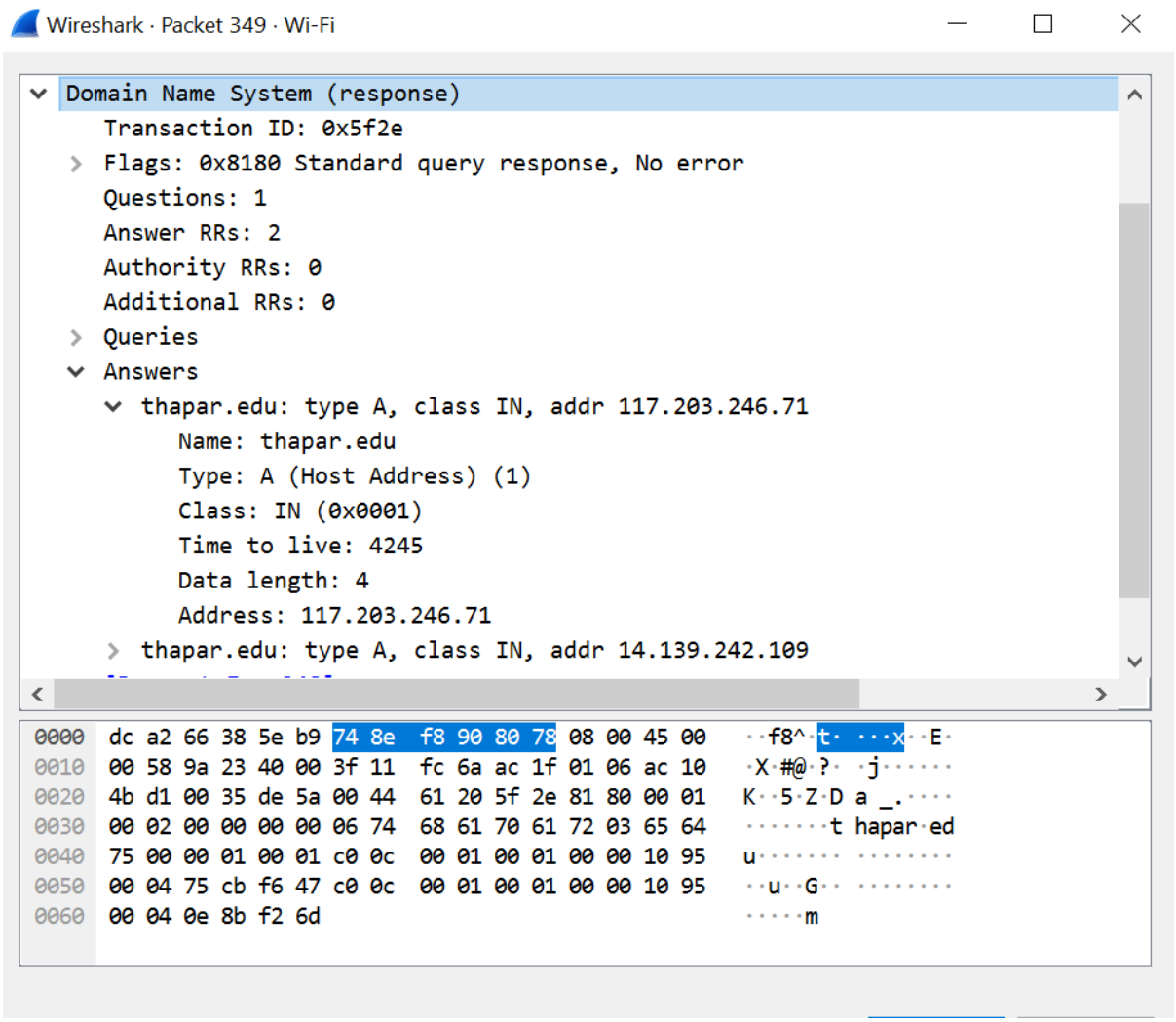


Figure 1.11 DNS response fields

Bot binaries are programmed to connect at the rendezvous point. DNS service is used since hard-coding IP address can be easily detected by honeypots. Domain generation algorithms are used by bots to evade DNS blacklisting where several algorithmically generated domains are used as C&C Server. Since the algorithm is known to botmaster and bots, the botmaster can register these domains in advance and as such can use these domains for C&C server for a certain period before shifting to the next domain. Bot-generated traffic can be differentiated from normal traffic by finding deviations of variable parameters in DNS traffic. For instance, the TTL value in the DNS response record tells the time for which the record can be cached before discarding. TTL values tend to be smaller for Flux networks as longer time help in detection.

Table 1.1 DNS query format

DNS Query Format		
Header Section	ID	Unique ID per query
	Flags	Indicates whether request/ response/ Authoritative answer/ truncation, etc.
	Number of Questions	In question segment
	Number of Answer Response records	In answer segment
	Number of Authority Response Records	In Authority segment
	Number of Additional Response Records	In extra records segment
Question	Queries	Consists of Query Name, Query Type, and Query class.
Answer	Answers	Consists of Name, type, class, TTL, Response Data Length and Response Data
	Authority Answers	Consists of Name, type, class, TTL, Response Data Length and Response Data
	Additional records	Consists of Name, type, class, TTL, Response Data Length and Response Data

1.5.1. DNS Flags

DNS flags are used for customizing the DNS request and for using advanced features in DNS protocol. The Query/Response (QR) bit, if set to 0, indicates that it is a DNS query. If the QR bit is set to 1, it indicates that it is a DNS response. DNS Opcode is used to specify the operation code e.g. query, status, update. Authoritative answer flag, if set to 1, indicates that the nameserver is an authority for the domain name. The truncate flag indicates whether the message is truncated or not. Recursion desired flag (optional feature) indicates the nameserver to query recursively until the domain is resolved. Recursion available flag is used by the nameserver to indicate whether recursive query provision is available or not. Authenticated data flag indicates whether the data in the answer section is authenticated by the nameserver as per its policy or not. Checking Disabled flag indicates whether the non-verified data is tolerable to the resolver or not. Various DNS flags are shown in Figure 1.12.

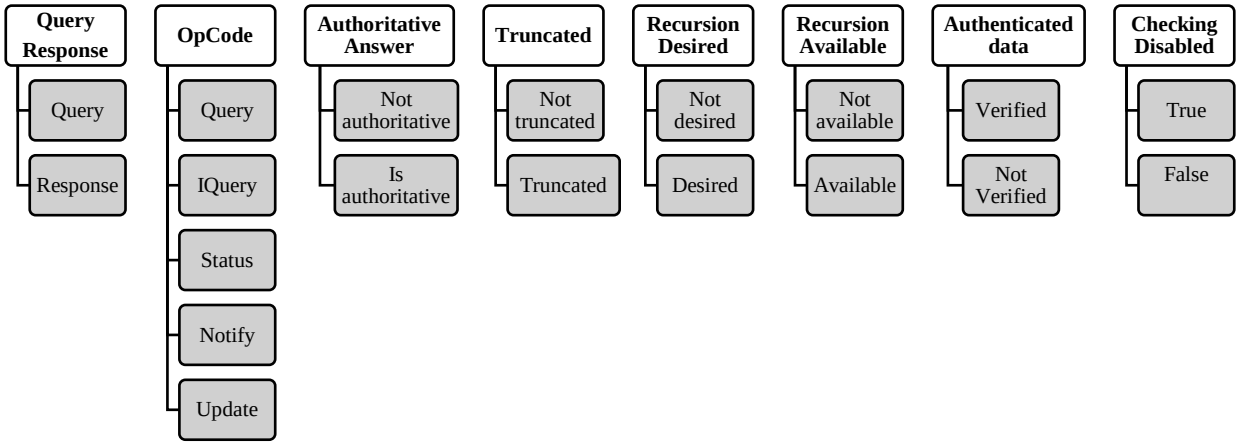


Figure 1.12 DNS flags

1.5.2. DNS Request Types

DNS protocol can be used for a variety of operations like resolving the domain name to an IP address, finding the Authoritative nameserver of a domain, requesting the mail exchange of a domain, etc. These operations can be performed by specifying the DNS Resource Records type in the DNS query. Commonly used DNS resource records types are shown in Table 1.2. A detailed listing is shown in [47], [48].

Table 1.2 Common DNS resource records type

Type	RR Type Value	Description
A	1	IPv4 address
NS	2	Authoritative name server
CNAME	5	Canonical Name /Alias
SOA	6	Start of Authority
PTR	12	Reverse Name Resolution
MX	15	Mail Exchange
TXT	16	Text string
AAAA	28	IPv6 Address

1.6. Resilience

All bots are programmed to connect with rendezvous point in rallying phase as shown in Figure 1.13. Hardcoded IP address is the simplest way to connect to rendezvous point but offers limited resilience as blacklisting the IP address at the firewall level can cripple the botnet. Static DNS provides the botmaster with the flexibility to add another layer of abstraction. It too offers limited resilience as the mapped IP address can also be blacklisted.

Dynamic DNS can provide multiple and dynamic IP addresses. This mechanism can also be controlled by updating a blacklist as provided by security vendors. Botmaster uses a flux mechanism to continuously change domain and Internet Protocol address of the Command and Control server to avoid evasion.

Higher resilience is achieved when bots use an algorithm to generate domain names and the botmaster fluxes the C&C Server to provide redundancy. This technique offers a high degree of resilience as domains generated algorithmically are used for only a little period and discarded thereafter. This technique thwarts the blacklisting mechanism used earlier to prevent botnets. Recently the use of Social Networking services like Twitter as C&C Server has set the alarm bells ringing as far as security of network infrastructure is considered.

Hands et al. [49] analyzed the malicious use of DNS by the cybercriminals and potential hints to detect such misuse. Domain-flux (multiple Fully Qualified Domain Name (FQDN) mapping to the same Internet Protocol address) and IP Flux (Continuous change in IP addresses resolved to a domain) are two main services used by the botmaster to build resilience. DNS signaling, on the other hand, hides stolen usernames and password in DNS request. While DNS tunneling can encode an entire protocol like HTTP and FTP, a Virtual Private Network (VPN) can be constructed between the botmaster and the bot using this technique.

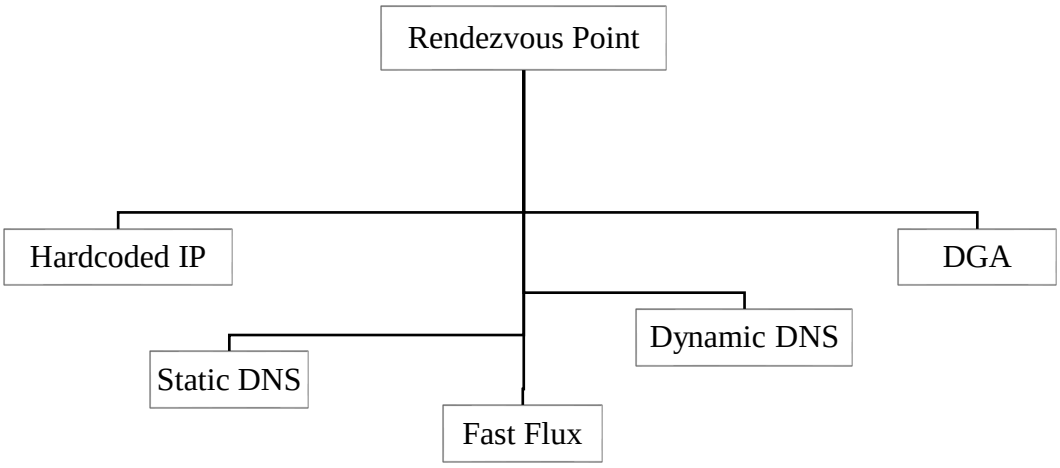


Figure 1.13 Techniques for rendezvous point

Domain generation algorithms [50] are used by bots to evade DNS blacklisting wherein several algorithmically generated domains are used as shown in Figure 1.14. Since the generation algorithm is known to the botmaster and the bots, the botmaster can register these domains in advance and as such can use these domains for C&C server for a certain period before shifting to a new domain. A detailed analysis of the algorithm used by DGA botnet is presented in [51] and [52]. Fu et al. [50] presented an in-depth analysis of DGA used by popular botnets.

```
DNS        71 Standard query 0x160f A xnnhcuty.sx
DNS        70 Standard query 0xe785 A wxopsrl.tw
DNS        78 Standard query 0x893b A weupkieiivdvcha.co
DNS        84 Standard query 0x0ebe A smyslivblmefjuelrpdmm.to
DNS        73 Standard query 0xd679 A ssofntbnbh.bz
DNS        74 Standard query 0x282d A cmapmeixdtk.co
DNS        70 Standard query 0xe785 A wxopsrl.tw
DNS        71 Standard query 0x56bc A ulfcnpp.pro
DNS        78 Standard query 0x893b A weupkieiivdvcha.co
DNS        74 Standard query 0x282d A cmapmeixdtk.co
DNS        74 Standard query 0xf127 A njnoulftlkh.tw
```

Figure 1.14 Host querying domains generated by DGA

Domain Generation Algorithms make use of specific inputs like the year, month and day to compute a domain which is known to the botmaster and embedded in bot binary. These inputs are permuted in such a manner to make an analysis of the algorithm difficult. Simple implementation and result of the DGA algorithm are shown in Figure 1.15.

Algorithm *Get_DGA_Domain*	***Result:***
Input*: year, month, day*	*Date FQDN*
Output*: fully qualified domain name*	*2018/8/1 aciidfce.com*
1: fqdn = ""	*2018/8/2 ichihhae.com*
2: ***For each*** *i* ***in*** *range(8):*	*2018/8/3 gcfajhaa.com*
/ Permute all inputs 8 times */*	*2018/8/4 eceadieg.com*
3: year = ((year ^ 10) >> 20) ^ ((year & 0xFFFFFFFF) << 3)	*2018/8/5 ccccfiec.com*
*4: month = ((month ^ 40) >> 50) ^ 60 * (month & 0xFFFFFFFF)*	*2018/8/6 acbcjacc.com*
5: day = ((day ^ 20) >> 5) ^ ((day & 0xFFFFFFFF) << 9)	
6: fqdn += chr(((year ^ month ^ day) % 10) + 97)	
/ Convert number to character */*	
7: ***End For***	
8: fqdn += '.com'	
/ add Top level domain */*	
9: ***return*** *fqdn*	

Figure 1.15 Sample DGA algorithm

While bots use algorithms to generate domain names, the botmaster fluxes the C&C server to provide redundancy. This technique offers a high degree of resilience as domains generated algorithmically are used for only a short period of time and discarded thereafter. This technique thwarts the blacklisting mechanism used earlier to prevent botnets. Recently the use of social networking platforms (e.g. Twitter [37] and Instagram [53]) and software development platforms (e.g. Github [54]) as a C&C server is complicating the security of network infrastructure.

1.7. Research Motivation

Botnets are the backbone for cybercrimes on the Internet. While law enforcement agencies across the globe manage to shut a certain botnet, a new one comes up with a different name. It is a persistent threat that changes the name and comes back stronger repeatedly. Economic losses due to these malicious activities are continuously rising. Botnet detection as such is the need of the hour. Both public and private sectors are affected by these malicious activities and as such devising strategies to mitigate the risk. A “Botnet Cleaning and Malware Analysis

Centre" named "*Cyber Swachhta Kendra"* [55] was established (Feb-2017) by the Government of India to create safe cyberspace in India.

One of the prime reasons for the existence of a large number of bot infections even today is the evolving nature of botnets. Botnets are continuously evolving and adding new stealth capabilities which are very difficult to detect. Besides this, new attack vectors are continuously added to the botnet. Use of social networking sites like Twitter, Instagram, etc. and development platforms like GitHub as C&C server is certainly difficult to detect. The exponential rise in network traffic is making real-time analysis difficult. Use of IoT devices to unleash DDoS attack is another research challenge.

After careful examination of the literature, the following research gaps are identified.

1. Botnet is a continuous evolving phenomenon. Although there are varieties of detection techniques, yet there is a need to have a robust and quick detection technique which can detect the latest botnets.
2. Usage of Domain generation algorithms for Botnet C&C communication is making life difficult for network administrators to prevent or disrupt botnet infrastructure and as such is an immediate and pressing challenge.
3. To restrict the use of Social Networking Sites for C&C Server remains a tedious and open problem.
4. Detecting Botnet in real time amid exponential growth of network traffic is still a challenging research issue.
5. Detecting Botnet employing IoT (Internet of things) devices is a recent trend which has not been considered in the past and as such is a pressing threat.
6. Anomaly based detection remains the only option in the absence of blacklisting when dealing with botnet communication as domains are used for a very limited amount of time.

1.8. Objectives

The objectives of this research work are as below.

1. To explore and analyze existing bots those have a detrimental effect on the network and report their detection mechanism.
2. To propose a set of solutions (techniques and algorithms) for botnet detection based on DNS traffic Analysis**.**
3. To test and validate the proposed work.

1.9. Our Contribution

The main contributions of this research are:

- Current research on DNS-based botnet is classified into five categories: Flow-based, Anomaly-based, Flux-based, DGA-based and Bot infection based.
- A comprehensive review of DNS based detection techniques is conducted.
- It provides a deep analysis of each technique within the category by comparing the detection mechanism, whitelist/blacklist criteria, targeted botnet, and dataset used.
- Compares the DNS features used and detection rate for existing techniques within the category.
- Essential attributes of a smart DNS based botnet detection system are proposed.
- This research work presents a novel technique named BotDAD [56]: Bot DNS anomaly detector for detecting bot-infected machines in a network using DNS fingerprinting which considers the complete DNS activity of a host per hour.
- The technique has been successfully evaluated on the campus dataset between April and May 2016 and on the online dataset. High detection rate promises a strong possibility of large-scale deployment.
- Instead of relying on a single point anomaly for labeling DNS fingerprint as Malicious, an improved labeling technique was implemented which uses multipoint anomaly detection. This ensures in minimizing the number of false positives.
- Two deep learning frameworks namely Scikit-learn [57] and Tensorflow [58] were used to validate the proposed model showing improved results.
- A graphical version named DeepDAD is presented which ensures ease of use in experimentation and comparison with future techniques.

1.9.1. Technologies Used

Python programming language was used for the implementation of this research work. The prime reason for selecting python was the availability of the rich ecosystem of packages for a wide variety of tasks like packet processing, machine learning, plotting, etc. The following packages were used in this research work.

- *dpkt* is a python library [59] for quick packet processing along with the definitions of common TCP/IP protocols. This library was used to process network traffic and extract DNS features.
- *geoip2* is a python library [60] to access GeoIP2 web services and database. This library was used to find city and country details of an IP address.
- *Matplotlib* is a python library [61] for creating figures. It offers a variety of plotting options like histogram, bar chart, scatter plot, heatmap, etc.
- *GephiStreamer* is a python library[62] to stream graph elements to Gephi (an open graph platform) [63]. This library was used to visualize malicious FQDN-IP mappings.
- *Scikit-learn* is a machine learning framework [57]. It provides various libraries for data mining and analysis. It is easily reusable and is compatible with popular python libraries like NumPy [64], SciPy [65] and Matplotlib [61]. This framework was used for machine learning.
- *Tensorflow* is a powerful machine learning platform [58]. This platform was used for deep learning.

1.10. Thesis Outline

The rest of the thesis is organized as per the following chapters:

Chapter 2: Literature Survey

It reviews the literature in the field of Botnet detection using DNS protocol. It starts with the motivation for research followed with research questions. It also includes the survey technique used to perform the literature survey. In the next section, a brief discussion is done on botnet

detection techniques. Chronological order of botnet detection techniques using DNS protocol is also discussed in this chapter. Botnet detection can be broadly classified into Flow-based, Anomaly-based, Flux-based, DGA-based and Bot infection detection based. Flow-based detection techniques attempt to classify the network flow into malicious and benign based on various parameters inspected in the network flow. Anomaly-based detection techniques attempt to find an anomaly in the various parameters or peculiar patterns in the traffic which is dissimilar from normal network behavior. Flux-based detection techniques try to find IP flux in network traffic wherein there is a continuous change in the IP Map associated with a domain and generally having very low TTL value in DNS resource record. DGA-based detection techniques attempt to differentiate domains queried in a network which are algorithmically generated (malignant) from the normal domains. Very recently, there are attempts being made to detect infected machines in a network instead of finding the C&C server. Bot infection detection techniques attempt to find bot-infected hosts in a network. Essential attributes of a smart DNS based detection system are also discussed in this chapter. Finally, it compares the DNS features used by various techniques. The detailed survey is published in [66] and this chapter is part of that paper with more additions.

Chapter 3: Bot DNS Anomaly Detector (BotDAD)

This chapter starts with the architecture of the Bot DNS Anomaly Detector. This chapter then elaborates the various modules in the architecture. It also includes the algorithms used in various modules to explain the internal working of the system. The BotDAD architecture consists of two main subsystems: 1) DNS Fingerprinting module, and 2) Anomaly detection engine. DNS Fingerprint generation module generates hourly DNS fingerprint of each host in the network by inspecting network DNS traffic. This fingerprint is then analyzed by anomaly detection engine which classifies it into two categories i.e. bot or clean based on the presence or absence of an anomaly. In the next section, details of DNS features used for DNS fingerprinting are discussed. Various features used for DNS fingerprinting can be broadly classified into the request, response, domain, IP and mapping type features. The detailed technique is published in [67] and this chapter is part of that paper with more additions.

Chapter 4: Implementation and Experimental results

This chapter starts with the description of the dataset used for experimentation. Day-wise and hour-wise details of the dataset are also discussed in this chapter. The problem encountered in measuring the number of DNS request for loading a webpage is also discussed in this chapter. It then describes the importance of features as a result of the RF algorithm. Finally, error metrics used for validation are also discussed in this chapter. To improve the accuracy of the model, hyperparameter tuning is also described in this chapter. The detailed implementation and results are published in [67] and this chapter is part of that paper with more additions.

Chapter 5: Deep learning-based Bot detection

This chapter starts with the introduction to the deep neural networks explaining the commonly used activation functions. Two machine learning frameworks namely Scikit-learn [57] and TensorFlow [58] were used to train and test the model with enhanced labeling technique which uses multipoint anomaly detection. Effect of hyperparameters on evaluation metrics is also discussed in this chapter. A graphical version of the tool named DeepDAD is presented which ensures ease of use in experimentation and comparison with future techniques.

Chapter 6: Conclusions and Future Directions

Conclusions and future directions are discussed in this chapter. This chapter concludes that the BotDAD can successfully detect bot infection using DNS fingerprinting. The chapter further concludes that detecting DGA based botnets require additional domain features. Additional features, in turn, lead to higher computation time which makes real-time detection difficult. Usage of machine learning techniques presents a possible solution to the problem of botnet detection. The problem, in this case, is the unavailability of the labeled real-world dataset for evaluation purpose which is currently not available in abundance. A dataset from virtual setup doesn’t completely mimic the real-world data and as such do not fit for real-time detection.

2. Literature Survey

2.1. Introduction

There are a considerable number of surveys on Botnet detection [22], [23], [41], [68]–[73]. Few surveys [74][75] focuses exclusively on the use of DNS protocol for Botnet detection. Since there is a regular addition to the volume of research in botnet detection, there is a strong need for a comprehensive and comparative evaluation of research. Although a comprehensive literature survey is a difficult task especially in Botnet detection considering the volume of publications, yet such a survey is required which provides an insight into the status of the research. This research provides an organized and efficient analysis of various Botnet detection techniques employing DNS Protocol, discusses the current threat landscape and reports major findings.

This chapter starts with the motivation for the research. Section 2.3 describes the survey technique used to conduct this research. Research questions are listed in section 2.4. Various sources of information and the search criteria used for this research is covered in section 2.5. Various parameters extracted for comparison are discussed in section 2.6. An overview of botnet detection techniques is covered in section 2.7. A comprehensive classification of botnet detection techniques using DNS is covered in section 2.8. Finally, discussion on dataset location, essential attributes of a smart DNS based botnet detection system, comparison of DNS features is presented in section 2.9.

2.2. Motivation for Research

- Botnets are in continuous media attention as it severely cripples' businesses, services, and operations.
- While law enforcement agencies across the globe coordinate to shut down a certain botnet, a new one comes up with different name employing new attack vectors and using

a different set of devices. It is a persistent threat that changes the name and comes back stronger repeatedly.

- Although Botnet detection is widely explored with many detection techniques, yet there is no end to this menace.
- There is a strong need to compare these techniques vis-à-vis considering the DNS features, detection rate, and the dataset used for detection.

2.3. Survey Technique

This research provides a systematic review of various DNS based botnet detection technique. The review process includes various steps to pick the most relevant contributions by the researcher in this domain. These include review questions, sources of information, search criteria, inclusion and exclusion criteria, conducting the review, results analysis, result reporting, and discussion as shown in Figure 2.1.

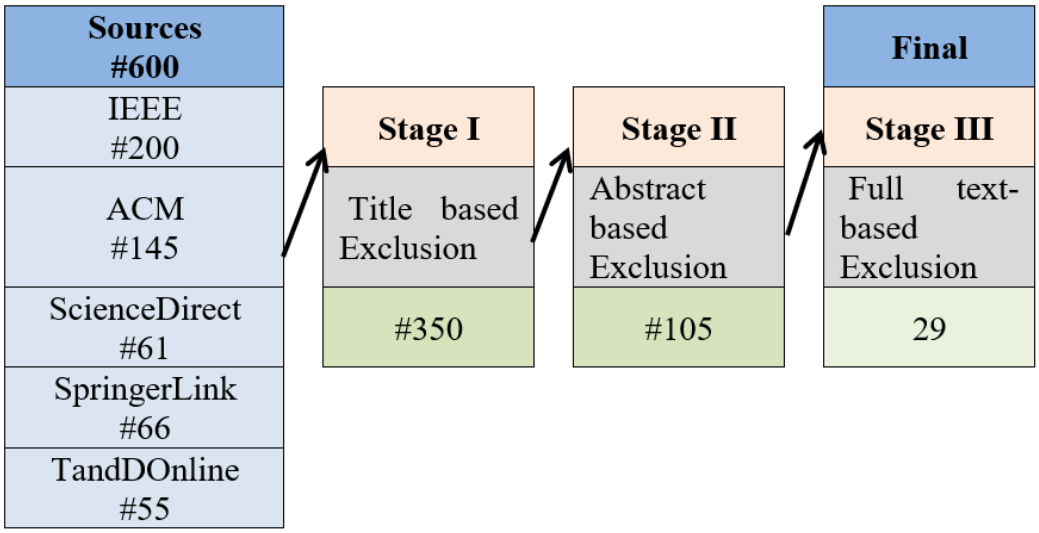


Figure 2.1 Study selection criteria

2.4. Research Questions

The very purpose of this review is to classify botnet detection techniques based on the DNS protocol. Research questions, needed for the proper planning of the survey, are listed in Table 2.1.

Table 2.1 Research questions

1. What is the status of Botnet infection?
2. Timeline and evolution of Botnets over the years?
3. How is DNS abused for Botnet Communication?
4. What is the possible categorization of DNS based botnet detection techniques?
5. What are the techniques available for detecting DGA based botnet?
6. How have these techniques evolved in the last decade?
7. What is the research status of Anomaly-based Botnet detection?
8. What are the various datasets used for botnet detection?
9. What is the error metrics used for validating detection techniques?
10. What is the ground truth problem of botnet detection in the real dataset?
11. What are the essential attributes of a Botnet detection system?
12. What are the various DNS features used?
13. How far has machine learning been used for botnet detection?
14. What are the open issues in DNS based botnet detection?
15. In what manner, the botnet problem can be completely solved?

2.5. Sources of Information and Search Criteria

The purpose of this review is to report botnet detection techniques that rely solely on DNS-based features for detection. As such, techniques based on some form of analysis of DNS protocol were included. Techniques which don’t rely on DNS protocol for detection were excluded.

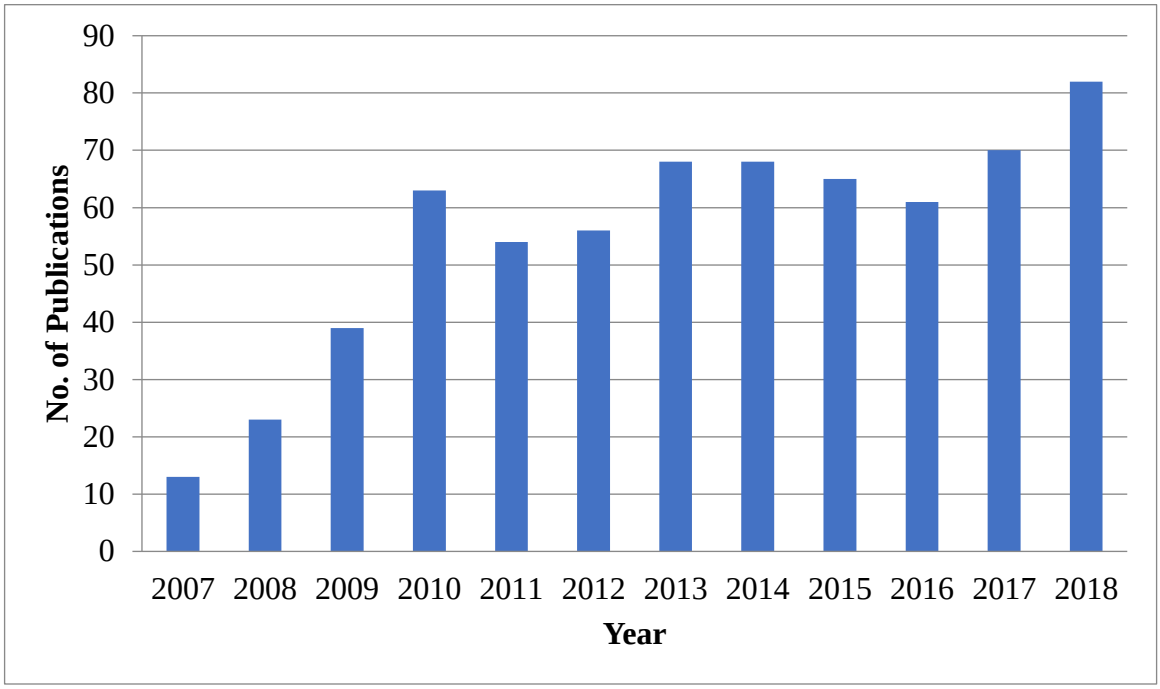


Figure 2.2 Year-wise number of publications

Table 2.2 presents the number of publications in the various sources searched for this review.

Table 2.2 Search token

S. No	URL	Search Token	Dates	Types	No. of Publications
1	https://link.springer.com/	Botnet DNS	2005-2017	[C B J]	479
2	http://dl.acm.org/	Botnet DNS	1983-2017	[J M B]	651
3	http://www.sciencedirect.com	Botnet DNS	2003-2017	[J B]	487
4	http://ieeexplore.ieee.org	Botnet DNS	2007-2017	[C J]	74
5	http://www.tandfonline.com/	Botnet DNS	All Dates	[J]	20
{C - Conference J- Journal B- Book M - Magazine}					1711

Year-wise number of publications which include botnet detection in the title is shown in Figure 2.2.

2.6. Data Extraction

In the data extraction phase, the following parameters were extracted from the selected publications:

- Detection technique: Various techniques are used for Botnet detection. While some techniques are based on clustering or classification, other techniques use measures like distance or similarity to detect botnet traffic.

- Dataset: Since dataset involved in botnet detection plays a crucial role. It is important to check the source of the dataset along with the duration and location of the dataset. It has been observed that techniques tested on multiple datasets are bound to produce better results than those tested on a single dataset. Hosts count, traffic volume, and location of the DNS sensors can also be a significant parameter when evaluating a technique.

- Targeted Botnet: The versatility of the technique can be measured by the types of botnets detected by the system. It has been observed that techniques which are validated for more than one botnet are more promising than those targeting only a single botnet.

- Whitelist and blacklist criteria: Different techniques use different criteria for filtering benign traffic. Filtering traffic can significantly reduce the computation required for detection. Filtering is mostly based on using blacklist and whitelist.

- Detection rate: Detection rate is an important parameter for comparing various techniques as it provides an insight into the effectiveness of the detection system.

2.7. Botnet Detection Techniques

The purpose of this research is to explore the advances made in DNS-based Botnet detection as per the review questions and report findings in a systematic and organized way. Botnet detection has got a substantial focus from industry and academia [68], [72], [76]–[81]. Figure

2.3 depicts the classification of botnet detection techniques which are classified into Signature-based and Anomaly-based.

- Signature-based techniques make use of a signature database to check for known botnets. These detection techniques are very quick and work well for known botnets and tend to have very low false positives. However, such techniques have a limited capability to detect unknown botnets. Botmaster frequently updates bot binaries to evade signature-based detection. Goebel et al. [82] presented a signature-based detection system named Rishi. Using n-gram analysis and counting mechanism, the technique checks for doubtful IRC handles, servers and infrequent server port numbers to report the existence of bots in a network.

- Anomaly-based detection is being used in all research areas[83]. Chandola et al. [84] have very elegantly articulated the need and use of anomaly detection in various research areas. Anomaly-based botnet detection attempts to find anomalous patterns in the Host and Network behavior like high traffic volume, increased latency, unusual system behavior, etc. These techniques are skillful of detecting novel or unidentified botnets. These techniques are further classified into Host-based and Network-based.

Host-based anomaly detection techniques attempt to find an anomalous pattern in the system calls, socket calls, registry key, etc. Stinson et al. [85] presented a host-based discovery prototype named BotSwat. The study postulated that the system call arguments resemblance with network data indicate that the system is being remotely controlled. The prototype system can monitor the execution of Win32 library functions and checks the network data as it propagates through a system call. The result indicated the presence of bots in a network based on the network data present in system call parameters.

Network-based anomaly detection, on the other hand, attempts to find patterns in the traffic flow which are distinguishable from the normal flow. Since bot needs to connect to the rendezvous point in various phases of botnet lifecycle, this creates additional traffic which can be analyzed either in real-time (active) or after capture (passive).

Active anomaly-based detection attempts to find an anomaly in real-time. Gu et al. [86] presented a technique to detect botnet by active probing based on the cause-effect correlation. The volume of network traffic has climbed sharply over the last few years thereby making real-time analysis very difficult. Moreover, the increasing complexity and obfuscation techniques used by the botmaster are making it difficult to detect in real-time. A considerable amount of processing and state information is needed to detect botnets.

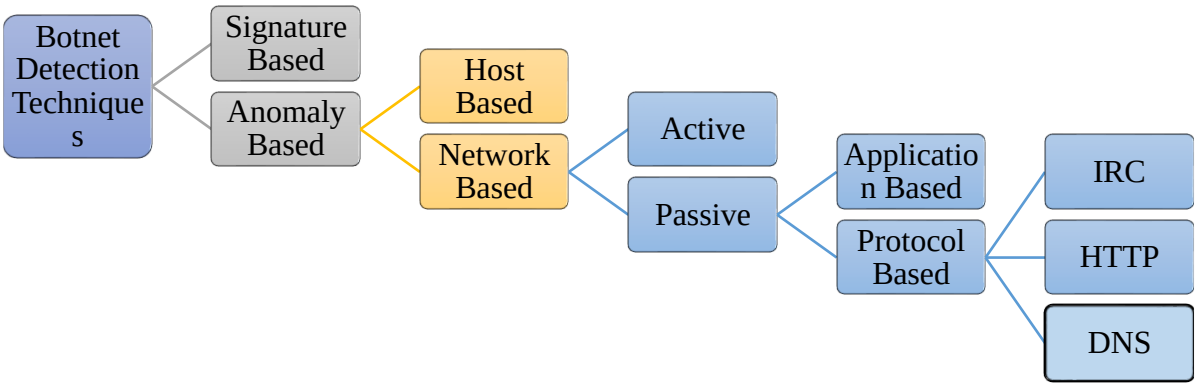


Figure 2.3 Botnet detection techniques

Passive Anomaly-based detection has received a lot of attention as it involves capturing network flow for analysis at a later stage without worrying about the packet drop or real-time computational complexity. Qiben et al. [87] presented PeerClean for detecting P2P botnet. There is a significant amount of computational complexity involved in deep packet inspection. Further, encryption thwarts all attempt for the same. As such various flow parameters are extracted and analyzed using machine learning model. The proposed model uses both supervised and unsupervised i.e. clustering as well as a classification for detection. The dataset used in this research consists of data recorded from the edge router of a large network

comprising of two subnets. Support Vector Machine (SVM) is used to cluster bots using P2P network belonging to the same botnet.

2.8. Year Wise Evolution of DNS-based Botnet Detection Techniques

Botnet detection can be broadly classified into Flow-based, Anomaly-based, Flux-based, DGA-based and Bot infection detection based. Flow-based detection techniques attempt to classify the network flow into malicious and benign based on various parameters inspected in the network flow. Anomaly-based detection techniques attempt to find an anomaly in the various parameters or peculiar patterns in the traffic which is dissimilar from normal network behavior. Flux-based detection techniques try to find IP flux in network traffic wherein there is a continuous change in the IP Map associated with a domain and generally having very low TTL value. DGA-based detection techniques attempt to differentiate domains queried in a network which are algorithmically generated (malignant) from the normal domains reported first by Stone-Gross et al. [88].

Very recently, there are attempts being made to detect infected machines in a network instead of finding the C&C server. Bot infection detection techniques attempt to find bot-infected hosts in a network. Classification of DNS-based Botnet Detection Techniques is shown in Figure 2.4.

2.8.1. Flow-based Detection

Flow-based detection techniques attempt to classify the network flow into malicious and benign based on various parameters inspected in the network flow. These techniques rely on detecting deviation originating in the network flow due to the introduction of malicious flow arising out of botnet traffic. Intrusion Detection System (IDS) like snort [89] is generally used to analyze the traffic and generate an alert. Common parameters used in flow-based detection include the number of packets, total packets, bytes/packet, bytes/second, number of three-way TCP handshake, number of connection teardown, etc. The prime focus of flow-based detection techniques is to inspect network flow and detect botnet communication.

State of the Art

The earlier DNS based detection technique started with the DNS-based Blackhole List (DNSBL). DNSBL maintains a list of domains which are malicious. Ramachandran et al. [90] successfully demonstrated a DNSBL counter-intelligence technique for detecting bot membership. The approach makes use of the fact that botnets are using counter-intelligence to check if their IPs are listed or not before sending spam. Moreover, selling bot at a premium requires all the bot members to be clean (i.e. not present in the DNSBL). The dataset from Bobax Sinkhole (Nov 17, 2005 – Dec 31, 2005) was used along with the weighted query graph for finding the bot membership. The focus of this research was limited to spamming botnets.

Strayer et al. [91] presented a detection technique using traffic flow features like bandwidth, duration, and packet timing. This approach is based on finding the botnet movement by checking traffic only at the core network locations. The dataset from Dartmouth campus in the CRAWDAD project [92] was used. The botnet test facility comprising of IRC and code server, 13 infected machines and an attacker was used. Five filters, namely, TCP flow-based, port scanning filter, bulk data transfer filter, 300 Bytes packet size cutoffs, and brief flow filter were applied. Traffic flow characteristics like TCP flags, total packets, total bytes, duration, bytes/packet, bits/second were collected to analyze the data. Classifier (J48, Naïve Bayes, and Bayesian Networks) and correlation (5-space Euclidean distance) were used to classify and correlate the data. The research highlighted the need for a promising training set when using machine learning to detect malicious and non-malicious traffic. The result showed that Bpp (bytes per packet) has the highest biased power among all attributes.

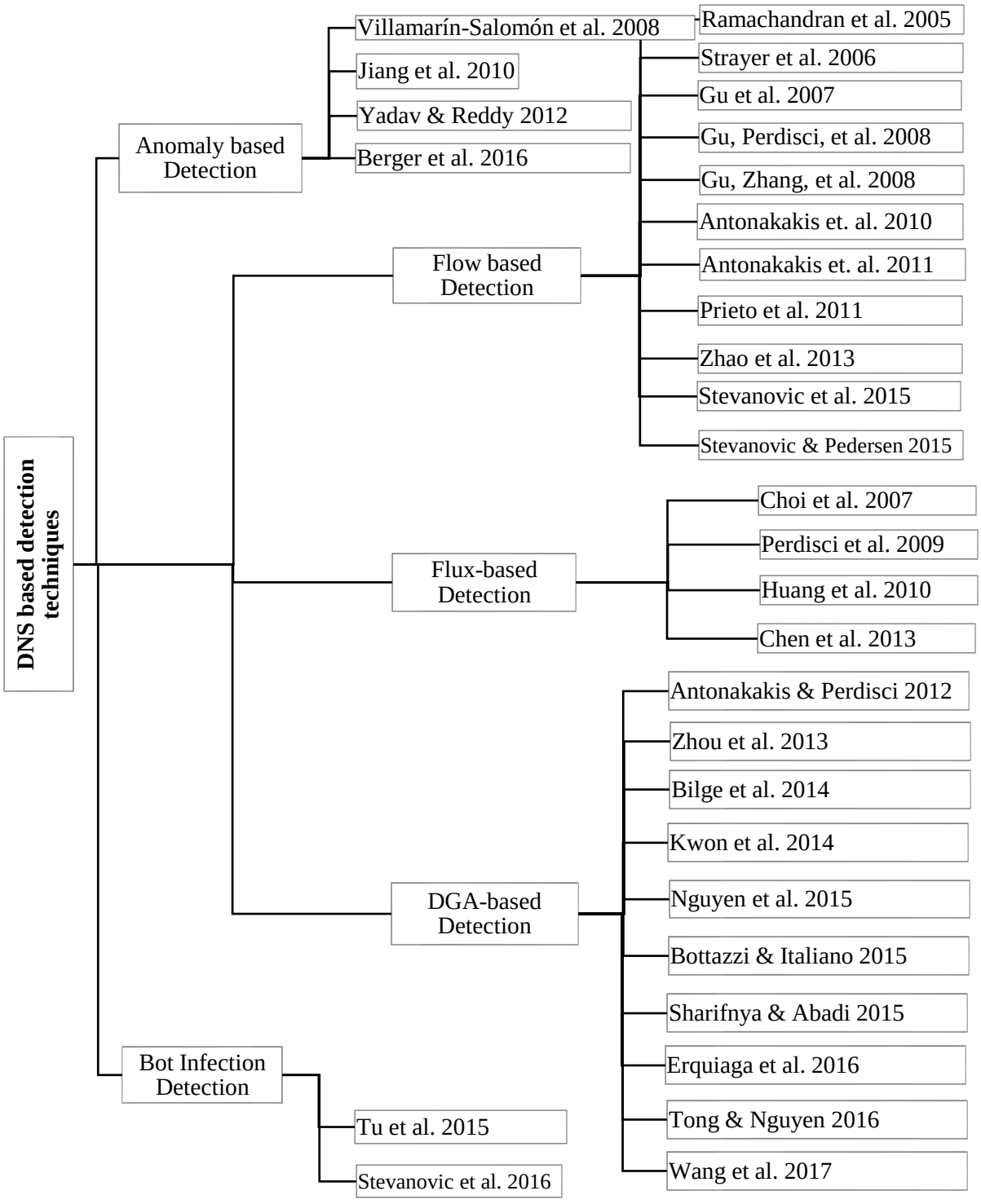


Figure 2.4 DNS-based botnet detection techniques

Gu et al. [93] presented a correlation engine for malware infection detection called BotHunter as shown in Figure 2.5. An evidence-trial approach was used to detect a malicious infection that occurs in communication. Bot infection process follows a series of steps which can be correlated using the correlation engine. Infection process typically consists of scanning, vulnerability exploitation, Trojan/backdoor download, C&C communication, update and command execution. The system looks for various activities that relate to phase and collects evidence per internal host. Finally, the Intrusion Detection System (IDS) based dialog correlation was used to detect botnets. Bots can, however, evade detection from BotHunter by using encrypted C&C.

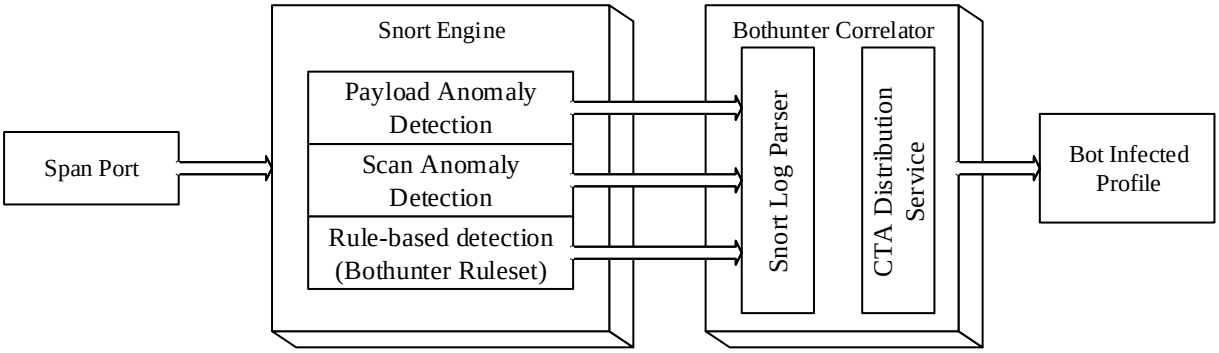


Figure 2.5 The architecture of the BotHunter system

Gu et al. [94] presented an anomaly-based detection system for identifying push (i.e. IRC) and pull (i.e. HTTP) style of Command and Control Communication named BotSniffer. Snort was used as IDS for sniffing the message between a bot and C&C Server. The proposed technique consists of an Activity Detection System which looks up for botnet related activities e.g. scan, spam, and binary downloading and protocol match on network traffic to create an activity log which acts as an input for correlation engine. The correlation engine groups the clients which connect to the same server and performs "spatial-temporal correlation and similarity detection". It used "response crowd density check algorithm" to perform analysis to report an alert. Encrypting the C&C communication and random noise injection can be used to evade detection from BotSniffer.

Gu et al. [95] developed another detection system for Centralized and P2P botnets named BotMiner. The system monitors two main things in a network traffic i.e. i) "who is talking to

whom (C-Plane)" ii) "Who is doing what (A-Plane)" and attempts to find malicious activities in a network. C-Plane grouping attempts to cluster hosts which share comparable communication between groups. While A-Plane clustering group hosts which share similar activities. Finally, a cross-plane clustering is used to find an anomaly. Cross plane clustering is used to ensure more accuracy in detecting host participation in a botnet. Unlike BotHunter and BotSniffer, BotMiner can detect centralized (HTTP/IRC) as well as peer-to-peer botnets.

Antonakakis et al. [96] presented Notos, a system for detecting malicious domains. Notos calculates a dynamic reputation score for an unknown domain. Notos was evaluated against DNS traffic consisting of 14 million users. Three group of features namely network based, zone based and evidence-based were used to calculate the score. The system obtained a TPR of 96.8% and FPR of 0.38%. Notos had one limitation that it could not detect new domain and address space which is used only once for malicious purpose and never reused.

Antonakakis et al. [97] proposed another system to detect malicious domains in upper level of DNS hierarchy named Kopis. Kopis used request diversity, requester profile and resolved IP reputation based statistical features to calculate score. The system was evaluated against DNS traffic from two major domain name registrar for eight months in 2010.The detection rate of Kopis was reported to be 98.4% and low FP rate of 0.3%. Kopis had one limitation that it used long duration of the observation period i.e. a day. DGA domains are used for a very short period and could be inactive by the time it is reported by Kopis.

Table 2.3 Flow-based detection

Reference	**Technique**	**Whitelist/ Blacklist Criteria**	**Targeted Botnet**	**Dataset**	**Pros / Cons**
Ramachandran et al. 2005	Weighted Query graph	Well known Blacklist	Bobax (Spamming)	Bobax Sinkhole (Nov 17, 2005 – Dec 31, 2005	Limited to spamming Botnets
Strayer et al. 2006	Classification, correlator, and topology analysis.	Top sites	Kaiten	Dartmouth Campus under CRAWDAD Project (1st Nov 2003 to 28th Feb 2004)	Limited to IRC based Botnet

Gu et al. 2007	IDS-Driven Dialog Correlation	-	Phatbot, Rxbot, Sxbot & Ago bot	Campus network traffic from Oct 2006 to Feb 2007	Easy evasion using encryption
Gu, Zhang, et al. 2008	Spatial-Temporal Correlation and Similarity	Custom Whitelist and Watchlist	Rbot, Spybot & Sdbot	Traffic from University Campus Network for 189 days between 2005-2007	Low false positive rate. Limited to HTTP/IRC based botnets
Gu, Perdisci, et al. 2008	C-Plane and A-Plane Clustering and cross Plane Correlation	Whitelist comprising of US Top hundred and Global hundred websites from Alexa	Spybot, Sdbot, Rbot, Nugache & Storm	10 days of campus network traffic in late 2007	High detection rate and low false-positive rate. Can detect HTTP / IRC / P2P botnets
Antonakakis et al. 2010	Dynamic reputation score	Alexa, malwaredomains.com, malwaredomainlist.com, Zeus tracker	Torpig, Kraken & Sribzi	27 million unique resolutions from two ISP	Unable to detect new domains and address spaces
Antonakakis et al. 2011	Statistical classifier	Malwaredomains.com, Zeus Tracker	Zeus	Eight months DNS traffic from two major domain name registrars and Two months traffic from country code TLD	Ability to detect malware domains independently where no IP reputation is available
Prieto et al. 2011	Suspicion rate on extra domain information collected using Whois, Dig, Wget & DNSDump	-	Zeus, Confick er & Kraken	Trace from Public University of Navarre (UPNA, Spain)	Low detection rate.

Zhao et al. 2013	Decision tree classifier	-	Zeus, Storm, Walede. C & BlackEnergy	Datasets from the French section of Honeypot Project and "Ericsson research in Hungary and Lawrence Berkeley National Laboratory"	Independent of group activity. Vulnerable to new regular network traffic flow.
Stevanovic & Pedersen 2015	Random Forests classifier	-	Storm, Waledac & Zeus	Non-Malicious traffic trace from LAN Malicious trace from honeypot and malware testing environment	Correlation between different analysis methods not considered.
Stevanovic et al. 2015	DNSMap Clustering	Alexa popular domain Whitelist FQDN/IP Blacklist	Generic	A network trace of regional ISP from Denmark.	Manual validation makes it difficult for implementation in real-world applications

Prieto et al. [98] presented a system for Botnet detection named Botnet Detection System(BDS). Developed in C, BDS consists of network tools (like Net-Whois, wget, dig) and perl script for DNS traffic analysis. Blacklist data is obtained using a testbed system infected with Zeus, Conficker and Kraken botnet. DNS record age, DNS email record, authoritative DNS server, and web presence is used to calculate the suspect ratio. The technique used active probing to obtain extra information using domain tools and as such could be slow if the number of new domains is high.

Stevanovic et al. [99] presented a technique for botnet detection from network activity using Random Forest Classifier. TCP, UDP, and DNS Traffic are analyzed as part of Botnet detection. Various new parameters like number of three-way TCP handshakes, the number of connection teardowns, mean number of answers, and the mean number of authority answers were analyzed to detect for botnet presence in network traffic. Network traffic from 40 bot-infected machines

was also analyzed for malicious traffic evaluation. The research indicates that the length of the time window is a far more significant parameter in case of TCP than UDP. It further concluded that 1 hour is the ideal time window with 1 KB packets for better performance.

Stevanovic et al. [100] proposed a technique to obtain the ground truth i.e. labeled traffic which is currently heavily dependent on the blacklist and the potential problems associated with using the lists. The technique investigates the mapping between the FQDN and Internet Protocol address. The technique is a semi-manual labeling approach which requires manually configuring the various thresholds. It uses automatic cluster analysis to label malicious/non-malicious traffic. The technique compares the approach with other labeling approaches and concludes that the outcome achieved by the system is, in fact, the ground truth.

Zhao et al. [101] presented a technique for botnet detection by extracting attributes at the TCP/UDP flow level to label the flow as malicious or benign using machine learning algorithm. While other techniques depend on bot communication through attack stage or setup stage, the proposed technique takes into consideration all the phases of the botnet lifecycle. Various novel attributes like average payload packet length, first packet size, the time between packets, packets exchanged, and the number of reconnects along with source Internet Protocol address and port, destination Internet Protocol address and port are used. Decision tree classifier, being a prevalent machine-learning algorithm, was used for detection. The system was developed in Java and uses the Weka machine learning framework. It showed improved results when compared with BotHunter. The system can be used for botnet detection in offline and real-time mode.

Discussion

Flow-based detection techniques employing IDS worked well for IRC based botnets but were limited to spamming botnets. Moreover, IDS rules could be easy evaded by using an encrypted communication channel. Detection technique presented by Prieto et al. [98] is time-consuming and have low detection rate. The research presented by Stevanovic et al. [99] indicates that the

length of the time window is a far more significant parameter in case of TCP than UDP. It further concluded that 1 hour is the ideal time window with 1 KB packet size for better performance. Further techniques which consider flow during all the stages of botnet lifecycle are bound to produce better detection rate than those techniques which considers only one stage. Table 2.3 presents a listing of key components of the Flow-based botnet detection techniques discussed in our review. Validation result for various techniques that employ flow-based detection is shown in Figure 2.6.

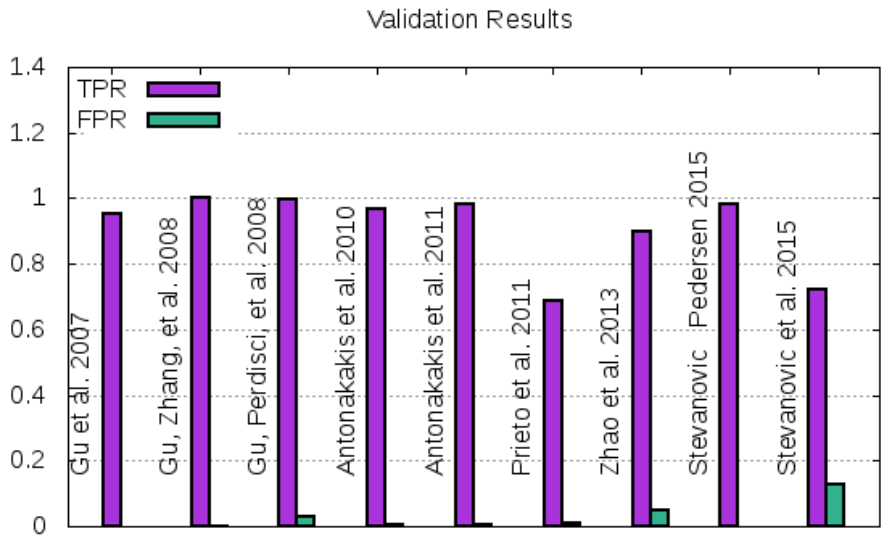


Figure 2.6 Validation results for Flow-based botnet detection techniques

2.8.2. Anomaly-based Detection

Anomaly-based detection techniques attempt to find anomalies in the various parameters or peculiar patterns in the traffic, query pattern, failed queries which are dissimilar from normal behavior. Systems infected with bot malware tend to have behavior dissimilar from normal systems which can be detected easily using anomaly detection techniques. Common parameters used in anomaly-based detection technique for botnet detection include low TTL value, Failed DNS queries, agile DNS-IP mappings, etc. The focus of anomaly-based detection technique is to detect anomalies arising out of botnet communication.

Table 2.4 Anomaly-based detection techniques

Reference	Technique	Whitelist/Blacklist	Targeted Botnet	Dataset	Pros and Cons
Villamarín-Salomón et al. 2008	Chebyshev's Inequality and Mahalanobis distance	-	DDNS based botnets	University of Pittsburgh CS Department Network traffic for 9 days w.e.f. 2/13/2007.	High false positives owing to the difficulty in distinguishing DDNS responses.
Jiang et al. 2010	Statistical graph decomposition /tNMF (tri Nonnegative Matrix Factorization)	MX Toolbox Blacklists	Conficker. A/B, Torpig, Dropper, Rustock.E & Kraken	3 Months Campus Network DNS Traffic from Jan 2009 to Mar 2009	Limited if the number of failed queries is small
Yadav & Reddy 2012	Temporal correlation/ Patterns in DNS Query of Bots	31 Trusted second-level domains including CDN, popular websites, blacklist vendors	Conficker	South Asia Tier-1 ISP dataset + 1-month campus DNS race	High detection rate for both pronounceable and non-pronounceable domains
Berger et al. 2016	Partial domain divergence using the Levenshtein ratio and Graph theory	Custom whitelist containing *.ntp.org, *.wordpress.com, *.apple.com	-	6 days dataset + 14 days dataset from ISP consisting of 100,000 systems	Real-time detection with small training time

State of the art

Villamarin-Salomon et al. [102] presented a DNS traffic based anomaly detection technique for botnet detection. Dynamic DNS is abused by bots to connect to the rendezvous point. The detection approach first looks for patterns in queries which are peculiarly high or concentrated over the short time frame. Finally, it checks the frequency of queries for domains which are no more active (NXDOMAIN). Chebyshev's inequality and Mahalanobis distance were used to find outliers. The research concluded that low TTL value is an essential attribute but not enough

attribute to detect an anomaly in DDNS traffic. The research also showed the difficulty in distinguishing DDNS names from other names in an enterprise network.

Jiang et al. [103] analyzed DNS failure graphs resulting in the identification of suspicious activities. The analysis classified Host-DNS interaction into three categories, namely, Host-star, DNS-star, and Bi-mesh. Host-star represents many DNS lookup by a few hosts. DNS-Star, in contrast, represents many hosts performing the same DNS lookup. Bi-mesh, on the other hand, represents those sets where a set of hosts queries a set of DNS lookups. DNS trace of campus network spanning 3-month period was used to classify various nefarious activities on the Internet like spamming, bots, etc. using tri-negative matrix factorization technique. Unlike the earlier technique, this anomaly detection technique focused on failure graph and even categorized the anomaly as spam, trojan, and bot. The study has a limitation in case if there are no or low failures in DNS responses.

Yadav et al. [104] developed a finding technique based on the analysis of various failed DNS queries i.e. Non-existent Domain (NXDOMAIN). Botmaster uses a flux mechanism to continuously change the domain name and IP of the C&C server to avoid evasion. While bots use algorithms to generate domain names, the botmaster fluxes the C&C Server to provide redundancy. Blacklisting as such is not an option since every time new domain names are being used. The key difference in domain name generated by an algorithm and real-world names is the relative frequency of the alphabets. A bot system employing an algorithm for domain name generation for communication with the C&C server is bound to produce more failed DNS responses as compared to the normal system. Three metrics namely Kullback-Leibler divergence, edit distance and Jaccard index were used to find the comparison between domains generated algorithmically and malicious/benign domains. Correlation matrix and entropy for the NXDOMAIN set were used to validate the proposed model. The proposed model has a high detection rate with very low false positive. However, botmaster may modify the DGA algorithm to circumvent the metrics used for comparison of distributions.

Berger et al. [105] presented a prototype system that can detect cybercrime by detecting an anomaly in agile DNS mappings in real-time. The proposed system maintains a mapping between FQDN and IP addresses and continuously monitors any changes in mapping. The

proposed system consists of four modules where valid DNS Query Requests Responses are processed first followed by checking for duplicates. It then prepares a mapping table and analysis the mapping using Graph Analysis module to find the suspicious mapping. This suspicious mapping is then analyzed for cybercrime detection. This system, however, fails to detect anomaly if the number of mappings is too small.

Discussion

Earlier, Low TTL value was a very strong anomaly indicator in Anomaly-based detection techniques. However, the use of Low TTL value for Load balancing in CDN leads to an increase in false positives. A few failed DNS queries resulting in NXDomain responses provided another anomaly indicator for detecting malicious activity. However, these techniques are limited if the number of failed queries is small. Three metrics namely Kullback-Leibler divergence, edit distance and Jaccard index introduced by Yadav et al. [104] for botnet detection have produced a high detection rate. Likewise, an anomaly in agile DNS mappings can be used for botnet detection.

Table 2.4 presents a listing of key components of the Anomaly-based botnet detection techniques discussed in our review.

2.8.3. Flux-based Detection

Flux-based detection techniques try to find IP flux in network traffic used for malicious activities wherein there is a continuous change in the IP Map associated with a domain and generally having very low TTL value. Dynamic DNS provides a way for updating a name server in real-time which is used extensively in Content Delivery Networks (CDN) for load balancing. Dynamic DNS is abused for Botnet communication to maintain high availability by rapidly changing IP addresses associated with a domain. Common parameters used in flux-based detection techniques include the number of queries, number of distinct resolved IP addresses, number of answered DNS response records, dissimilar Autonomous System Number (ASN), etc. The prime focus of flux-based detection technique is to detect IP flux being used for botnet communication.

State of the art

Choi et al. [106]–[108] developed a framework for Botnet detection using group analysis i.e. BotGAD (Botnet Group Analysis Detector) as shown in Figure 2.7. DNS traffic is mapped after filtering using a black and white list. A domain map is created for all IP's that request the domain. The map information is used by correlation domain extractor and matrix generator to calculate the similarity between the two domains. Group analysis works on the phenomenon that bots that are part of the same botnet tend to have similar group activities. These group behaviors can be analyzed using similarity measures. DNS features like count of domain labels, mean size of domain labels, number of queries directed, number of unique sender Internet Protocol addresses, and number of distinct resolved Internet Protocol addresses were used for clustering the data. The detection system can detect unknown botnets with a high degree of accuracy and is scalable for a large deployment. However, it cannot detect bots infected with a hard-coded IP address for C&C communication.

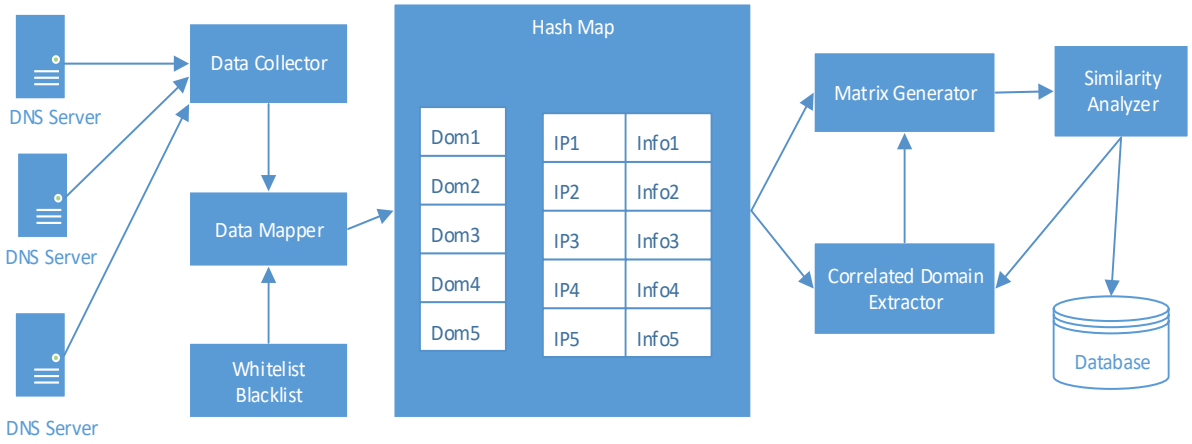


Figure 2.7 BotGAD framework

Perdisci et al. [109] proposed a novel method for the passive investigation of Domain Flux service Networks using a detection system as shown in Figure 2.8. Content Delivery Networks (CDN) makes use of DNS Flux service to offer a high degree of availability and load balancing. However, these flux services are abused for malicious purposes. Various DNS response parameters like TTL, the number of answered DNS Response records, cumulative set of answered DNS Response records over time, dispersed networks for the answered DNS

Response records were clustered using single linkage hierarchical algorithm. Various active and passive features can detect domain flux services without using any domain blacklists. The study concluded that IP growth ratio is the most significant feature which is obvious since flux networks have a rapid change in IP addresses.

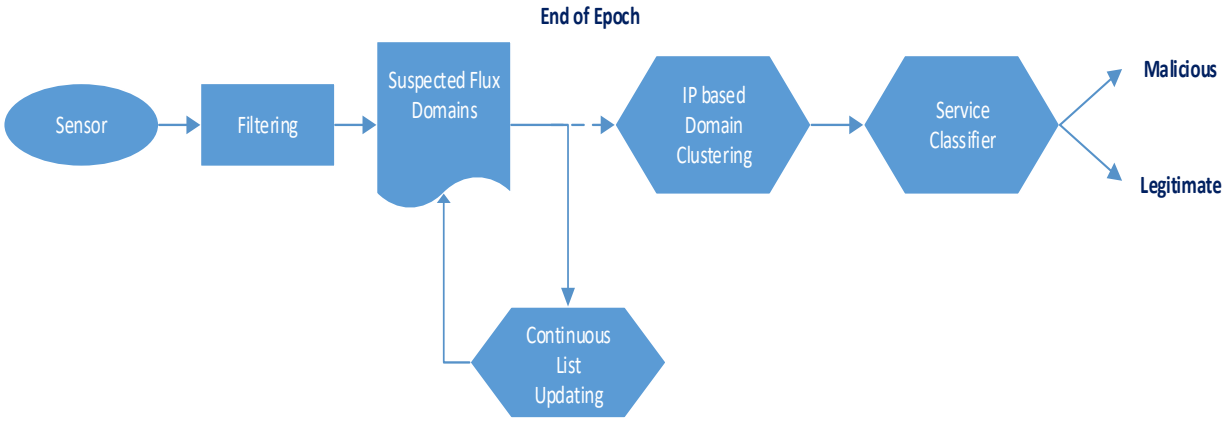


Figure 2.8 Detection system overview

Huang et al. [110] presented an instantaneous detection system for fast-flux service network named “Spatial Snapshot Fast-flux Detection (SSFD)”. While earlier techniques rely on temporal–based characteristics and tend to take a long time for detection, this technique works in real-time and can detect fast-flux in less than one second. The system consists of three components. The DNS packet monitor captures DNS traffic using wireshark. In, spatial snapshot feature extraction component, IP address is extracted from the answer and additional section, which is then mapped to physical spatial coordinates using an open project i.e. hostip.info. Time zone is used to calculate the information theory i.e. entropy to find out the unit of consistency in distribution. Spatial service distance is calculated as a measurement of the consumer-provider relationship. In the attack detection engine, Bayesian network classifier is used to detect Fast-flux service networks. The detection system can cause an error when presented with missing IP-Geo location value.

Table 2.5 Flux-based detection techniques

Reference	Technique	Whitelist/Blacklist Criteria	Targeted Botnet	Dataset	Pros and Cons
Choi et al. 2007	Correlated domain clustering and Similarity analysis	KISA blacklist, Sinkhole domain list, DNS-BH & Cyber –TA list	Non-P2P botnet	Traffic from /16 campus network, ISP DNS traffic 7/7/2009 & ISP DNS June 15, 2010	High accuracy. Ability to detect unknown botnets. Early detection.
Perdisci et al. 2009	Domain Clustering and Service classifier	Traffic Volume reduction filter using TTL and set of resolved IPs	-	Two large ISP networks with 2.5 billion queries/day for 45 days	Can be used in Spam detection applications
Huang et al. 2010	Spatial service distance and Bayesian network classifier	Top websites on Alexa, Blogs on Top, FFWeb, DNSBL, ATLAS & FluXoR	Fast Flux-based botnet	One public dataset Two collected dataset	Missing IP-Geo location can cause an error.
Chen et al. 2013	Webpage dynamicity and degree of periodic repeatability	Web of things, Free PC Security, McAfee Site Advisor, Blocklist & Removal Center	Fast Flux-based botnet	1000 most renowned Enterprises website Malicious websites from spam	Evaluation based on spam dataset and as such cannot be generalized to all botnets.

Chen et al. [111] presented a detection technique of fast-flux domains used by Web-based botnets. The idea behind detection is that regular web server provides active content while C&C web communication is mostly static. The connections on web pages are grouped for detail inspection. Web pages' attributes are checked for anomalous web communication by calculating web page dynamicity and degree of periodic repeatability. Dissimilar ASN, registration time and reverse lookup of DNS is used by the system to distinguish benign domain from malicious. Registration time of fast-flux domains (with an average of 18.5 days) is less as compared to normal domains. The study also detected that fast-flux domains tend to have more than one dissimilar ASN. Reverse lookup of domains was used to check the legitimacy of the domain.

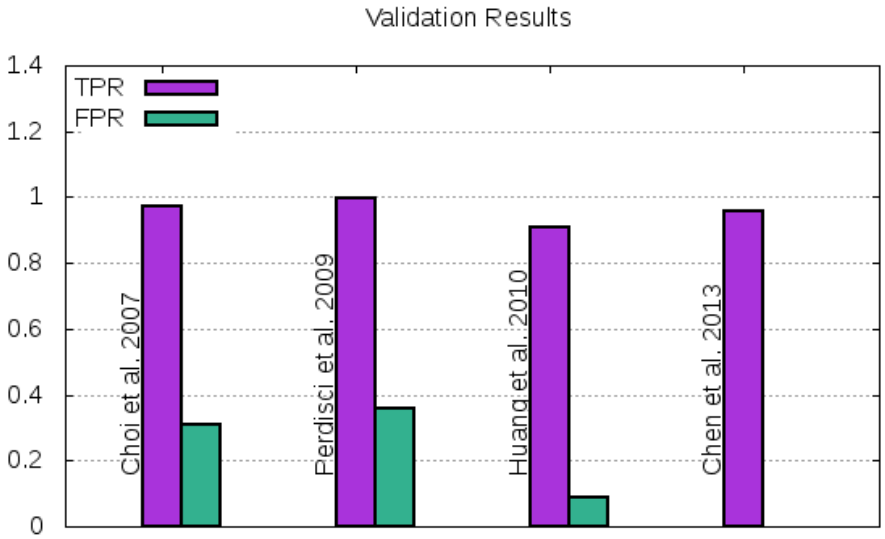


Figure 2.9 Validation results of Flux-based detection techniques

Discussion

Flux-based detection using group analysis presented by Choi et al. [106]–[108] provided an efficient mechanism as bots infected with similar malware tend to have similar group behavior. Group analysis has the potential to detect unknown botnets which generally goes undetected. However, if the number of infected systems in a network are too small, group analysis is not

very accurate. IP growth ratio turns out to be the most significant feature for detecting botnets employing a fast-flux technique for botnet detection. Likewise, IP-Geo location value is another significant feature for detecting fast-flux networks as IP address mapped with a flux domain are generally widely spread. Web page dynamicity is another important feature for detecting flux-based botnets as the registration time of fast-flux domains is generally less than normal domains. Table 2.5 presents a listing of key components of the Flux-based botnet detection techniques discussed in our review. Validation results for flux-based detection techniques are presented in Figure 2.9.

2.8.4. DGA-based Detection

DGA-based detection techniques attempt to differentiate domains queried in a network which are algorithmically generated (malignant) from the normal domains. Security agencies used domain blacklisting to flag domains used as C&C communication. To evade domain blacklisting techniques, Botmasters program the bot in such a manner to connect domains which are generated algorithmically and communicate for the smaller duration and discarded thereafter. Common parameters used for DGA-based detection include a number of Top-Level Domains (TLD), number of Second Level Domains (SLD), N-gram score, N-gram frequency, entropy, etc. The prime focus of DGA based detection techniques is to detect Algorithmically generated domains used for Botnet communication.

Table 2.6 DGA-based detection techniques

Reference	Technique	Whitelist/Blacklist Criteria	Targeted Botnet	Dataset	Pros and Cons
Antonakakis & Perdisci 2012	Alternating Decision tree learning Algorithm and Hidden Markov Model	-	Zeus V3, BankPatch, Bobax, Conficker, Murofet & Sinowal	15 months DNS traffic by a large North American ISP with 2M Hosts per day	High false positive and false negative. Dependent on NXDomain traffic

Zhou et al. 2013	Domain set active time distribution	Top 10K websites from Alexa	Conficker, Kraken, Torpig, Srizbi & Bobax	Pilot DNS server in China with 150K packets per second	Delay in detecting domains not present in backlists
Bilge et al. 2014	J48 & Genetic Algorithms	Zeus blacklist, Anubis, Wepawet, Phishtank & Alexa top 100 Global	Conficker, Kraken, Bobax & Srizbi bits	2.5 months of traffic consisting of 100 billion queries	High detection rate. Real-time deployment.
Bottazzi & Italiano 2015	Knowledgebase Construction and Clustering on length and number of hits	-	-	One month's traffic of a Corporate Network having 60K hosts	Easy evasion by varying length of the domain.
Sharifnya & Abadi 2015	Suspicious group Analysis Detector, Kullback-Leibler Divergence, Spearman's rank correlation coefficient	Whitelist consisting of Alexa top 100. Blacklist from Murofet	Conficker. C	Dataset of benign and malicious DNS queries	Low false alarm rate. Considers the history of hosts suspicious DNS activities.
Nguyen et al. 2015	Collaborative filtering, Density-based clustering & Cosine Similarity	Alexa 1 million domain whitelist	Necurs, Vawtrak & Palevo	Two weeks of DNS traffic log of about 18000 systems	Cannot detect P2P botnet

Erquiaga et al. 2016	Markov chains detection algorithm	-	DGA Malware	A dataset from Malware Capture facility Project [112]	The high rate of true negative values
Tong & Nguyen 2016	Modified Mahalanobis distance and K-means	-	Conficker, Tinba, Bebloh, Tovar-Goz & Kraken	Top 1 million websites from Alexa [113] and DNS-BH Malware Domain Blocklist	Dependent on botnet family.
Kwon et al. 2016	Signal processing technique focusing on the simultaneous and periodic behavior pattern	Less than 13 queries in one hour	-	20 DNS traffic from Malware dumps and real-world DNS Servers	Scalable
Wang et al. 2017	Numerical Analysis of Request time and request count distribution and Chinese Whispers algorithm	Spamhaus, BRBL, SpamCop, and AHBL	Kraken, Conficker, Cycbot & Murofet	Traffic from 10000 Users of Educatio n Network of Tainan City (May 2013 to June 2015)	Low false alarm rate and online detection

State of the arts

Antonakakis et al. [114] presented a technique for DGA based botnet detection named the Pleiades. Pleiades inspects DNS queries that result in Non-Existent Domains. The system

consists of two main components: the DGA Discovery component, DGA Classification and C&C detection component as shown in Figure 2.10. In DGA Discovery, all NXDomains are clustered based on the statistical similarity e.g. length, frequency, etc. The idea was to discover domain clusters that belong to the same DGA based botnet. In DGA classification and C&C detection, two models are used. Statistical multi-class classifier model was used to label the cluster e.g. DGA-Conficker-A. Hidden Markov model is used for detecting candidate C&C domains by finding single queried domain by a given host in the cluster. One of the prime limitations of this technique was the consideration of the domain as a single character sequence. The study also explained the limitation in providing the exact count of infected hosts.

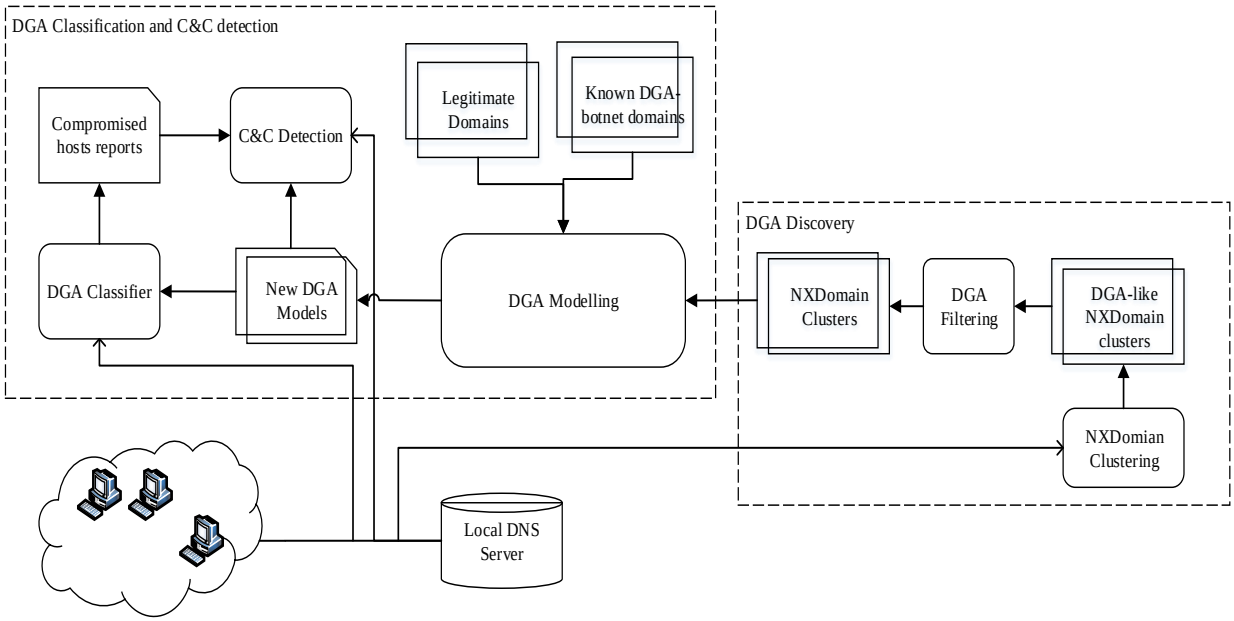


Figure 2.10 Overview of Pleiades

Zhou et al. [115] presented a system for DGA based botnet detection using DNS traffic analysis. The system consists of two modules. In the pre-handle module, a whitelist consisting of 10k Alexa domains is applied to the captured traffic to significantly reduce benign traffic. In the DGA detection module, remaining domains are clustered based on the similar live time span and similar visit pattern. The main idea behind the detection system is that the visit pattern and live time span of domain generated by DGA are different from normal domains which have

large live time span and dissimilar visit pattern. Unconfirmed domains need longer time for further investigation and as such the system is not effective for real-time detection.

Bilge et al. [116] presented a system for spotting malicious domain named EXPOSURE. Four sets of features namely “Time based, DNS answer based, TTL value-based and Domain name-based” are collected as part of feature attribution phase. “Change-Point Detection Algorithm” and “Similar Daily Detection” algorithm is used to classify the domain into malicious or benign. J48 decision tree algorithm was used in the training phase of the classifier. The detection scheme reported a very high detection rate of 99.5% with 0.999 as Area under the curve (AUC) and a low false-positive rate of 0.3%. EXPOSURE can be evaded if the TTL value is set to a larger value or by decreasing the number of DNS queries.

Bottazzi et al. [117] presented a data mining approach for detection of algorithmically generated domains. Proxy logs from a large Italian organization consisting of 60,000 workstations and 100,000 users were mined for one month. The approach consists of extracting the Second-Level Domain (SLD) from the logs and constructing a knowledge base. For each day, a list of SLD is identified which consists of never seen domains and non-RF-1035 compliant domains. For each day, lexical analysis is done on the SLDs to find the amount of vowel, the number of consonants and amount of numbers. Finally, clustering is done on the length of SLD and the number of numbers. Results show that 5 out of the top 8 SLDs cluster indicate the domains are algorithmically generated. Pronounceable domains generated using DGA, however, could not be detected using this technique.

Sharifnya et al. [118] presented a botnet detection technique based on the distinction between the domain names generated algorithmically or randomly and between legitimate ones. To detect botnet, a negative reputation system is used. The proposed model is different from other models as it associates a negative score with each host in the network. This score is further used to classify bot according to J48 Decision trees. Initial filtering based on the whitelist is used to separate trusted domains. Domain Group Analysis is done to find hosts which query the same domain. Domain labels are then analyzed if they are algorithmically generated using the correlation coefficient. A score is finally calculated between 0 and 1 indicating whether the host

is involved in bot activities or not. Unlike other techniques, it considers the history of malicious domain activity for each host in the network.

Nguyen et al. [119] developed a method for detecting botnet employing Domain Generation Algorithm using collaborative filtering and density-based clustering. A whitelist filtering is used for domain filtering followed by K-means clustering with K=2 on 2-gram frequency value used as an input. Finally, "Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm" is used to find a bot that corresponds to the same botnet. One of the prime limitations of the technique is the inability to detect peer to peer botnets.

Erquiaga et al. [120] presented a technique based on behavioral analysis to detect DGA malware traffic. In the technique, flow is represented as a four-tuple: source and destination Internet Protocol address, port and the protocol. Each flow is aggregated to form a connection. Behavior model is then applied to each connection using well-defined steps. Detection algorithm consists of three phases in which for each connection a Markov chain model is applied which results in a transition matrix and initialization vector. In the second phase of detection, the traffic to be evaluated is connected into a series of letters. In the final stage, the resultant string is evaluated against all detection models. An alert is generated if the probability of the behavioral model exceeds a certain threshold. The research has distinctively labeled DNS traffic into five groups: normal traffic, Non-DGA traffic (used for spam), DGA type-1, DGA type-2 and fast flux. The study concluded that more realistic results can be obtained if the dataset consists of both normal and botnet traffic.

Tong et al. [121] presented a technique for DGA based botnet detection using semantic and cluster analysis. The proposed system comprises three stages: Domain filtering, DGA filtering, and DGA Clustering. In domain filtering, top 1 million Alexa domains are used as a whitelist. Semantic features like N-gram score, N-gram frequency, entropy, and meaningful character ratio are used to filter out domains. In DGA filtering, correlation matrix and Mahalanobis distance of the domains filtered in the first phase is calculated to filter benign domains. In DGA clustering, K –means is used to cluster the domains filtered in the second stage to different groups such that domains generated by the same domain generation algorithm fall in the same

group. The technique considers only linguistic features and as such is bound to be less accurate. The study concluded that improvement can be made by using IP and domain-based features.

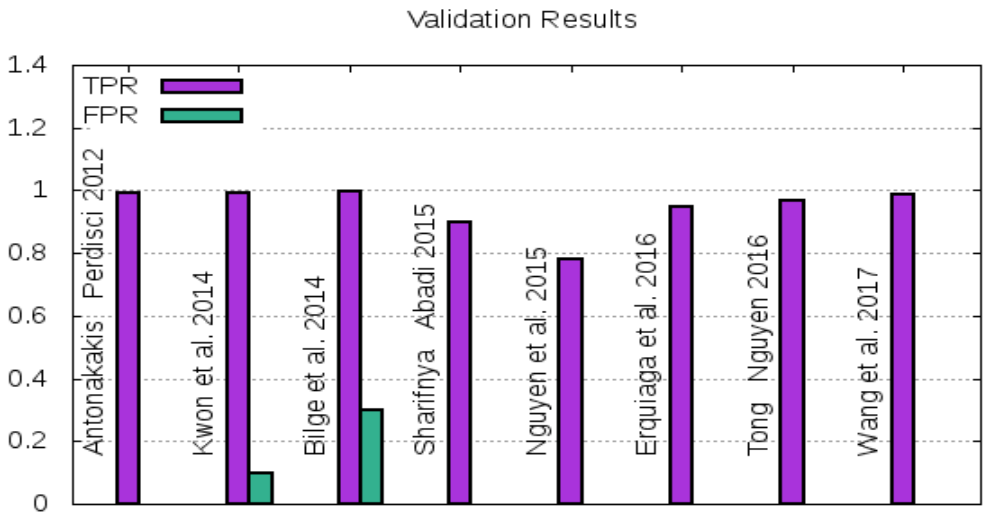


Figure 2.11 Validation results of DGA based botnet detection techniques

Kwon et al. [122] presented a scalable system named PsyBoG for botnet detection from a big DNS traffic. Due to increased Internet penetration, the volume of network traffic is increasing on a regular basis, thereby making real-time analysis difficult and as such, there is a need for a scalable system for botnet detection. PsyBoG is a scalable system designed while keeping in mind the increasing volume of traffic. Periodic behavior pattern along with simultaneous behavior pattern is evaluated using DNS traffic to find botnet group. Power spectral density, a signal processing technique is used to find periodic behavior patterns. Power distance is used to find out simultaneous behavior patterns. The system is designed for high accuracy, robustness applicability apart from high scalability. PsyBoG can be evaded by using randomized and slow query pattern.

Wang et al. [123] developed a technique for detection of DGA based botnets named DBod. The technique works on the analysis of failed DNS Requests and creates a cluster of infected and clean machines in a network. The technique was evaluated in an academic environment for 26 months and had shown effective detection results. A score function was introduced which assigns a unique score to a cluster which classifies them as malicious or benign. DBod is unable

to detect dormant bots. Increasing the time epoch leads to the greater possibility of detection but leads to higher computation. The study recommends 1 hour as an acceptable time for the proposed technique. Table 2.6 presents a listing of key components of the DGA-based botnet detection techniques discussed in our review. Validation results of various DGA- based botnet detection techniques are presented in Figure 2.11.

Discussion

With the invent of DGA for botnet communication, the focus of the detection techniques shifted from IP map to domain name. New domain features like the length of the domain name, number of domain tokens, Distinct TLDs, etc. were explored and used for detecting DGA based botnets. Whitelist comprising top domains from Alexa was used to filter domains generated by algorithms to normal domains. Semantic features like N-gram score, N-gram frequency, entropy, and meaningful character ratio were used to filter out benign domains from malignant domains. However, DGA based detection techniques can be evaded if the length of domain token is random and if there are delay and randomness in the query pattern.

2.8.5. Bot Infection Detection

Bot infection detection techniques attempt to find bot-infected hosts in a network. While earlier techniques aim to detect C&C infrastructure, there is a growing demand to detect bot-infected machines in a network. Enterprises cannot wait for the shutdown of a botnet which is a time-consuming activity and requires a great deal of international cooperation. Common parameters used for bot infection detection include a number of FQDN, IP addresses, FQDN blacklist presence, IP blacklist presence, etc. The focus of Bot infection detection techniques is to detect bot infection in a network i.e. to list the infected hosts in a network.

State of the art

Tu et al. [124] presented detection technique for Bot-infected machines using the comparable sporadic DNS queries. An analysis of time-interval series correlation of the DNS queries was established to find similarity between the same botnet. Moreover, domains flux infect systems are bound to produce many failed DNS queries. The proposed model was evaluated against five

distinct botnet samples to gauge the success of the model. One of the main limitations of this technique is the small duration of the dataset and it considers only the time interval of the queries for analysis.

Stevanovic et al. [125] presented a technique for identification of infected clients using DNS traffic investigation which is an improvement over the technique presented in [124] which considers only time interval. Apart from checking domain-flux and fast-flux, the method also finds the compromised clients. The system contains four phases as depicted in Figure 2.12. In the first phase, DNSMap is used to find agile FQDN to IP mappings by using bipartite graphs. In the second phase, various features from the graph, FQDN, IP, FQDN blacklist and IP blacklist are extracted and used for further processing. In the third phase, a supervised machine learning algorithm (Random Forests classifier) is used which classifies benign and malicious domains. In the client analysis phase, the output of the machine learning algorithm is used to find out compromised clients.

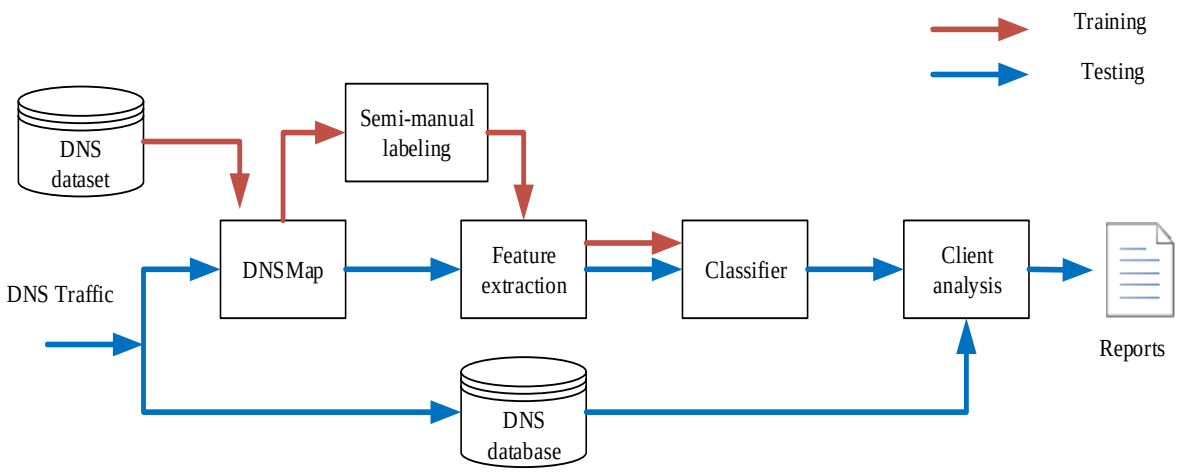


Figure 2.12 Overview of bot infection detection system [125]

Discussion

Bot infection detection is still new and in the early stages. DGA based detection techniques are extended to detect bot-infected machines. Features like FQDN, IP, FQDN blacklist and IP blacklist are used for detecting bot infection. One of the challenges in this technique is the unavailability of the labeled dataset and the challenge in obtaining the ground truth. Another

challenge is in comparing a technique with existing one as most of the technique is tested on different datasets and are not easily available. Table 2.7 presents a listing of key components of the Bot-infected machine-based botnet detection techniques discussed in our review.

Table 2.7 Bot Infection detection techniques

Reference	Technique	Whitelist/Blacklist Criteria	Targeted Botnet	Dataset	Pros and Cons
Tu et al. 2015	Autocorrelation and Similarity detection	-	Conficker, Zeus, Bobax, Murofet & Kraken	24 Hours Traffic from Virtual Environment	Small dataset duration. Domain's lexical feature not considered
Stevanovic et al. 2016	Detecting Suspicious Agile FQDN-IP mappings and Random Forests classifier	Whitelisting based on DNSMap analysis	Generic	Multiple real-world traffic from the National mobile network and a local fiber ISP	A low true positive rate

2.9. Discussion

The purpose of this research is to filter botnet detection techniques into various categories to get an understanding of the DNS features being misused by botmasters to evade detection and the possible ways to detect such misuses by closely monitoring these attributes. The discussion starts with the relevance of dataset location, followed by essential attributes of a proposed DNS-based detection system and finally compares DNS features used by various techniques.

2.9.1. Dataset Location

It was observed that dataset from enterprise networks (Campus, Department, etc.) used in various detection systems tends to have a smaller number of live hosts (<10k) while dataset from ISP has a higher number of active hosts. Several active hosts in a network play a significant role as it directly impacts the volume of the DNS traffic. Besides this, identifying

compromised clients is not possible at the ISP level if the systems are behind the NAT. Likewise, Group analysis is difficult at an enterprise-level if only a few systems are infected with DGA-botnet. Detection technique thus needs to be taken into consideration in both cases.

2.9.2. Essential attributes of a Smart DNS-based Botnet Detection System

Based on the research of the existing detection techniques, the following essential attributes of a Smart DNS-based detection system are proposed.

- Real-time detection: One of the key requirements is the ability of the system to quickly detect botnet communication. Time is an essential commodity especially in this area as average up-time for a domain generated by DGA-based botnets is very less. Moreover, fluxing techniques provide a way to flux the IP's in a short time as small as a few seconds. Use of pipeline structure in the detection system is a must to provide real-time detection.

- Versatile: Versatility is the ability of the detection system to detect unknown or new botnets employing new techniques for bot communication. DNS protocol is being abused in a variety of ways. A DNS-based botnet detection system should be versatile in the sense that it should be able to detect domain-flux, IP-flux, and DGA-botnets.

- Scalable: The growing volume of Internet traffic is making it difficult to perform deep packet inspection and similar techniques which require considerable time and memory. A Smart DNS–based detection system should be designed while keeping scalability in mind. High performance under heavy network traffic is expected from a Smart DNS-based detection system.
- Notifications: Instant alert generation along with the alert-level is vital to the success of a Smart DNS-based detection system. Notification can simplify the task of network administrators in curbing the menace of a botnet.

- Blacklist and whitelist independence: A Smart DNS-based detection system should not be dependent on any blacklist or whitelist. The research has shown that even the top domains on the Internet can be compromised. Moreover, Social networking sites like Twitter are recently being used for C&C communication. Using a whitelist based on the popularity of domains can be a catastrophic choice. A smart DNS-based detection system should not rely on any blacklist or whitelist.

- High detection rate: One of the prime criteria of any detection system is the detection rate. The goal is to ensure that nothing remains undetected be it a C&C server or a bot-infected device.

- Low false positive: Another essential attribute is the low false alarm rate. Lower the rate greater is the confidence in the detection system and wider acceptance.

- No easy evasion: The detection technique should not be such that it can be easily evaded by any minor upgrade. The cost involved should be so high that considerable effort and time be needed to evade the detection system.

2.9.3. Comparison of DNS Features

DNS Protocol has been widely used in Botnet detection techniques. Initial research focused on basic attributes like IP address, TTL, blacklist presence, etc. However recently various derived features like City and country of the IP address, Average number of answers, etc. are being used.

Table 2.8 provides a comparison of various DNS features used by selected techniques. The earlier techniques started with the basic attributes of the IP address. Flow-based attributed were subsequently added to detect botnet in traffic parameters. DNS Request and response features were added as the evasion techniques evolved over the years. Flux techniques used for botnet

communication lead to botnet detection based on FQDN-IP mappings. Use of Domain generation algorithms gave rise to botnet detection based on domain-based features.

2.9.4. Open Issues and Challenges

A botnet is the most serious threat on the Internet. Despite many collaborative attempts by Governments and law enforcement agencies, Botnet still exists and grab top headline all around the world. A botnet is an open challenge for security researchers all around the world. While actively monitoring DNS traffic, can identify the existence of Botnet in a network, a two-level approach is needed to combat the Botnet phenomenon.

C&C detection and removal

At the server-side, it is important to find all the C&C servers and block all such domains to cripple one end of the communication. Numerous attempts (like domain blacklisting, law enforcement crackdown and arrests) have been made to curb this menace but with limited success. New domains are constantly added as C&C servers for a shorter period and discarded thereafter. The existence of vulnerabilities in normal domains also gives the botmaster an alternative to use an existing domain instead of a new domain. Use of social networking sites as a C&C server is further complicating the removal process.

Bot infection detection and removal

On one hand, efforts are needed to restrict all domains used as Command and Control Server. On the other hand, it is important to curb the demand i.e. identifying and cleaning bot-infected machines. Vulnerable systems on the Internet are continuously added to Botnet using zero-day exploits [126], [127], email attachments, etc. Use of IoT devices as bots has recently been the security problem of many enterprises. Level of stealthiest of some bots is so high that sometimes it remains undetected for years [128].

Table 2.8 Comparison of DNS features used

Reference	DNS request-based features	DNS response-based features	Domain-based features	IP based features	Blacklist-based features	Flow-based features	FQDN IP mapping features
Ramachandran et al. 2005				✓			
Strayer et al. 2006						✓	
Gu, Zhang, et al. 2008				✓			
Gu, Perdisci, et al. 2008						✓	
Gu et al. 2007			✓	✓			
Antonakakis et al. 2010			✓	✓			✓
Antonakakis et al. 2011	✓	✓	✓	✓	✓		✓
Stevanovic & Pedersen 2015	✓	✓	✓	✓			
Stevanovic et al. 2015		✓	✓	✓	✓		
Prieto et al. 2011		✓	✓				
Zhao et al. 2013				✓		✓	
Villamarín-Salomón et al. 2008		✓					
Jiang et al. 2010		✓	✓	✓			
Yadav & Reddy 2012		✓	✓	✓			

Berger et al. 2016			✓	✓			✓
Choi et al. 2012	✓	✓	✓	✓	✓		
Perdisci et al. 2009		✓	✓	✓			
Huang et al. 2010		✓					
Chen et al. 2013		✓	✓	✓			
Sharifnya & Abadi 2015			✓				✓
Nguyen et al. 2015	✓		✓				
Bilge et al. 2014		✓	✓	✓			
Wang et al. 2017	✓	✓	✓				
Antonakakis & Perdisci 2012			✓				
Erquiaga et al. 2016						✓	
Bottazzi & Italiano 2015		✓	✓				
Tong & Nguyen 2016		✓	✓				
Zhou et al. 2013	✓		✓				
Kwon et al. 2014	✓		✓				
Tu et al. 2015	✓						
Stevanovic et al. 2016		✓	✓	✓	✓		✓

Various open issues and challenges for effective botnet detection are described below:

Problem of comparison

Due to the unavailability of proper dataset and implementation description, comparison of botnet detection methodology is not easy. Garcia et al. [41] presented a comparison of various botnet detection methods. Statistical methods of error detection are not good enough from a botnet detection point of view. Various new error metrics dealing with IP addresses instead of net flows were introduced. Further, time frames were added to the error metrics to make them more relevant. The research concluded that a common platform for comparison of the different methodology can significantly improve results.

Evolving botnets

Botnets are continuously evolving [30], [129], [130] and adding stealth capabilities. While blacklisting can significantly help in curbing the communication for the known botnet, it generally doesn't work for unknown botnets or new botnets. Moreover, the use of a domain generating algorithm for botnet communication thwarts all blacklisting attempts. Although significant research is being done on detecting DGA based botnets, yet botnet employing DGA continues to flourish.

Network Capture location: Enterprise vs ISP

A dataset from enterprise networks (Campus, Department, etc.) tends to have a smaller number of live hosts (<10k) while dataset from ISP has a higher number of active hosts. Identifying compromised clients is not possible at the ISP level if the systems are behind the NAT. Likewise, Group analysis is difficult at an enterprise-level if only a few systems are infected with DGA-botnet.

DHCP churn

Another important issue in effective DNS based botnet detection is the DHCP churn [131]. This is particularly applicable to wireless devices where there is a lot of entry and exit per minute. Different devices may obtain the same IP address once the lease assigned to system expires or the system disconnects the wireless network. Tracking individual device is difficult especially when DHCP logs are unavailable. Bot infection detection for wireless devices remains a challenge in such a scenario.

Use of Social Network as C&C

Curbing the use of social networking sites (e.g. Twitter [37] and Instagram [53]) and development platforms (e.g. Github [54]) as a C&C server is an open challenge since this platform cannot be blocked. Detecting C&C servers encapsulated in encrypted traffic hosted on top websites remains an open issue.

2.10. Summary

This chapter presented the related work done in the area of botnet detection using the DNS protocol. Botnet detection techniques are classified into Flow-based, Anomaly-based, Flux-based, DGA-based and Bot infection detection-based techniques. Flow-based detection techniques attempt to classify the network flow into malicious and benign based on various parameters inspected in the network flow. Anomaly-based detection techniques attempt to find anomalies in the various parameters which are dissimilar from normal behavior. Flux-based detection techniques try to find IP flux in network traffic used for malicious activities. DGA-based detection techniques attempt to detect domains queried in a network which are algorithmically generated. Bot infection detection techniques attempt to find bot-infected hosts in a network. After careful assessment of the related work, essential attributes of a smart DNS based botnet detection system are proposed.

3. Bot DNS Anomaly Detector (BotDAD)

3.1. Introduction

This chapter presents a novel technique named BotDAD: Bot DNS Anomaly Detector for detecting bot-infected machines in a network using DNS fingerprinting which considers the complete DNS activity of a host per hour. It starts with the architecture of BotDAD followed by the detailed description of individual components, DNS features used, and technique used for anomaly detection.

Section 3.2 explains the overall architecture of BotDAD. Section 3.3 explains the DNS fingerprinting module consisting of DNS features extractor, Host profiler, and Fingerprint generator. Section 3.4 explains all the features (P1-P15) used in DNS fingerprinting for an active host in the network on an hourly basis. Section 3.5 covers Anomaly detection engine. Section 3.6 briefly explains the classifier used in the machine learning module.

3.2. BotDAD Architecture

Figure 3.1 provides the architecture of BotDAD. It has two main subsystems: DNS Fingerprinting module and Anomaly detection engine. DNS Fingerprint generation module generates hourly DNS fingerprint of each host in the network by inspecting network DNS traffic. It consists of three sub-modules: Feature Extractor, Host Profiler, and Fingerprint Generator. DNS feature extractor module processes the network capture file to extract DNS request and response parameters that are useful for fingerprint generation. Host profiler module parses the output of the DNS extraction module i.e. request and response parameters and builds in-memory host profile. DNS fingerprint generator module parses the in-memory host profile and creates a DNS fingerprint for each host in the network. This fingerprint is then analyzed by anomaly detection engine which classifies it into two categories i.e. bot or clean based on the

presence or absence of an anomaly. The output of the Anomaly Detector is then used to train the classifier and create a trained model. The trained model is then used to predict the outcome which saves time and make the next analysis better. The use of machine learning enables us to make quick detection for future DNS fingerprints with high accuracy.

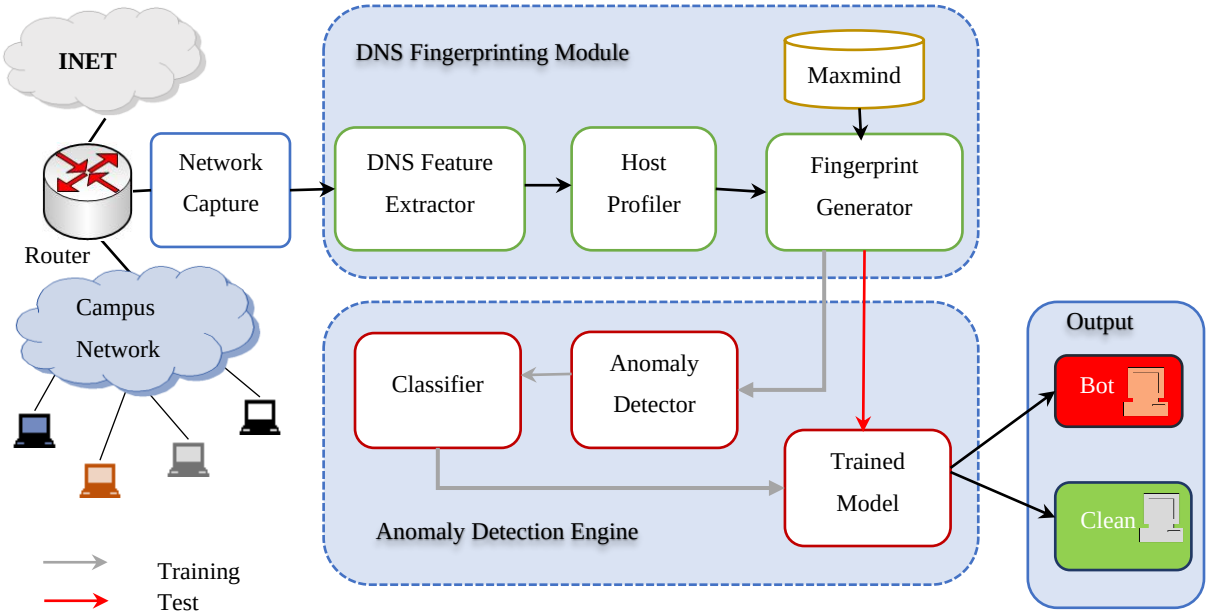


Figure 3.1 BotDAD architecture

3.3. DNS Fingerprinting Module

DNS Fingerprint generation module generates a DNS fingerprint of each host in the network. To generate the fingerprint, hourly network capture file is parsed for various features and host profile is generated as per schema shown in Figure 3.2. Maxmind database [60] is used to get additional details about resolved IP addresses like city and country. Complete parameters used in DNS fingerprint is shown in Table 3.1. It consists of three modules: Feature Extractor, Host Profiler, and Fingerprint Generator.

3.3.1. Feature Extractor

DNS feature extractor module processes the network capture file to extract various DNS request and response parameters that are useful for fingerprint generation. The request parameters extracted in this module include the transaction number, Host IP, FQDN, number of domain

tokens, request type (A, CNAME, PTR, AAA, etc.), length of FQDN, request timestamp and the DNS server IP. Response parameters extracted in this module include a transaction number, Host IP, FQDN, query type, answer code, TTL, resolved IP and response timestamp. Algorithm to extract feature is shown in Algorithm 3.1.

```
Algorithm ExtractFeature
Input: pcapfile
Output: request.csv, response.csv
1:  For each packet in pcapfile
2:      If QR Flag == request
3:          Save (txn_id, host_ip, domain, query_type, server_IP, timestamp)
            to request.csv
4:      Else
5:          Save (txn_id, host_ip, res_code, TTL, resolved_IP, timestamp)
            to response.csv
6:      End if
7:  End For
```

Algorithm 3.1 Extract Feature

The DNS Feature extractor module starts by selecting the first packet in the pcap file. If the query/response bit is set to 0, the module extracts transaction id, host IP, FQDN, query type, server IP and request timestamp from the request and append it to request output file. If the query/response field is set to 1, the module extracts transaction id, host IP, response code, resolved IP address and response timestamp from the response and append it to the response output file. Only DNS packets are processed by this module. All other packets are ignored by this module. The sequence is repeated for remaining packets in the capture file.

3.3.2. Host Profiler

Host profiler module parses the output of the DNS extraction module i.e. request and response parameters. It builds a DNS profile for all hosts in the network by grouping the queries from each host in a schema as shown in Algorithm 3.2. DNS queries from each host are further grouped according to the FQDN. To accommodate multiple queries for a single FQDN by a host, a list is used. DNS query reply message generally contains more than one response records. Therefore, a list is maintained to handle multiple response records for a given DNS

query request. At the end of the processing, an in-memory profile of all the hosts who used DNS service is constructed.

```
Algorithm BuildHostProfile
Input: request.csv, response.csv
Output: In-memory Host Profile
1:  For each entry in request.csv
2:      If Host_IP doesn't exist in Hosts dictionary
3:          Create new Host dictionary with key equal to Host_IP
4:      End if
5:      If the domain doesn't exist in Hosts Queried Domain dictionary
6:          Create new domain dictionary with key equal to the domain
7:      End if
8:      Add a new entry in query dictionary with key equal to txn_id
9:      Add query type, server_IP, and timestamp in the request object
10: End For
11: For each entry in response.csv
12:     If (Host_IP doesn't exist in Host's dictionary) Or
            (domain doesn't exist in Hosts Queried Domain dictionary)
13:         Or (txn_id doesn't exist in query_list Domain)
14:         Ignore entry as the request was not part of the pcap file
15:     End if
16:     Add res_code, TTL, resolved_IP, and timestamp in the response object
17: End For
```

Algorithm 3.2 Build Host Profile

For each entry in the request output file, the host profiler checks for the host entry in hosts dictionary. In the absence of an entry, a new entry is created with key equal to the host IP address. If such an entry already exists, the FQDN is checked in hosts queried domain dictionary. Again, in the absence of an entry, a new entry is created in Hosts queried domain dictionary with key equal to FQDN. Afterward, a new item is inserted in the query list using the transaction id. Query type, server IP and request timestamp are added in the request structure. The steps are repeated until all the records in the request's file are processed.

In order to add the response fields, each entry in the response field is processed. Like the sequence above, host IP and FQDN are checked in the hosts and hosts queried domain dictionary. The response is ignored in the absence of key as the request was not part of the capture period. Finally, the response fields are updated in the host schema by selecting item using the transaction id as a key. The steps are repeated until all the records in the response file

are processed. On the completion of the processing of request and response files, an in-memory representation of the host schema is established.

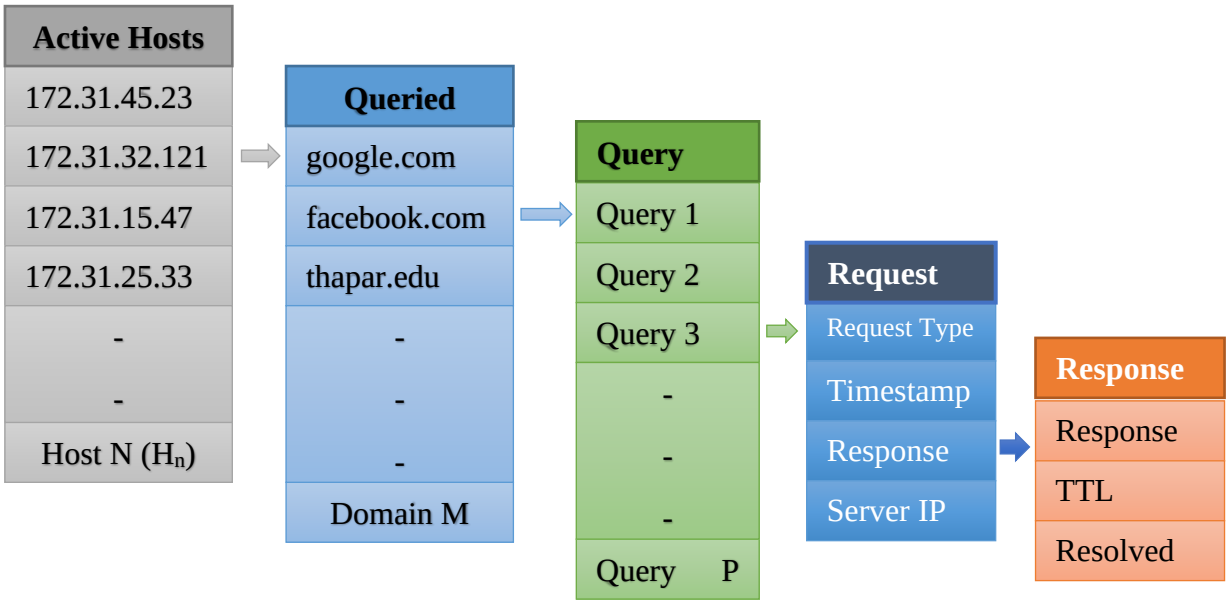


Figure 3.2 Host profiling schema

3.3.3. Fingerprint Generator

DNS fingerprint generator module parses the in-memory host profile and creates a DNS fingerprint for each host in the network as shown in Algorithm 3.3. For each host in the host's dictionary, all the parameters (P1-P15) are initialized to the default value. Some parameters like "number of distinct requests" can be calculated directly by using the length function of the hosts queried domain dictionary. Other parameters like "number of distinct cities" and "number of distinct countries" of the resolved IP address requires creating of a temporary list. Each new entry is checked for presence first and added if not in the list. Finally, in the end, the length of the list gives the parameter value. Parameters like "uniqueness ration" and "flux ratio" are calculated by using parameters calculated earlier and as such are calculated in the end. The values (P1-P15) i.e. DNS fingerprint thus obtained are saved in the output file. The steps are repeated until all the hosts in the host dictionary are processed.

```
Algorithm GenerateFingerprint
Input: In-core host profile, Maxmind database
Output: DNS_fingerprint.csv (P1-P15)
1:  Set P1 to P15 = 0
2:  For each host in Hosts dictionary
3:    P2 = Length (Hosts Queried Domain dictionary) / * P2=Number of  distinct requests */
4:   For each Domain in Hosts Queried Domain dictionary
5:       If P3 < Len(query_list)    /* P3=Highest number of request for single domain */
6:          P3 = Len(query_list)
7:       For each Query in query_list
8:          Increment P1                                   /* P1= Number of requests */
9:        If (reqType ==MX)
10:          Increment P6                                /* P6= Number of MX requests */
11:       If (reqType==PTR)
12:          Increment P7                                /* P7= Number of PTR requests */
13:      If (DNS Server doesn't exist in queried server list)
14:          Add DNS Server IP to the queried server list
15:      If (TLD not queried earlier)
16:          Add TLD to queried TLD list
17:      If (SLD not queried earlier)
18:          Add SLD to queried SLD list
19:      If (response code == Failed or NXDomain)
20:          Increment P12                              /* P12= Number of Failed Queries */
21:         Convert resolved IP address to City and Country using Maxmind database
22:         If (IP doesn't exist in resolved IP list)
23:            Add IP address to resolved IP list
24:         If (city code doesn't exist in resolved city list)
25:            Add City code to the resolved city list
26:         If (city country doesn't exist in the resolved country list)
27:            Add country code to resolved country list
28:        Increment request count using key as timestamp(hh:mm)
29:    End For
30:  End For
31:   P4 = P1/ (Host active duration in minutes)       / * P4=Average number of requests */
32:   P5 = greatest count from all timestamps        / * P5=Highest request per minute */
33:   P8 = Length (queried server list)                  / * P8=DNS servers queried */
34:   P9 = Length (queried TLD list)                   / * P9= Distinct TLD queried  */
35:   P10 = Length (queried SLD list)                  / * P10= Distinct SLD queried */
36:   P11 = P1/P2                                          / * P11= Uniqueness ratio */
37:   P13 = Length (resolved city list)                      / * P13= Distinct cities */
38:   P14 = Length (resolved country list)                 / * P14= Distinct countries */
39:   P15 = P2/ Length (resolved IP list)                  /* P 15  = Flux ratio */
40:   Add P1 - P15 to DNS_fingerprint.csv
41: End For
```

Algorithm 3.3 Generate DNS Fingerprint

3.4. DNS Features

Various features used for DNS fingerprinting (P1-P15) can be broadly classified into the request, response, domain, IP and mapping type features.

3.4.1. Request-based Features

- *The number of DNS requests per hour* (P1): is the count of DNS requests sent by a host in an hour. It provides a very early indication as to whether a host is infected or not. Bot-infected machines tend to have higher requests per hour than normal hosts.
- *The number of distinct DNS requests per hour* (P2): is the count of different domains queried by a host in an hour. It is a very important parameter in detecting DGA based botnets. Hosts infected with DGA malware tend to have a higher number of distinct requests than normal hosts.
- *The highest number of requests for a single domain* (P3): is the number of times a domain queried the maximum by the host. It helps to detect DNS tunneling wherein sensitive information is encoded in DNS queries and responses.
- *An average number of requests per minute (P4*): is obtained by dividing the total number of DNS requests sent by the active period of a host in minutes. It is useful in detecting machines infected with malware, which are not using short bursts of DNS requests, instead, are regularly contributing to DNS requests using sleep interval. It is calculated by dividing the number of requests sent by a host with the duration of time the host was active and using the DNS service.
- *The highest number of requests per minute* (P5): is the highest value obtained when the count of DNS requests is arranged minute wise for a given host. It helps detect bots infected with malware which are using a short burst of DNS requests to communicate with C&C server via multiple URLs generated by the DGA.
- *The number of MX record queries* (P6): is the count of mail exchange DNS queries issued by the host. It is a strong indicator of spam-based botnets in the network.
- *The number of PTR record queries* (P7): is the count of pointer DNS queries issued by the host. It helps detect hosts with anomalous behavior in a network and possible infection.

- *The number of distinct DNS servers queried* (P8): is the count of different DNS servers used by a host for DNS protocol. It helps detect machines with anomalous behavior in the network since it is very uncommon for a standard system to query more than one DNS server.

3.4.2. Domain-based Features

- *The number of distinct TLD queried* (P9): is the count of different top-level domains queried by the host. It is very effective in detecting DGA based bots which not only generate random domains with different Second Level Domain (SLD) but also TLDs.
- *The number of distinct SLD queried* (P10): is the count of different second-level domains queried by the host. It is a strong indicator of the presence of DGA based bots in the network.
- *Uniqueness ratio* (P11): is the ratio of the number of requests sent to the number of distinct requests sent under the assumption that the host has sent at least 1000 requests per hour. It is another strong indicator of the presence of DGA based bots in the network.

3.4.3. Response based Features

- *The number of Failed/NXDOMAIN queries* (P12): is the count of failed queries received by a host. It is a very strong indicator of host infection in the network. It maintains the number of responses received with response code equal to DNS_RCODE_NXDOMAIN by the host.

3.4.4. IP based Features

- *The number of distinct cities of resolved IP addresses* (P13): is the count of different cities of the resolved IP address. It is a strong anomaly indicator especially when IP addresses are spread across cities. The IP address to city mapping is obtained using Maxmind database[60].
- *Number of Countries of Resolved IP addresses* (P14): is the count of different countries of the resolved IP address. It is a strong anomaly indicator especially when IP addresses are

spread across countries. The IP address to country mapping is obtained using Maxmind database.

3.4.5. Mapping (FQDN-IP) Feature

- *Flux ratio* (P15): is the ratio of the distinct requests sent to the distinct resolved IP addresses under the condition that the host has sent at least 100 queries and has received at least 100 responses. It is a strong indicator of IP flux used for C&C communication.

DNS features have been studied extensively in variety of ways. However, these features were analyzed mostly from an ISP's perspective to detect C&C domains and not from host's perspective as discussed in Section 2.8. The prime reason being that DNS traffic captured at ISP level cannot be used for hosts profiling as hosts are mostly behind Network Address Translation (NAT). Host profiling using DNS traffic is possible only at network gateway of an enterprise i.e. before NAT. Our study reused 15 features extracted from hosts querying DNS service. The parameter namely "Number of queries" (P1), "Count of DNS servers contacted" (P8) and "Number of NXDOMAIN responses" (P12) were reported in [99]. The parameter namely "Count of distinct DNS records" (P2) was reported in [132]. The parameter namely "Number of distinct TLDs" (P9), "Number of distinct SLDs" (P10) and "Number of distinct Cities of resolved IPs Addresses" (P13) were reported in [133]. "Number of distinct countries of resolved IPs Addresses" (P14) were reported in [109]. "Average requests" (P4) was reported in [116]. To the best of our knowledge, "Highest request for a single domain" (P3), "highest number of requests per minute" (P5), "uniqueness ratio" (P11) and "flux ratio" (P15) are novel. Minimum, maximum and average values found for various parameters is shown in Table 3.1.

Table 3.1 Minimum, maximum, and average values for DNS parameters obtained

Parameter	Feature	Min.	Max.	Avg.
P1	Number of DNS requests per hour	1	53000	136.1
P2	Number of distinct DNS requests per hour	1	4918	51.9
P3	The highest number of requests for a single domain	1	52500	21.1
P4	Average number of requests per minute	0	4053	5.3

P5	The highest number of requests per minute	1	8580	24.2
P6	Number of MX record queries per hour	0	66	0.005
P7	Number of PTR record queries per hour	0	5462	0.89
P8	Number of distinct DNS servers queried per hour	1	42	1.01
P9	Number of distinct TLD domains queried per hour	1	185	4.25
P10	Number of distinct SLD domains queried per hour	1	2213	23.2
P11	Uniqueness ratio per hour	1.001	4022	14.06
P12	Number of failed/NXDOMAIN queries per hour	0	174	0.003
P13	Number of distinct Cities of resolved IP Addresses	0	188	1.7
P14	Number of distinct Countries of resolved IP	0	52	0.98
P15	Flux ratio per hour	0	329	0.43

3.5. Anomaly Detection Engine

An anomaly is the deviation of data from the normal value. The anomaly arises due to various real-life events which influence the value of data. Anomaly detection attempts to find such data values which are deviated and their possible correlation with the real-world event. Chandola et al. [84] presented a comprehensive survey of anomaly detection in various research areas. One of the challenging issues in anomaly detection is where and how to draw the fine line between normal and anomalous data. The output of anomaly detection is a set of labels attributed to the input data. Bhuyan et al [134] presented another interesting survey on Network anomaly detection. It covers various aspects of Intrusion detection, compares various network anomaly detection techniques and presents guidelines for performance measurement.

Stevanovic et al. [100] presented an analysis of common problem faced in determining the ground truth in the malware investigation of DNS traffic. It is difficult to say with the highest degree of certainty the number of bots existing in a real network at an instance. Moreover, all the bots in the network need not be active (contributing to DNS traffic in our case) at the same time and throughout the day. Botnet sometimes remains undetected for years [128] which is again a very serious concern in obtaining the ground truth.

Anomaly detection module in BotDAD attempts to find anomalous DNS query behavior from various hosts in a network. Bots infected with DGA malware often tend to generate many DNS requests. These requests query for domains with distinct top-level domains (TLDs) and second-level domains (SLDs) leading to the anomaly. These queries are requested in short bursts leading to a significant rise in the number of queries and as such leading to the anomaly. Another source of an anomaly in this malicious communication is the ratio of the number of distinct requests against the number of resolved IP addresses. Many domains used as C&C server often resolve to same IP address leading to the anomaly.

To determine the threshold values, an experiment was performed to check the number of DNS requests successfully resolved for loading homepage of top 100 Alexa domains [113]. The distribution for the number of DNS queries needed to load the homepage for the top 100 Alexa domains is shown in Figure 3.3.
Problems encountered in the correct detection of the number of queries are as follows:

1. DNS cache is maintained at OS and application level (e.g. Google Chrome browser) which has a greater influence in determining the number of queries sent as a query could be resolved in DNS cache if the TTL value of the domain is greater than zero.

2. Domains filtered/blocked at firewall level (Organizational policy) leading to the incorrect determination of the number of DNS request sent to load a webpage. Such domains are shown in red in Figure 3.3 and as such did not contribute to the number of requests resolved.

3. Settings (such as page prefetch), enabled by default in browsers to speed up page loading, significantly increases the query count by a host as it prefetches all the IP addresses mapped to a domain referenced in a web page.

To overcome these problems, DNS cache was cleared prior to sending a new request. Further, a browser which doesn't maintain an internal DNS cache was used to load the homepage. The average number of requests resolved for loading top 100 Alexa domains was found out to be

21. This number is particularly important in determining the threshold for various parameters used in our research.

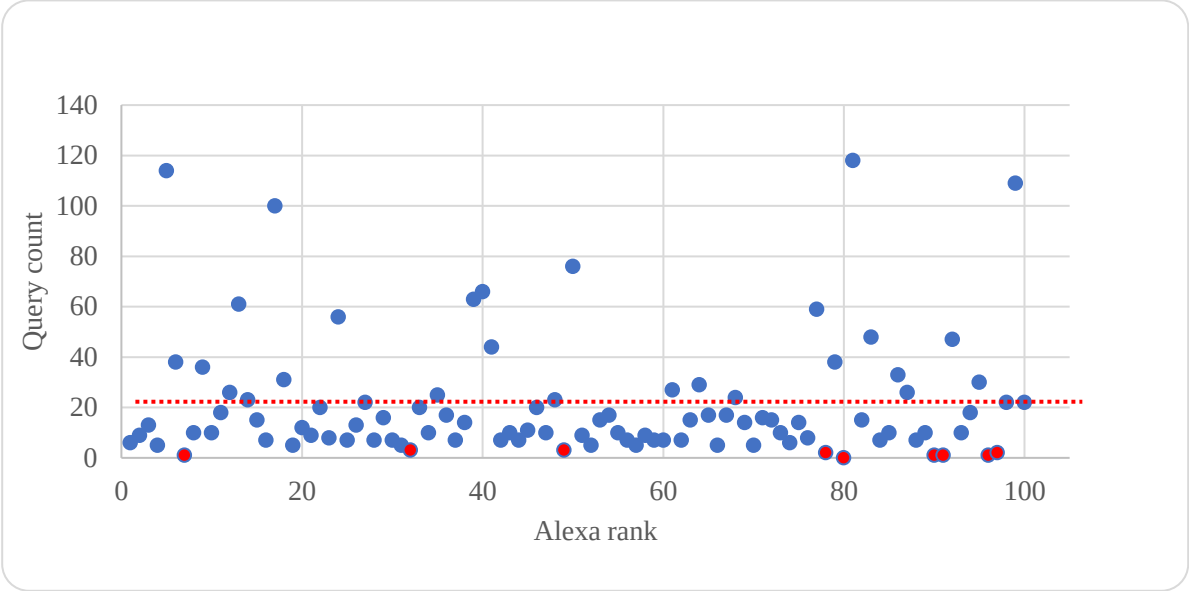
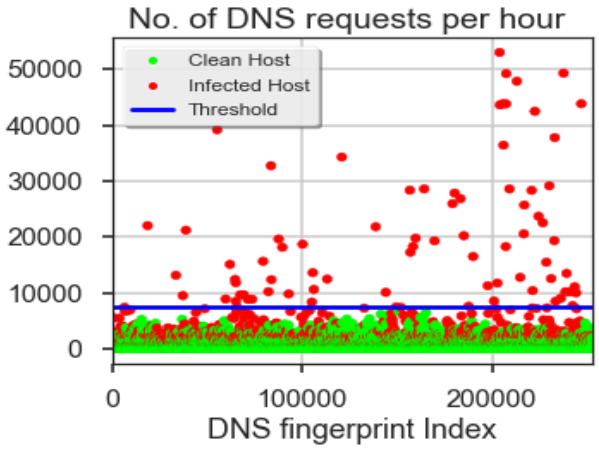
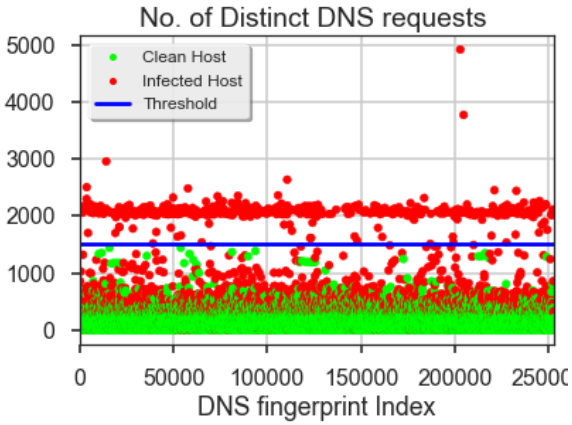


Figure 3.3 Scatter plot of the number of DNS queries for the top 100 Alexa domains

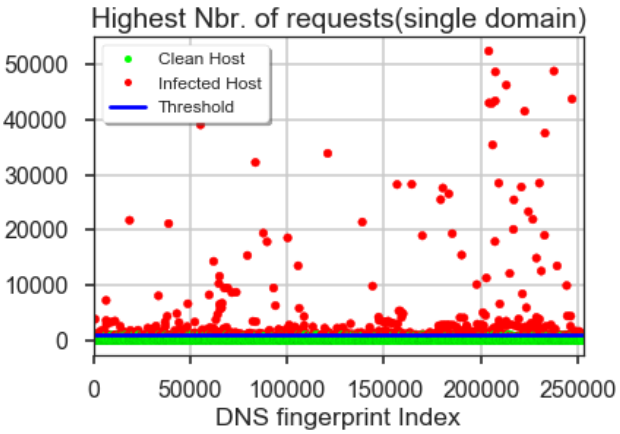
Host's Fingerprint generated by the DNS fingerprinting module acts as an input to the anomaly detector module which checks for values which exceed a certain threshold. Any value which exceeds the threshold is labeled as a bot. If there is no threshold violation, it is labeled as clean by the anomaly detection module. Multiple value threshold ensures that anomalies are detected at multiple feature level. The scatter plot for the various parameter used in DNS fingerprinting is shown in Figure 3.4. Scatter plot is quite useful in identifying the outliers/anomalies present in the data. Dots which exceeds a selected threshold is colored in red. Dots which don't exceed any threshold is colored as green. A threshold value for each feature is presented with a blue line in the scatter plots. Red dots above the blue threshold line in the scatter plot indicates the anomaly in that feature value. Whereas, red dots below the blue line indicates the anomaly in some other feature value.


No. of DNS requests per hour
Clean Host
Infected Host
Threshold
50000
40000
30000
20000
10000
0
0
100000
200000
DNS fingerprint Index

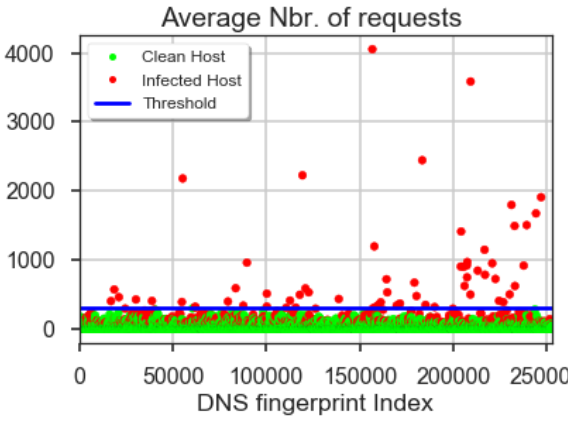
(a)


No. of Distinct DNS requests
Clean Host
Infected Host
Threshold
5000
4000
3000
2000
1000
0
0
50000
100000
150000
200000
25000
DNS fingerprint Index

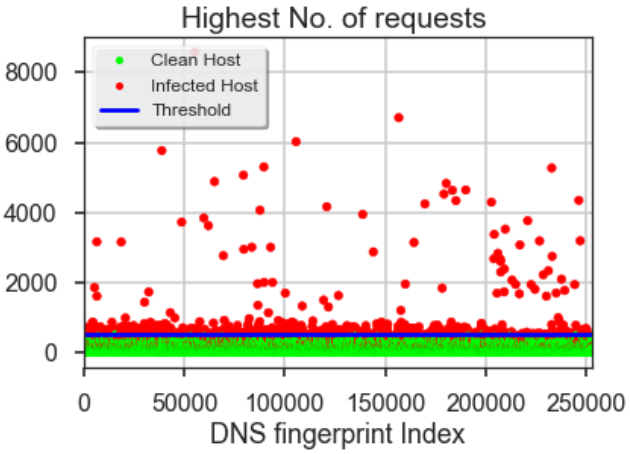
(b)


Highest Nbr. of requests(single domain)
Clean Host
Infected Host
Threshold
50000
40000
30000
20000
10000
0
0
50000
100000
150000
200000
250000
DNS fingerprint Index

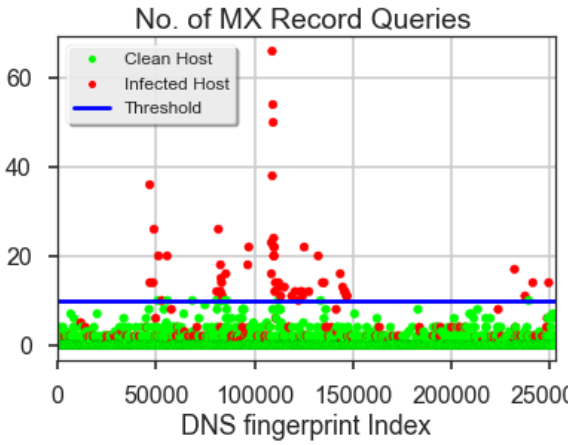
(c)


Average Nbr. of requests
Clean Host
Infected Host
Threshold
4000
3000
2000
1000
0
0
50000
100000
150000
200000
25000
DNS fingerprint Index

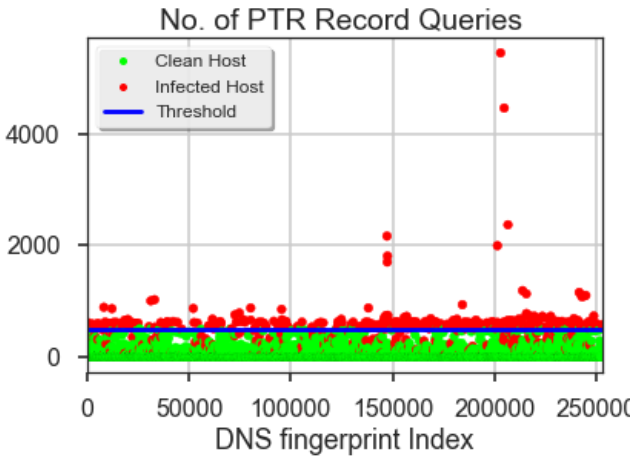
(d)


Highest No. of requests
Clean Host
Infected Host
Threshold
8000
6000
4000
2000
0
0
50000
100000
150000
200000
250000
DNS fingerprint Index

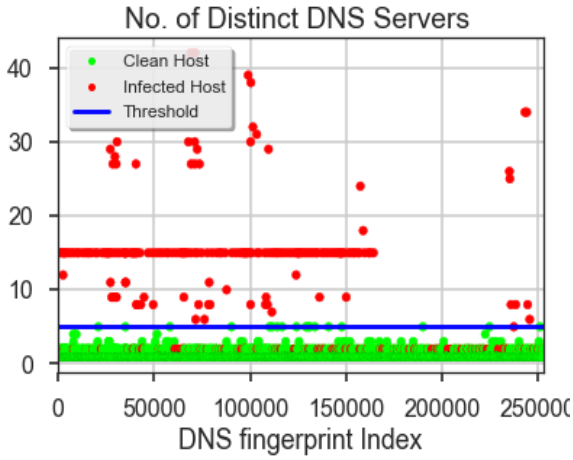
(e)


No. of MX Record Queries
Clean Host
Infected Host
Threshold
60
40
20
0
0
50000
100000
150000
200000
25000
DNS fingerprint Index

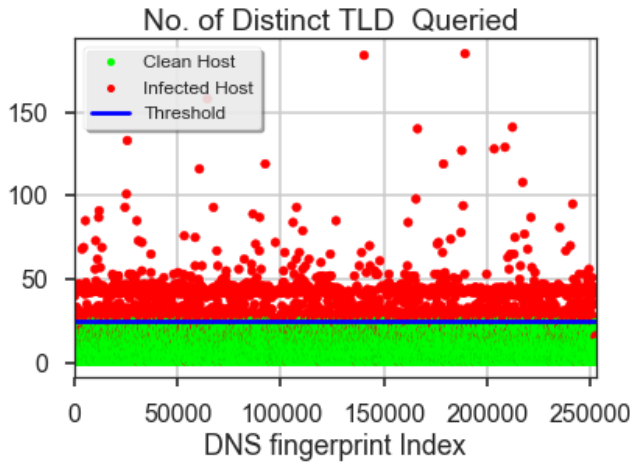
(f)


No. of PTR Record Queries
Clean Host
Infected Host
Threshold
4000
2000
0
0
50000
100000
150000
200000
250000
DNS fingerprint Index

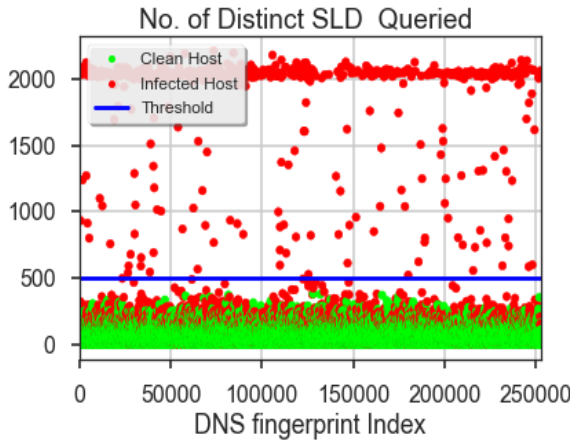
(g)


No. of Distinct DNS Servers
Clean Host
Infected Host
Threshold
40
30
20
10
0
0
50000
100000
150000
200000
250000
DNS fingerprint Index

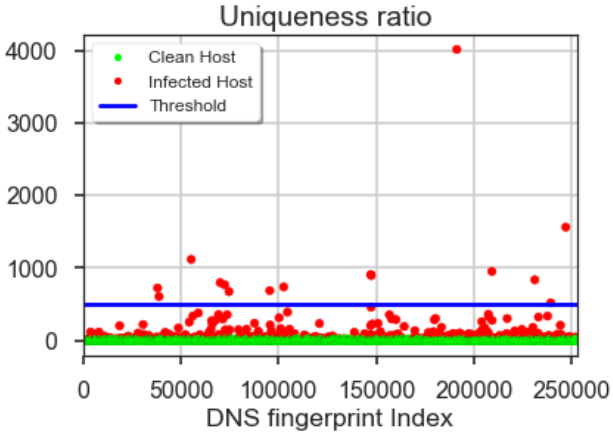
(h)


No. of Distinct TLD Queried
Clean Host
Infected Host
Threshold
150
100
50
0
0
50000
100000
150000
200000
250000
DNS fingerprint Index

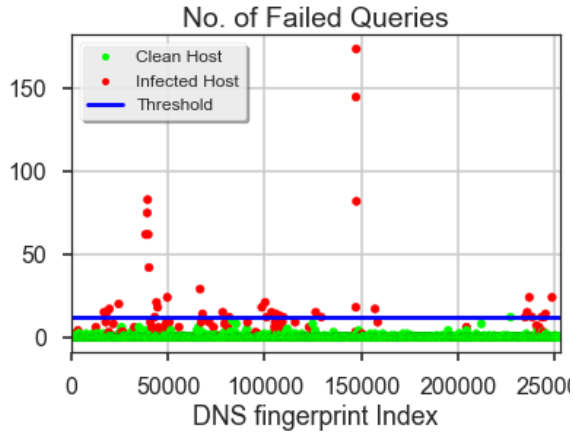
(i)


No. of Distinct SLD Queried
Clean Host
Infected Host
Threshold
2000
1500
1000
500
0
0
50000
100000
150000
200000
250000
DNS fingerprint Index

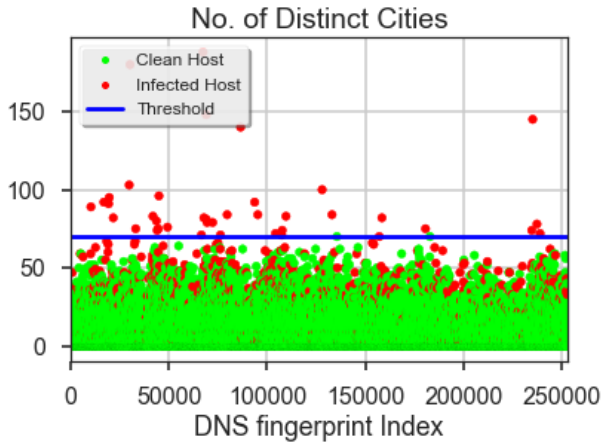
(j)


Uniqueness ratio
Clean Host
Infected Host
Threshold
4000
3000
2000
1000
0
0
50000
100000
150000
200000
250000
DNS fingerprint Index

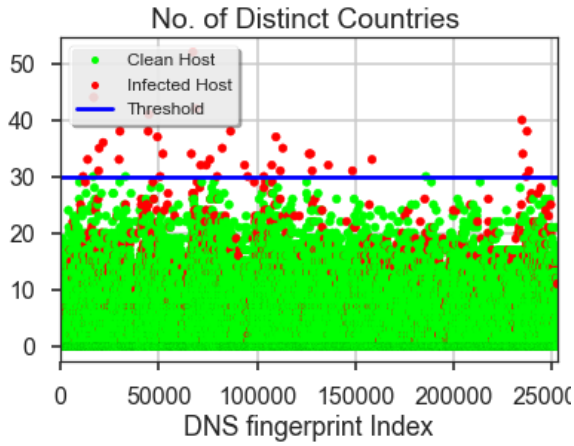
(k)


No. of Failed Queries
Clean Host
Infected Host
Threshold
150
100
50
0
0
50000
100000
150000
200000
DNS fingerprint Index

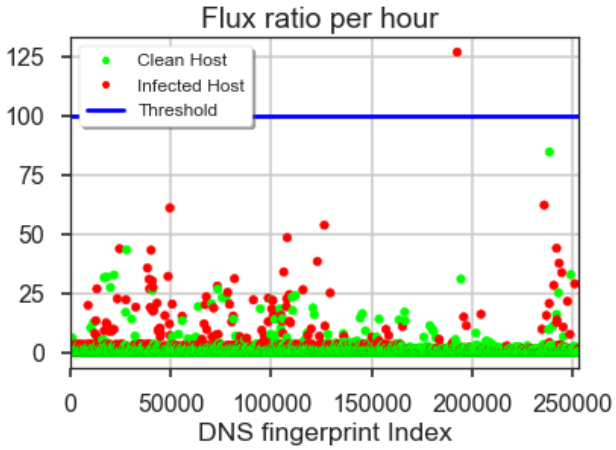
(l)



(m)



(n)



(o)

Figure 3.4 Scatter plot of various parameters used in DNS fingerprint

3.6. Training the Classifier

Classification is the problem of finding the category of a new record based on existing records for which category is known (also called as labeled dataset). It typically involves training the classifier with the labeled dataset and testing the trained model on the new record. Random forest classifier is an ensemble algorithm which uses decision trees to classify the class of the input features. The random forest classifier has enhanced accuracy and controls overfitting as compared to decision trees. It is easily customizable and can handle large datasets. Typical parameters used for tuning Random Forest Classifier include:

- The number of estimators: The number of decision trees in the forest.
- Maximum depth: The maximum depth of the decision tree.
- Maximum features: The number of features to consider when looking for the best split.

Once the DNS fingerprint entry is labeled as a bot or clean by the anomaly detection module, it is used as an input to train and test the classifier module. A total of 6,08,737 DNS fingerprints were generated in the 10 days evaluation period which is equal to sixty thousands of DNS fingerprints per day. This further resolves into 2500 active host per hour using DNS service on an average. Due to the presence of several hosts whose number of requests were very small thereby leading to unnecessarily large computation with little result, only those hosts which had at least 100 DNS requests were processed. This led to a considerable reduction in the number of DNS fingerprints to 2,52,742.

The classifier module loads all the DNS fingerprints along with the labels produced by the anomaly detection engine. LabelEncoder [135] is used to encode the categories (i.e. bot and clean) into zero and one. The resultant data is then split up into training input, training labels, test input, and test labels. The data split up use 70:30 ratio where 70% of the data is used for training the classifier while 30% is used for testing. The model is trained on training input and training labels. The resultant trained model is then used to predict test input. The predicted test labels are then compared with actual test labels to calculate accuracy. Cross-validation is done using GridSearchCV [136] with varying hyperparameters values like the number of estimators, maximum depth, maximum features, minimum sample split, minimum sample leaf, criterion, etc. in RF classifier. Intel distribution for Python [137] which uses an optimized mathematical library for computation was used in the classifier module. Scikit-learn [57] was used as a machine learning package.

3.7. Summary

This chapter described the design of BotDAD and its sub-components. BotDAD is a DNS anomaly detector system for detecting infected hosts in a network by monitoring DNS traffic

on an hourly basis. It uses a feature extractor module to extract DNS attributes and build a host profile for all hosts in the network. The host profile is then parsed to generate DNS fingerprint. BotDAD creates DNS fingerprint of each host in the network and uses anomaly detection engine to label them as bot or clean. BotDAD uses a machine learning classifier to develop a trained model for future predictions. Hyperparameters tuning for the Random Forest classifier is done to improve the accuracy of the trained model.

4. Implementation and Experimental Results

4.1. Introduction

This chapter discusses the implementation of BotDAD in practice. The proposed system was tested against DNS network traffic captured from campus on an hourly basis. Each file is then parsed to produce a fingerprint database. The fingerprint database is then used as an input to the machine learning module to produce results. The results are then evaluated using standard metrics.

This chapter starts with the description of the dataset used for experimentation. Section 4.2 discusses the day-wise and hour-wise details of the dataset. Section 4.3 describes the importance of features as a result of the Random Forest classifier. Section 4.4 discusses the problem of the imbalanced dataset and the solution. Section 4.5 covers the error metrics used for validation. BotDAD detection on online datasets is covered in section 4.6. To improve the accuracy of the model, hyperparameter tuning is described in section 4.7. BotDAD screenshots are shown in section 4.8. Limitations of BotDAD tool is covered in section 4.9. The detailed implementation and results are published in [67] and this chapter is part of that paper with additions.

4.2. Dataset

Campus network traffic consisting of more than 4000 active users (in peak load hours) for random days in the month of April-May 2016 was collected and stored on an hourly basis for the proposed work. Dates and corresponding capture file size is shown in Table 4.1. The volume of traffic for the dataset is shown in Figure 4.1.

Table 4.1 Capture period and Pcap size

Day #	Date	Files Size (Bytes)	Files Size (GB)
Day 1	24-04-16	8,136,119,258	7.58
Day 2	25-04-16	10,700,800,279	9.97
Day 3	27-04-16	12,329,022,191	11.48
Day 4	28-04-16	10,756,459,156	10.02
Day 5	29-04-16	10,402,040,969	9.69
Day 6	30-04-16	7,486,883,674	6.97
Day 7	01-05-16	8,328,626,700	7.76
Day 8	07-05-16	7,708,523,953	7.18
Day 9	08-05-16	8,116,375,802	7.56
Day 10	09-05-16	7,833,444,297	7.30
Total Size			**85.49**

While there is drop-in traffic from 2 AM to 8 AM, it starts rising from 8 AM till 5 PM and then it maintains a normal position. There is little similarity between the daily traffic because of the influence of the academic, cultural and sports events on the volume of network traffic. Day wise hourly traffic for the dataset is represented in Figure 4.2.

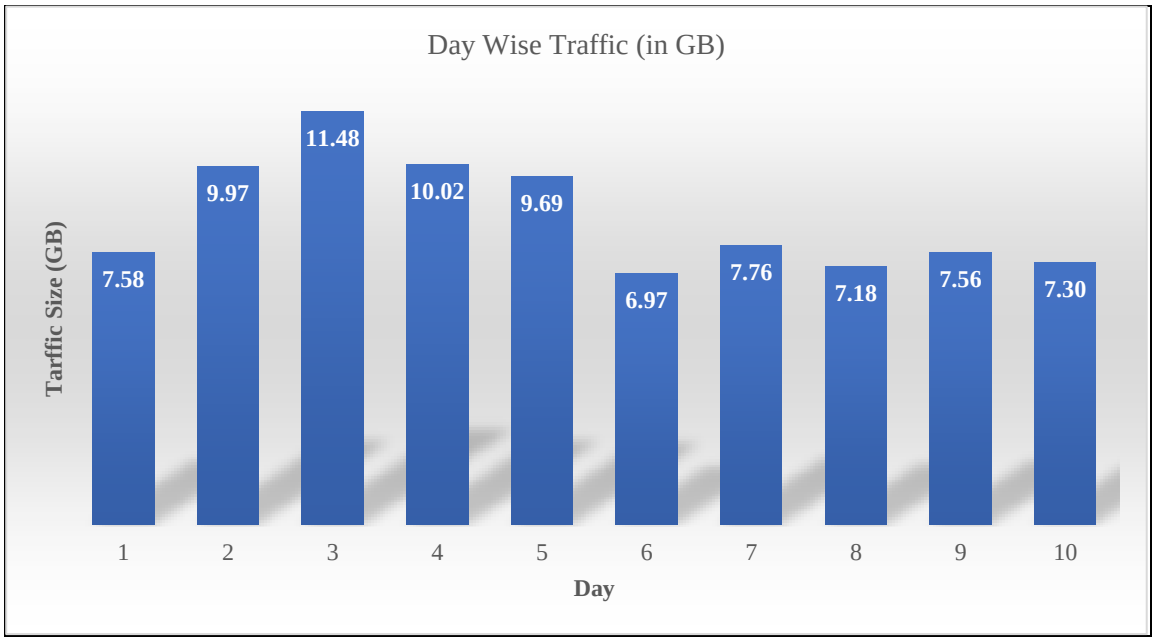


Figure 4.1 Day-wise traffic (in GB)

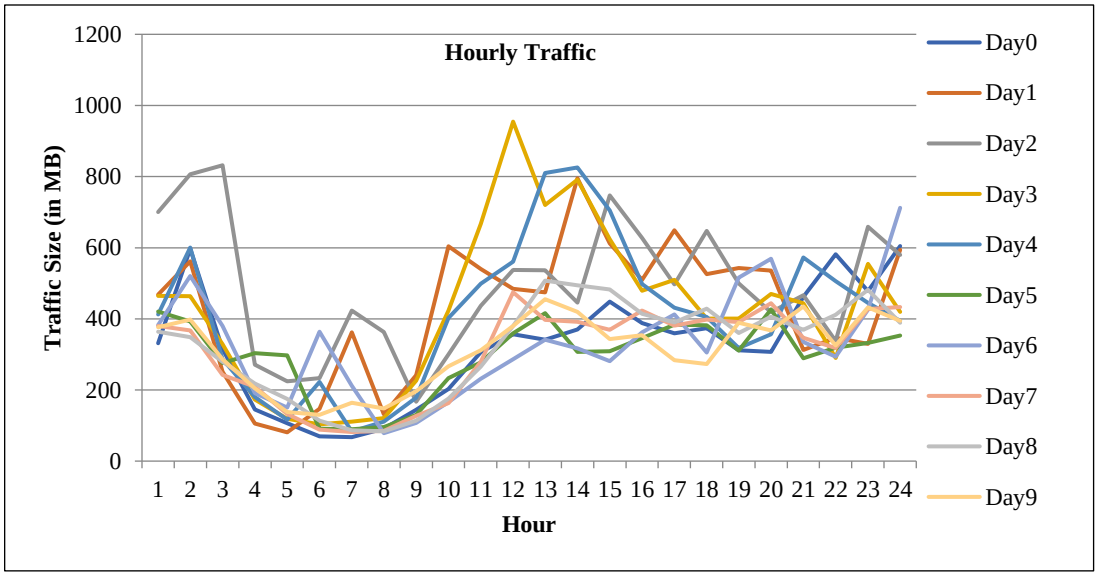
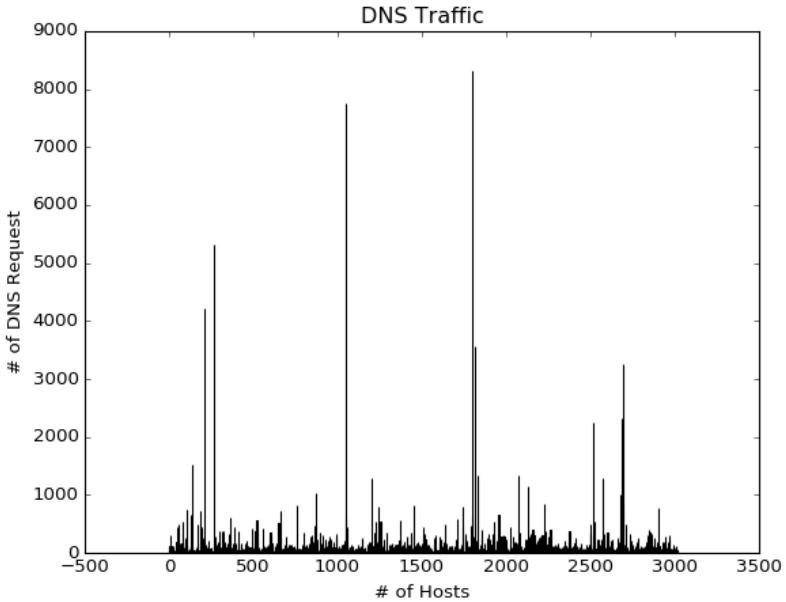
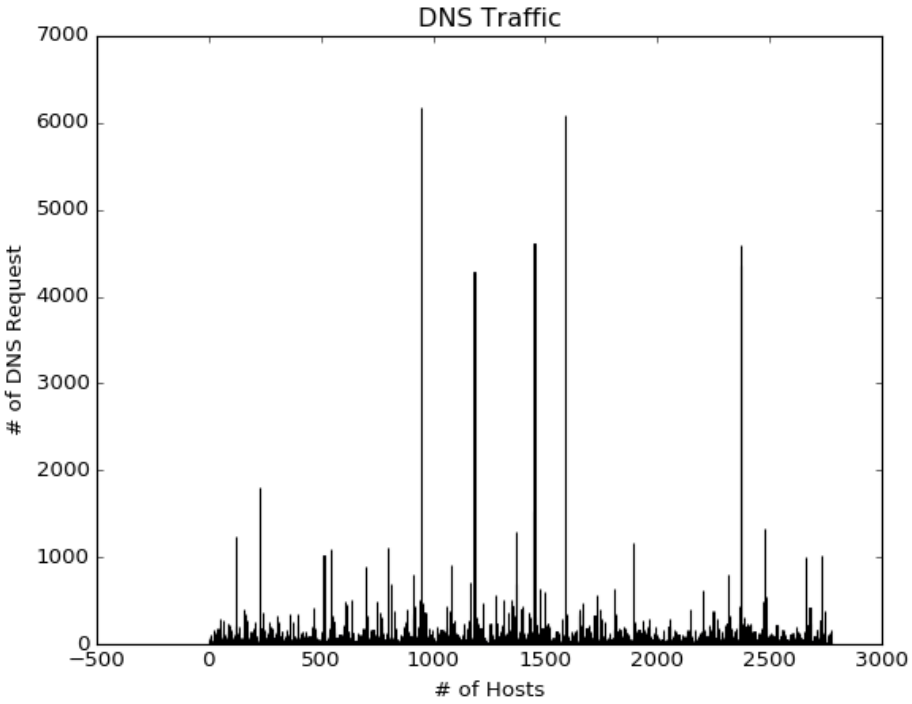


Figure 4.2 Hourly traffic captured for ten days on campus

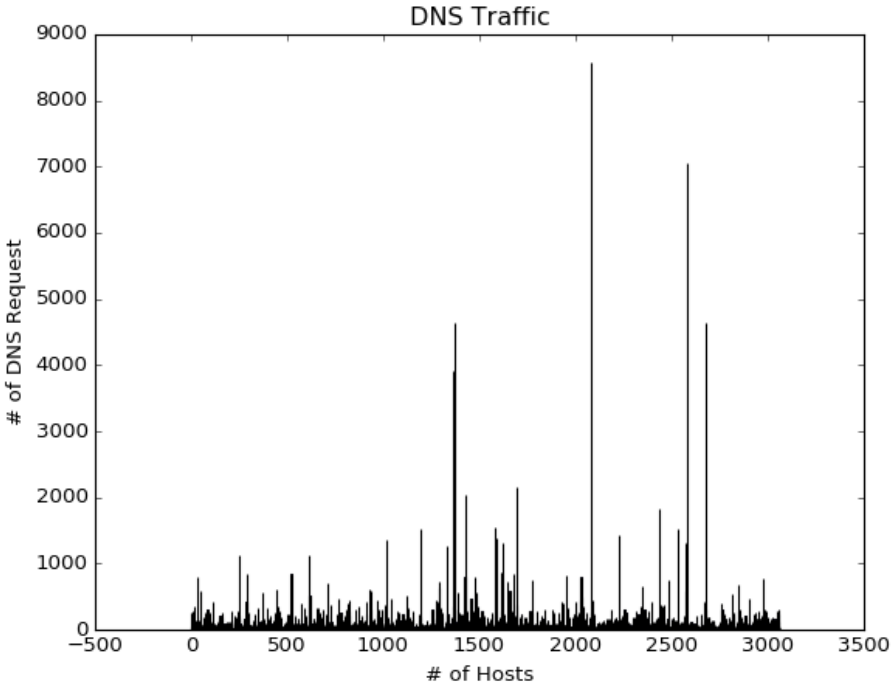
The number of DNS requests sent by the hosts in an hour is shown in Figure 4.3. It clearly indicates that there are some hosts whose DNS request count is abnormally high. This is an early sign of the presence of infected hosts in the captured traffic. Abnormally high queries per hour (e.g. 8,000) clearly shows an anomaly in the query pattern. The maximum number of domain tokens in the FQDN was found out to be between 2 to 4 as shown in Figure 4.4.



a)



b)



c)

Figure 4.3 Histogram for the number of DNS requests sent by hosts (a-c)

On further investigating the query pattern of hosts with abnormally high queries per hour, it was found out that some hosts are generating very high short bursts of DNS requests for 3-4 minutes before going to normal behavior. Figure 4.5 (a) and (b) represents the query pattern of infected hosts in a network thereby indicating a short burst of massive DNS requests per minute. Figure 4.5 (c) and (d) presents the query pattern of benign hosts in a network without any significant burst.

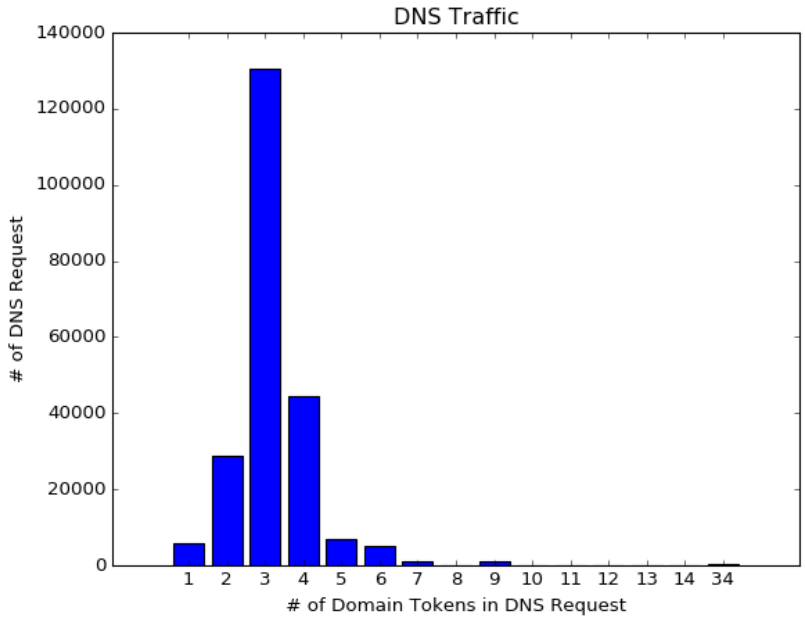


Figure 4.4 Histogram for the number of domain token in DNS queries

On further exploring these short burst of DNS traffic, it was observed that these domains are algorithmically generated. DGA domains can be classified into two main categories.

1. Pronounceable Domains: These are the domains which are algorithmically generated but consists of a combination of words which are pronounceable and seems to suggest a genuine domain name as shown in Table 4.2.
2. Non-Pronounceable Domains: These domains are algorithmically generated but not pronounceable as shown in Table 4.3.

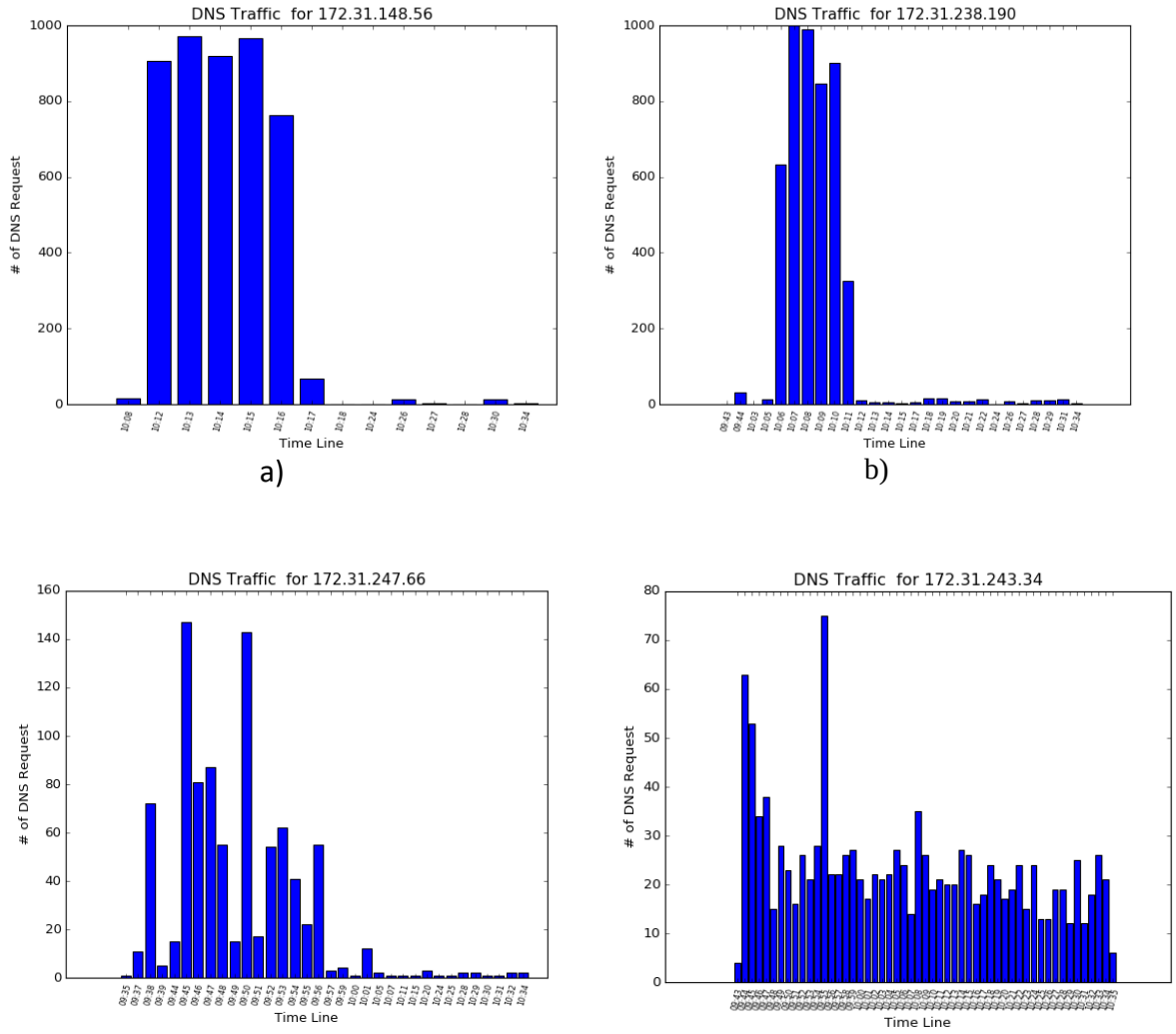


Figure 4.5 DNS query pattern for infected hosts (a and b) and clean hosts (c and d)

In order to validate the domains queried by the host classified as bots, these domains were checked in DGArchive [138] which is a repository for DGA domains. It was observed that these domains are in fact used by Conficker botnet to communicate on 21/04/2016 which matches the date of our dataset capture period. Figure 4.6 presents a screenshot of domain lookup in DGArchive.

Table 4.2 Pronounceable domains

Pronounceable domains
myburgeriscoming.com
myplatesareclean.com
thissoundisgood.com
youcantstolemine.com
theskyisblack.com
drinkwineandbefine.com
iwanttobeus.com
smokeisbad.com
ifyouseeit.com

Table 4.3 Non-Pronounceable domains

Non-Pronounceable domains
Cnhzwv.ws
Fofkq.ws
Awbuec.ws
Jzchyn.ws
Iaugaz.ws
Arnmtyqfa.ws
Ozcbdqexlkv.ws
Klwvjpd.ws
Mwjydmbq.ws
Itsdipxxta.ws

In order to further explore these domains, FQDN to IP mappings were plotted using a python library named GephiStreamer [62] for streaming graphs to Gephi [63]. The graphs thus obtained are shown in Figure 4.7. The graphs clearly indicate the presence of pronounceable and non-pronounceable flux in the FQDN to IP mapping.

Perform lookup

Current search: 10 domains, query completed in 0.406 seconds -

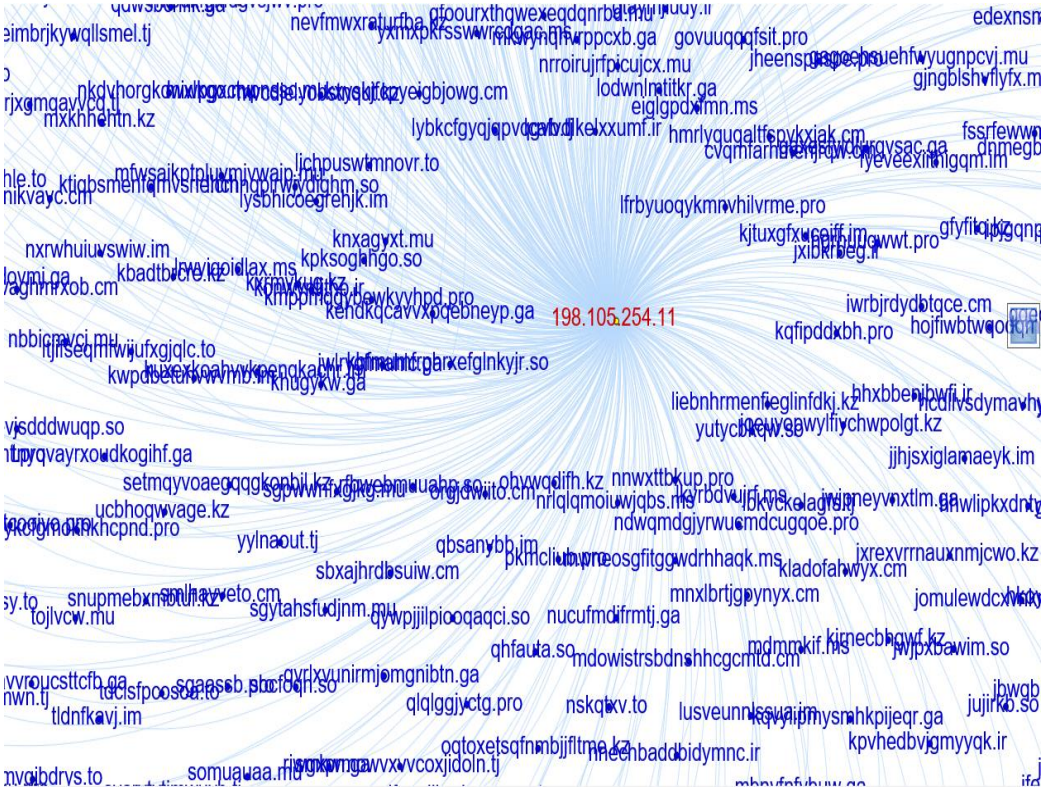
Database Results

#	Domain	Domain ID	Family	Valid from	Valid until
1	cnhzwv.ws	176	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
2	mwjydmbq.ws	139	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
3	klwvjpd.ws	127	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
4	awbuec.ws	122	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
5	arnmtyqfa.ws	94	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
6	fofkq.ws	63	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
7	itsdipxxta.ws	14	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
8	iaugaz.ws	5	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
9	jzchyn.ws	0	conficker_dga_69998fdd	2016-04-21 00:00:00	2016-04-21 23:59:59
10	ozcbdqexlkv.ws	-	-	-	-

Figure 4.6 DGArchive query lookup results


vdcggauqwft.ws
xdqkqytov.ws
ealgolupqc.ws
jzchyn.ws
ijbkq.ws
cnhzwv.ws
awbuec.ws
bwbiv.ws
oljedpvd.ws
38.102.150.27
fofkq.ws
iaugaz.ws
itsdipxxta.ws
klwvjpd.ws
qsmueeb.ws
ozcbdgexlkv.ws
arnmtyqfa.ws
hqzdradv.ws
mwjydmbq.ws

a)


198.105.254.11

b)

c)

Figure 4.7 Pronounceable (c) and non-Pronounceable flux (a &b)

Several infections detected per hour by BotDAD are then analyzed and it was observed that there is a repetition of bot-infected machines in consecutive hours. Some hosts were found to be labeled positive for five consecutive hours. Heatmap for several infections per hour is shown in Figure 4.8.

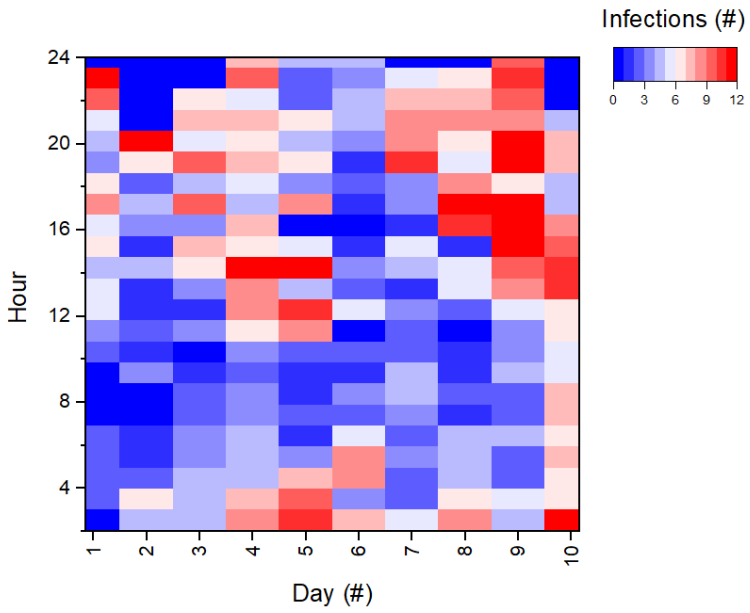


Figure 4.8 Heatmap of hourly bot infection detected by BotDAD

4.3. Feature Importance

In machine learning projects, it is often desirable to know the relative importance of features in determining the outcome so as to get an insight into the input features and possible future improvements. A detailed discussion on features importance in Random Forest classifier is explained in [139]. To determine the effectiveness of various features used in BotDAD, the feature importance was calculated in the model as shown in Table 4.4. "*No. of distinct DNS servers*" queried by the host was found out to be the most significant feature. It was closely followed by "*Flux ratio*" and "*No. of Distinct SLD Queried*". Figure 4.9 presents a bar chart of various feature importance.

Table 4.4 Feature importance

Rank	**Parameter**		**Percentage**
1	No. of Distinct DNS Servers	(P8)	0.235813
2	Flux ratio per hour	(P15)	0.181127
3	No. of Distinct SLD Queried	(P10)	0.170091
4	No. of DNS requests per hour	(P1)	0.159189
5	Uniqueness ratio	(P11)	0.087488
6	No. of Distinct TLD Queried	(P9)	0.066934
7	No. of Distinct DNS requests	(P2)	0.030053
8	No. of Distinct Countries	(P14)	0.018931
9	Highest No. of requests (single domain)	(P3)	0.017911
10	No. of Distinct Cities	(P13)	0.011652
11	No. of Failed Queries	(P12)	0.006967
12	Highest No. of requests	(P5)	0.00521
13	No. of PTR Record Queries	(P7)	0.004929
14	No. of MX Record Queries	(P6)	0.003642
15	Average No. of requests	(P4)	0.000061

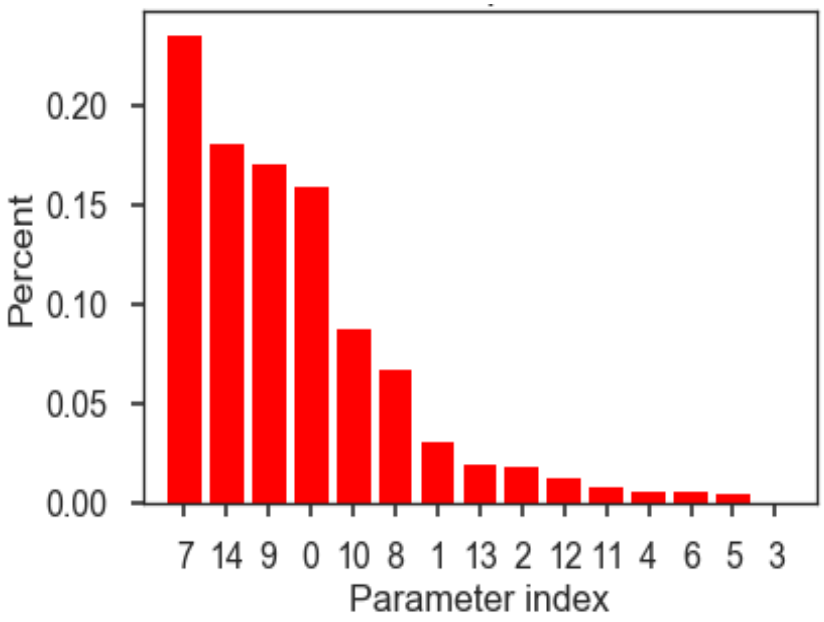


Figure 4.9 Bar plot of feature importance

4.4. The Problem of the Imbalanced Dataset

One of the problems encountered in training the machine learning classifier was the imbalance in the number of DNS fingerprints available for bot and cleans hosts. There were 247498 benign cases and 5243 malignant cases in our dataset and as such there was great bias in the learned model. To overcome this problem, Synthetic Minority Oversampling Technique (SMOTE) algorithm [140], [141] was used which upscales the lower class or downscales the higher class. This is particularly useful when the number of elements of one class is outnumbered by elements of other class. SMOTE is an improved method of enhancing the number of infrequent cases than merely repeating current cases. Typically, SMOTE is used when the target class is under-represented. This led to training on an unbiased model as the number of bot fingerprints was now in the same range as the number of clean fingerprints.

4.5. Error Metrics

Confusion matrix generated on the test data from the trained module is shown in Figure 4.10. To measure the effectiveness of the detection system, the following measures were calculated.

- $True\ Positive\ Rate(TPR)\ also\ called\ as\ Sensitivity = \frac{TP}{TP+FN}$
- $True\ Negative\ Rate\ (TNR)\ also\ known\ as\ Specificity = \frac{TN}{TN+FP}$
- $False\ Positive\ Rate\ (FPR) = \frac{FP}{FP+TN}$
- $False\ Negative\ Rate\ (FNR) = \frac{FN}{FN+TP}$
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

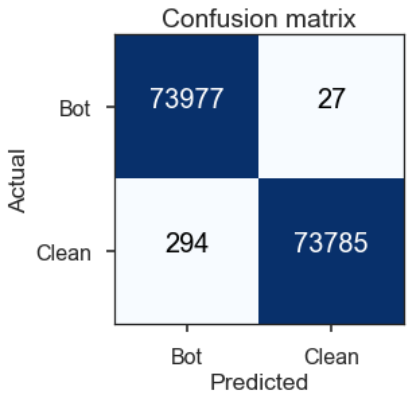


Figure 4.10 Confusion matrix

High TPR of 0.996, TNR of 0.9996, FPR of 0.003 and FNR of 0.0003 was obtained using the detection system. The accuracy of the detection system was found to be 0.9978 as shown in Figure 4.11.

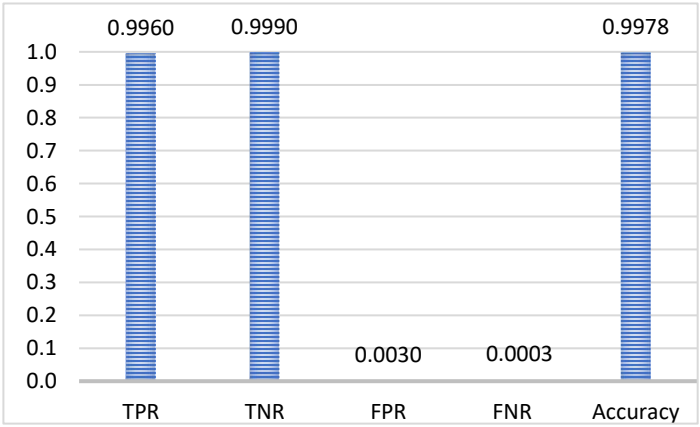


Figure 4.11 Validation results

4.6. Analysis of false positives and false negatives

On manual analysis of false positives, it was observed that some of the hosts querying domains that are hosted on Content Delivery Networks (CDN) like akamai.net, cloudflare.net, cloudfront.net, amazonaws.com, etc. resulted in large number of resolved IP addresses in the response records. This resulted in higher Flux ratio resulting in false positives. This behavior of hosts resembles similar behavior to bots using domains employing IP flux. These false positives can be minimized by selecting the Second Level Domain(SLD) and Top Level Domain (TLD) in the URL (e.g. djya0coy7zpa6.cloudfront.net) and discarding the other domain tokens. However some domains like (*.ac.in, *.ac.uk, etc.) may require 3-LD domain tokens as well. Whitelisting CDN domains is another option to minimize the false positives. However, this approach may result in excluding bots using C&C hosted on CDN domains.

On manual analysis of false negatives, it was observed that some of the bots which remain active for a smaller duration in the capture period (1-hour pcap file) or joined the network late during the capture period resulted in false negatives. Since the fingerprint is calculated over a duration of one hour, hosts joining network late during the capture period have a possibility of being labelled as clean. False negatives can be minimized by calculating hourly fingerprinting of each hosts on the basis of actual hour spent using DNS service instead of the hourly capture period.

4.7. Detection of Online Datasets

To evaluate the effectiveness of the BotDAD, online datasets available under “Malware Capture Facility Project” [142] were tested. Two online datasets [143], [144] containing network traffic of hosts infected with DGA malware were selected to validate the model. The dataset consists of a single capture file consisting of days of network traffic of infected host which was converted into hourly capture file using *tshark* [145], which is a command-line tool available with *wireshark*. These hourly packet captures were then analyzed by BotDAD. BotDAD was able to report bot-infected machine for all the hourly capture files thereby proving the effectiveness of the proposed system to detect bot-infected machines in a network.

4.8. Hyperparameter Tuning

Hyperparameters are the attributes of an algorithm that can be changed to improve the outcome of an algorithm. To improve the accuracy of the machine learning model, a three-cross validation with various hyper-parameters available in random forest classifier was performed. Various hyperparameter values explored for accuracy are:

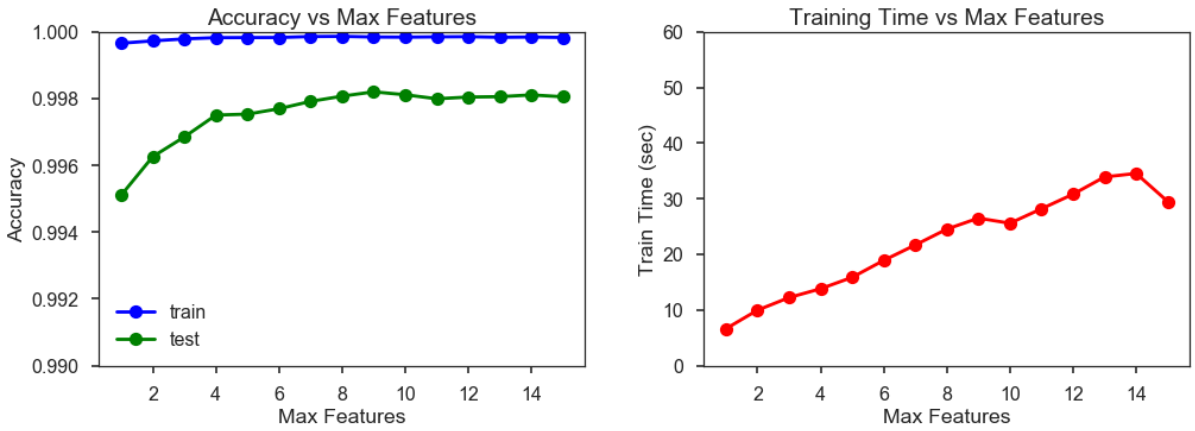


Figure 4.12 Effect of maximum features on accuracy

- Number of Features

This parameter in the Random Forest classifier is used to configure the number of features to consider when looking for the best split. Figure 4.12 suggests that as the number of maximum features increases, the accuracy of the detection system also increases. However, there is a greater degree of computation cost involved as the number of features uses. The computation time nearly doubles as the features are increased threefold. The computation time triples as the features are increased six-fold.

- Number of estimators

This parameter is used to configure the number of trees in the forest. It has a default value of 10. Figure 4.13 shows, as the number of estimators increases, the accuracy of the detection system also increases. Thereafter, it reaches a saturation point where no further significant improvement is observed. The computation time, on the other hand, increases linearly and saturates toward the end.

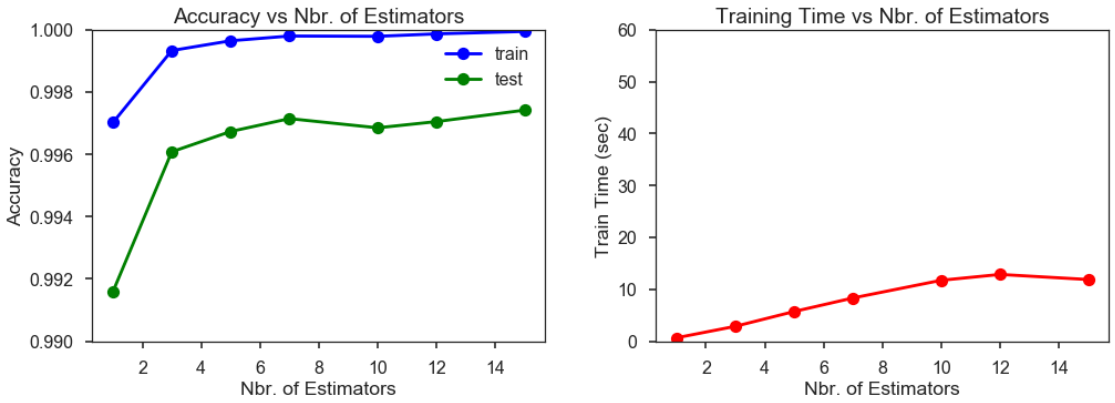


Figure 4.13 Effect of number of estimators on accuracy

- Maximum Depth

Maximum depth is another parameter that can be tuned in Random forest classifier. It sets the maximum depth of the tree. Figure 4.14 displays, as the maximum depth of the RF classifier increases, the accuracy of the detection system increases rapidly and reaches a point thereafter there is no significant improvement. The computation time, on the other hand, doesn't show significant change.

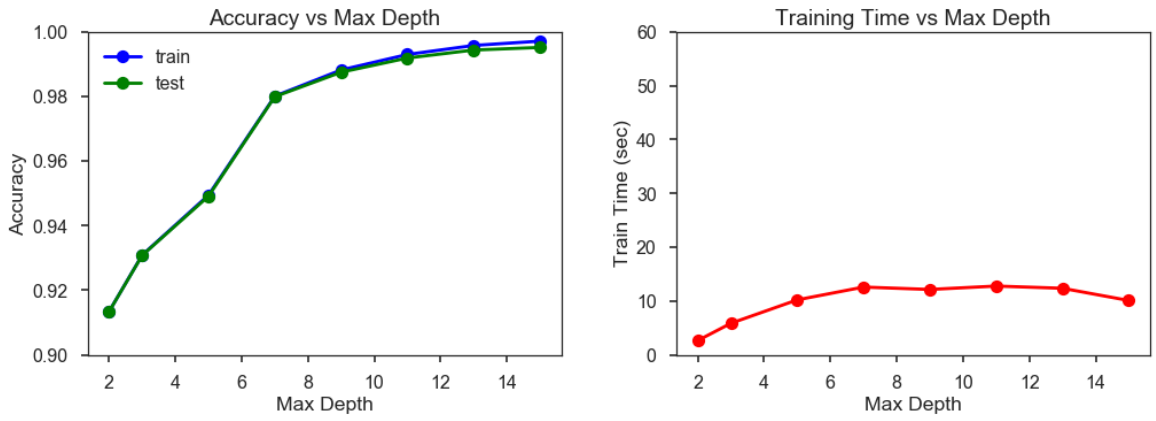
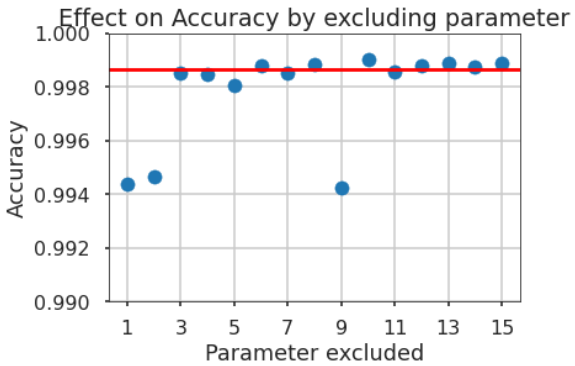


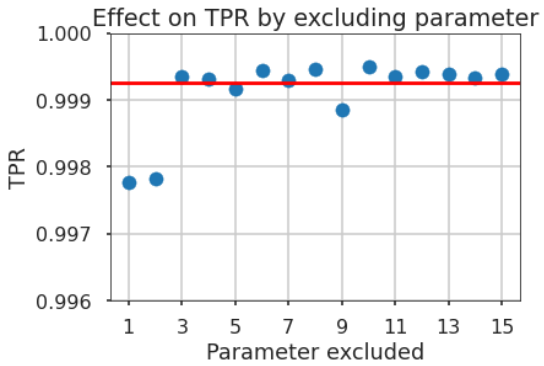
Figure 4.14 Effect of maximum depth on accuracy

4.9. Features subset for better performance

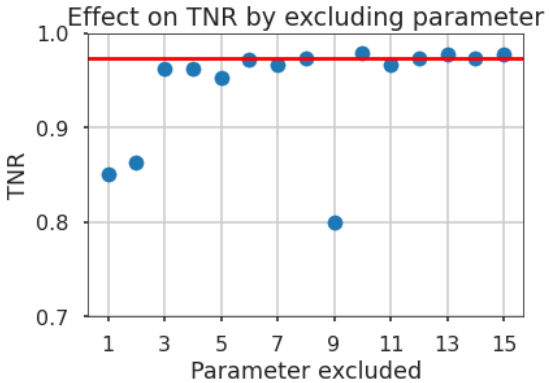
In order to select subset of parameters for better performance, we experimented the effect on the various performance metrics by excluding each feature and compared it with the performance of the complete parameters. The result thus obtained are shown via a scatter plot in Figure 15(A-E). Each blue dot indicates the value of the performance metric by excluding the corresponding parameter in the x axis. The red line indicates the value of the performance metric by using all the parameters. As evident from Figure 15(A), excluding parameter "The number of DNS requests per hour " (P1), "The number of distinct DNS requests per hour" (P2) and "The number of distinct TLD queried" (P9) has a significant decline in the accuracy of the system clearly indicating the importance of the parameters in the system. Similar pattern could be observed in the TPR and TNR scatter plots. There was significant increase in FPR and FNR when parameter P1, P2 and P9 were excluded again indicating their relative importance in the parameter list.



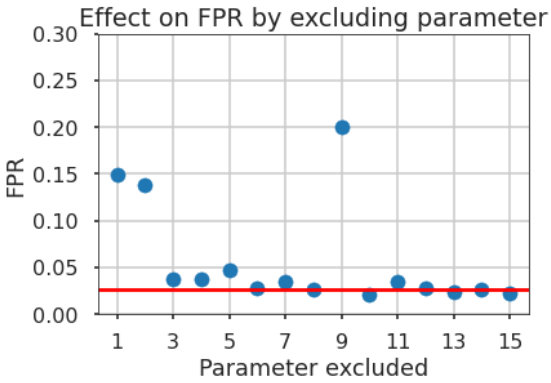
A



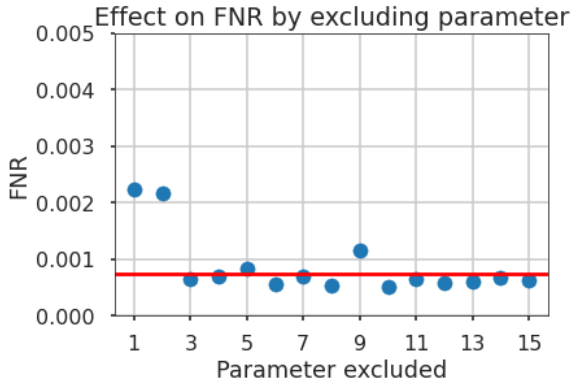
B



C



D



E

Figure 15 Effect on A) Accuracy B) TPR C) TNR D) FPR and E) FNR by excluding parameter (P1 to P15)

In order to select the subset of parameters with maximum performance value, we tried excluding more than one features and analyzed the effect on the performance metrics. When parameters P6 (Number of MX records queries per minute), P8 (Number of distinct DNS servers queried per hour) and P10 (Number of SLD domains queried per hour) were excluded we obtained the highest accuracy of 0.9990 with TPR, TNR, FPR and FNR values of 0.9995, 0.9767, 0.023 and 0.0004 respectively. These results surpassed the performance metric values obtained by using all the parameters (Accuracy = 0.9986, TPR =0.9992, TNR= 0.9733, FPR = 0.0266 and FNR = 0.007) clearly indicating these parameters have limited contribution in the performance valuation.

4.10. BotDAD Interface

BotDAD [56] is available on the Github development platform with installation and setup instruction available at [146]. Screenshots of BotDAD parsing is shown in Figure 4.16. The module starts with parsing the PCAP file and generates the CSV file for both request and response features. Fingerprinting module generates the DNS fingerprint by using the Host profiling schema. Anomaly detection checks for the anomaly and labels the record as either bot or clean. It finally displays a console to investigate the results and plot the graphs. It offers the following options in the console.

- h: List the hosts with over 100 distinct DNS requests.
- d: Displays the hosts DNS queries and responses.
- p: Plot the host's query pattern minute wise.
- F: Searches for all hosts who have queried for a given domain.
- f: Searches for all hosts who have received a given IP address (as a response) for any domain.

```
BotDAD Ver 0.2
===============
        Verbose : 1
        Mode    : 3

=============== PCAP Processing Started at  2018-11-20 05:12:38.737000 ===========

Packets (#)             Time Taken
        10              0:00:00.187000
       100              0:00:00.218000
      1000              0:00:00.343000
     10000              0:00:01.346000
    100000              0:00:12.093000
   1000000              0:02:05.056000

=============== PCAP Processing completed at 2018-11-20 05:17:28.093000 ==========

Total number of Packets Processed        : 2375526
Total number of DNS Query                : 1041483
Total number of DNS Responses            : 1334043
Total number of Unknown Response Records : 0
Total number of Failed Responses         : 122691
Total Time taken                         : 0:04:49.387000

Number of infected Hosts = 49

Number of Clean Hosts = 1872

l - list         m - Save Map    p - plot       d/D - Display/Save       h - saveHtml    x - saveCSV
 F - Find Req URl        f - Find Resolved IP    q - quit
console>
```

a) PCAP parsing

```
console>
l
Hosts with over 100 distinct requests
1.      172.31.157.166  9715
2.      172.31.148.4    354
3.      172.31.234.108  1885
4.      172.31.250.252  1061
5.      172.31.123.27   517
```

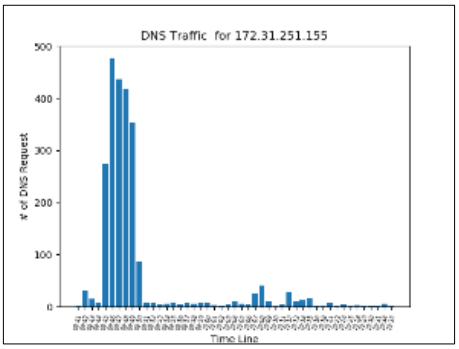
b) List of top hosts

```
console>
d
Enter Hostname :
172.31.251.155
('172.31.251.155', 4557, 0, 2154, 0, 0, 0, 0, 0, 0, 0, 0)
Request:        34267   aqgmekpyyhxyrnly.eu     1       21/04/16 09:46:20
Request:        51328   oqtoxetsqfnmbjjfltme.kz 1       21/04/16 09:46:39
Response:       0       10      198.105.254.11  21/04/16 09:46:39
Response:       0       10      104.239.213.7   21/04/16 09:46:39
```

c) Displaying host's DNS Queries

```
console>
p
Enter Hostname :
172.31.251.155
('Hostname : ', '172.31.251.155')
(' Number of URLs :', 2154)
```

d) The plot of hosts query timeline



e) The outcome of the plot command

```
console>
F
Enter Request URL :
facebook.com
172.31.148.4
172.31.250.135
172.31.151.171
172.31.247.66
```

f) Domain search

```
console>
f
Enter Resolved IP :
104.239.213.7
172.31.11.231
172.31.238.190
172.31.251.155
172.31.14.18
```

g) IP address search

Figure 4.16 BotDAD screenshots (a-g)

4.11. Limitations

This research successfully demonstrated a bot infection detection technique using DNS fingerprinting. However, the detection technique has the following limitations.

- Bots are not active throughout the day. If the bot is active during the investigating time epoch, it is very likely to be detected. However, if the bot is inactive in the investigating time epoch, it is very oblivious to be reported clean by the detection method.

- If there is a change in the IP address of the host during the time epoch, then there is a greater possibility of a host being declared clean instead of infected as the anomaly detection works on DNS fingerprint which is generated taken into consideration one hour of DNS traffic behavior.

- Our research considers the hourly traffic sample and generates a fingerprint for that period only. This can be extended to longer time intervals for detecting sophisticated and stealth bots in a network.

4.12. Summary

This chapter presented the implementation and result part of the thesis. After discussing the essential attributes of a smart DNS based botnet detection system in Chapter 2, an anomaly-based detection system named BotDAD was proposed. The system is implemented using Python as a programming language and external packages. The system is evaluated against DNS network traffic captured from Patiala campus on an hourly basis. The system was able to detect Bot infected machines in the network. The domains used for C&C by these bots were validated against online DGA domains database. The system provides high accuracy of 0.9978.

5. Deep Learning based Bot Detection

5.1. Introduction

With the invent of deep learning techniques, it is possible to achieve a high detection rate especially when the records in the dataset are high. These techniques have completely revolutionized the way we interact with the digital world in domains like speech recognition, image recognition, etc. To improve the accuracy of the BotDAD, a multi-layer neural network was used instead of Random forests classifier used in our previous work.

This chapter starts with the introduction to deep neural networks in section 5.2. Section 5.3 explains the commonly used activation functions. Section 5.4 discusses the deep learning-based DNS anomaly detector tool named DeepDAD explaining the architecture and the enhancements over the tool presented in Chapter 3. Section 5.5 explains the anomaly detection module. Implementation using two deep learning frameworks is covered in Section 5.6. Effect of hyperparameters on evaluation metrics is discussed in this section. Comparison with earlier work is covered in section 5.7. Finally, the graphical version of the tool is presented in section 5.8 which ensures ease of use in experimentation and comparison with future techniques.

5.2. Deep Learning

A deep neural network consists of an input layer, output layer, and multiple hidden layers as shown in Figure 5.1. Each layer, in turn, consists of neurons, also called a perceptron. These neurons take as input the values provided by the previous layer and uses an activation function to provide input to the next layer. The input layer is Layer 0 of the deep neural network as it contains input values that are used to train the model. The output layer is the last layer of the

deep neural network as it produces the output. Multiple hidden layers are used to train the model. Hidden layers take as input the weighted values from the previous layer and use activation function to produce output which is then passed on to the next layer. Largely hidden layers can capture complex relations between input features.

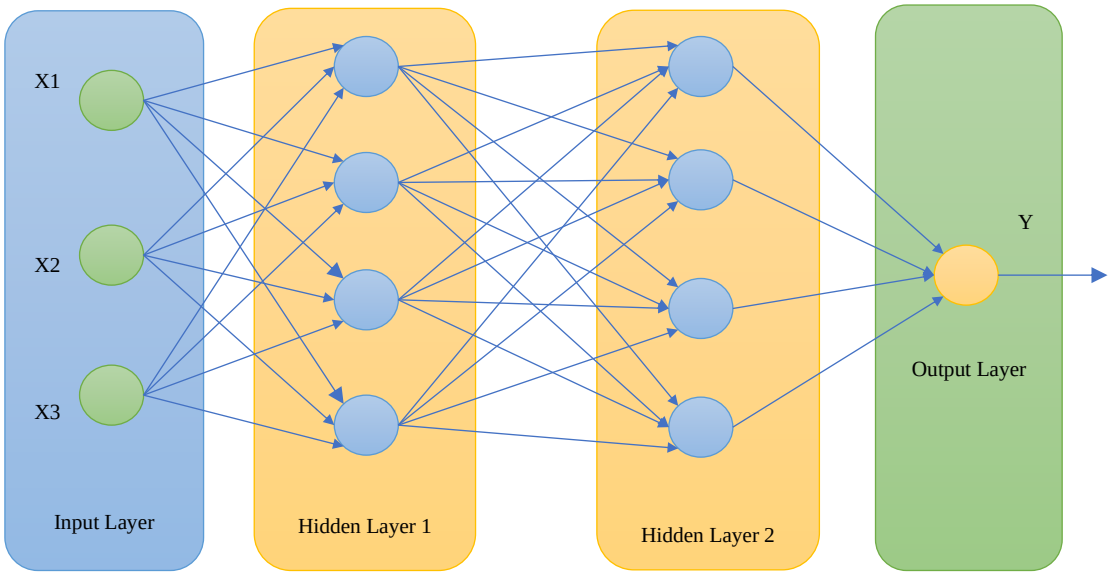
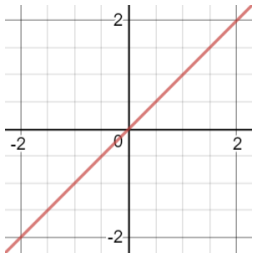
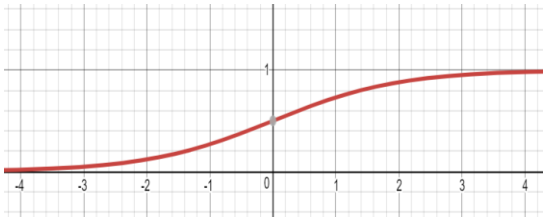
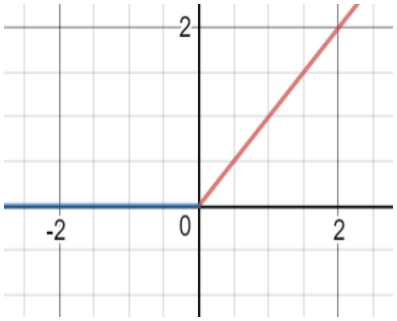
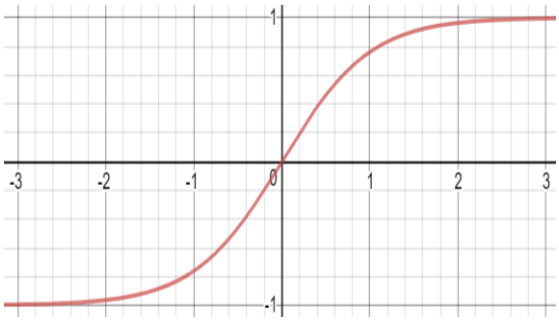


Figure 5.1 Simple Deep neural network with two hidden layers

5.3. Activation Functions

Activation functions are the functions that map the input to the neuron into the output. Commonly used activation functions include *identity*, *logistic*, *relu* (Rectified Linear Unit) and *tanh* function. The mathematical equation of these commonly used activation functions along with the graph is shown in Table 5.1.

Table 5.1 Equation and graph for common activation functions

Function	Equation	Graph
Identity	$f(x) = x$	
Logistic	$f(x) = \frac{1}{(1 + e^{-x})}$	
Relu	$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x > 0 \end{cases}$	
Tanh	$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	

5.4. DeepDAD

DeepDAD is a Deep Learning based DNS Anomaly Detection tool which considers multipoint anomaly detection and uses deep learning algorithms for machine learning. Following are the limitations of our research presented in chapter 3 and chapter 4.

Single point anomaly was used in the previous version of BotDAD which leads to the possibility of false positives. Recent machine learning techniques employing Neural Networks /Deep learning [147]–[151] have shown promising results as compared to Random Forest especially when the volume of the dataset is large.

Instead of relying on a single point anomaly for labeling DNS fingerprint as Malicious, an improved labeling technique which uses multipoint anomaly detection is implemented. This ensures in minimizing the number of false positives.

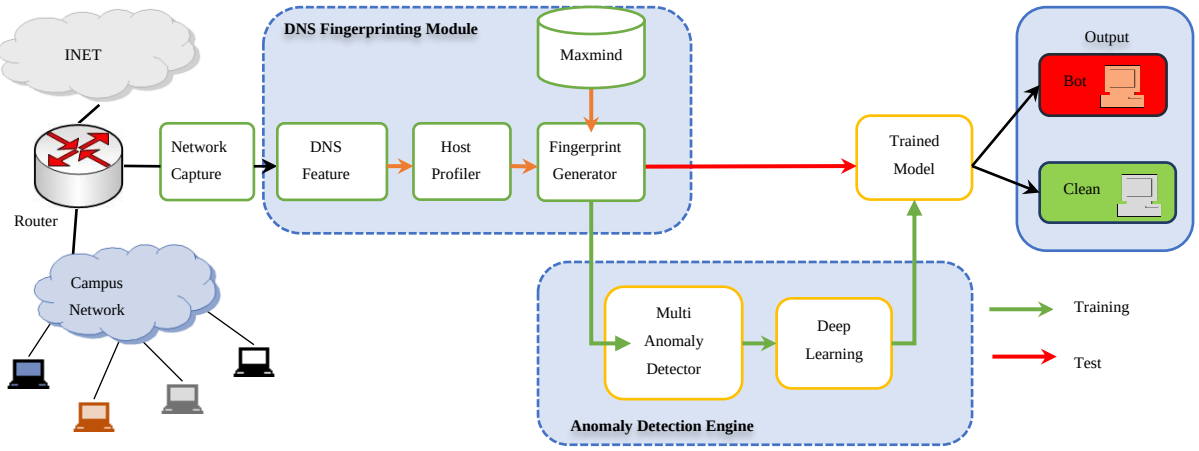


Figure 5.2 Architecture of DeepDAD

DeepDAD architecture is described in Figure 5.2 consisting of two main subsystems as 1) DNS Fingerprinting module and 2) Anomaly detection engine. The architecture is like BotDAD with two main changes in anomaly detection engine.

- DNS Fingerprint generation module produces hourly DNS fingerprint of each active device in the network by examining DNS traffic. This fingerprint is then analyzed by anomaly detection engine which classifies it into two categories i.e. *malicious* based on the presence of at least more than one anomaly and *clean* if none or one threshold is reached.

- Deep learning is first used to train the classifier and create a trained model. The trained model is then used to predict the outcome which saves time and make the next analysis better. The use of machine learning enables us to make quick detection for future DNS fingerprints with high accuracy.

DeepDAD can detect malicious DNS activities used for botnet communication with higher accuracy. DeepDAD offers a graphical user interface which allows ease of use and customization. Domains can be verified against DGArchive with the click of a button. DeepDAD also offers to export FQDN-IP mappings to Gephi for graph visualization and analysis.

5.5. Anomaly Detection

In addition to investigations made in our earlier work, an extensive focus was done on detecting anomaly for infected hosts in this work. Parameter-wise number of infected hosts was calculated as shown in Figure 5.3. In order to strengthen our anomaly detection engine, only those hosts infected with more than one anomaly were labeled as bot. The very basis for this decision was the non-malicious nature of hosts infected with the single anomaly. Moreover, it was later observed in results that infected hosts tend to trigger multiple anomaly flag.

One of the problems encountered in training the deep learning classifier was the imbalance in the number of DNS fingerprints available for bot and cleans hosts. There were 251946 benign cases and 795 malignant cases in our dataset and as such there was greater bias in the learned model. To overcome this problem, SMOTE algorithm [140] was sued which upscales the lower class or downscales the higher class to create a balanced dataset.

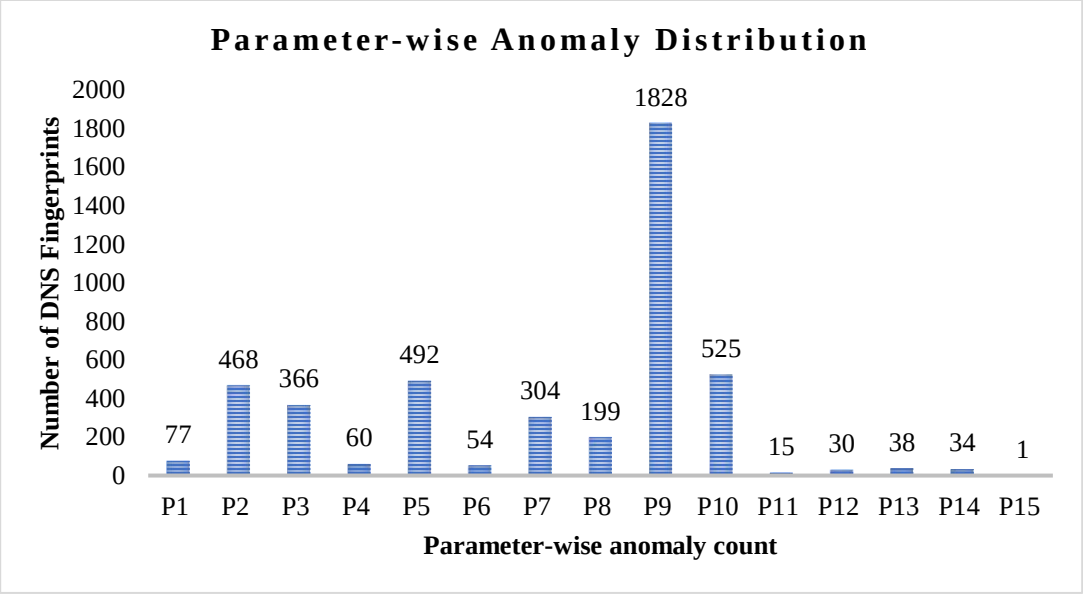


Figure 5.3 Parameter wise number of infected hosts

To investigate one step further, calculating the anomaly count of all the infected hosts was done as shown in Figure 5.4. It was observed that 2943 DNS fingerprints were infected with only one anomaly. A considerably high number of DNS fingerprints (2298) were infected with two anomalies. The number of DNS fingerprints infected with 3, 4, 5, 6 and 7 anomalies were found out to be 148, 350, 149, 43 and 6 respectively.

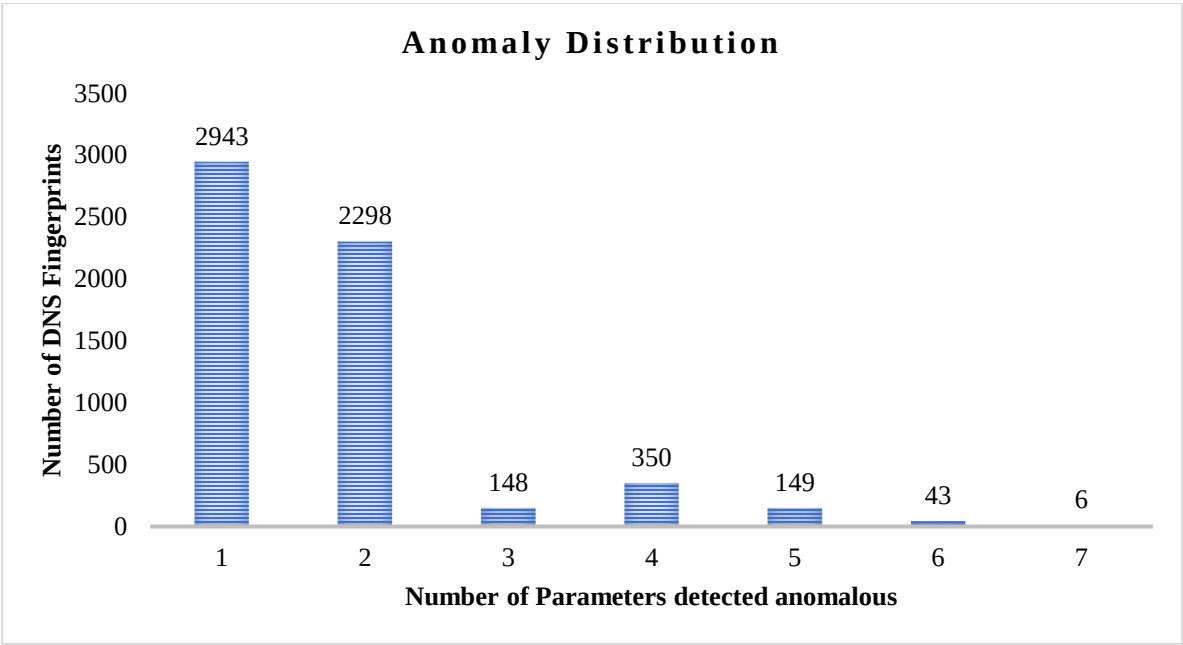


Figure 5.4 Number of Hosts infected with 1 or more anomaly

5.6. Deep Learning Frameworks

Two machine learning frameworks namely Scikit-learn [57] and TensorFlow [58] were used to train and test the model.

5.6.1. Scikit-Learn

Scikit-learn is a machine learning framework in Python. It provides various libraries for data mining and analysis. It is easily reusable and is compatible with popular python libraries like NumPy [64], SciPy [65] and Matplotlib [61]. The model was trained using *MLPClassifier* which is a Multi-Level Perceptron classifier. Following hyperparameters were tuned to check the effect on accuracy.

Activation function

Activation functions are the functions that map the input to the neuron into the output. While training the model with the above activation function, it was observed that Relu and Tanh activation function provided the maximum accuracy followed by identity function. Logistic activation function provided the minimum accuracy as shown in Figure 5.5.

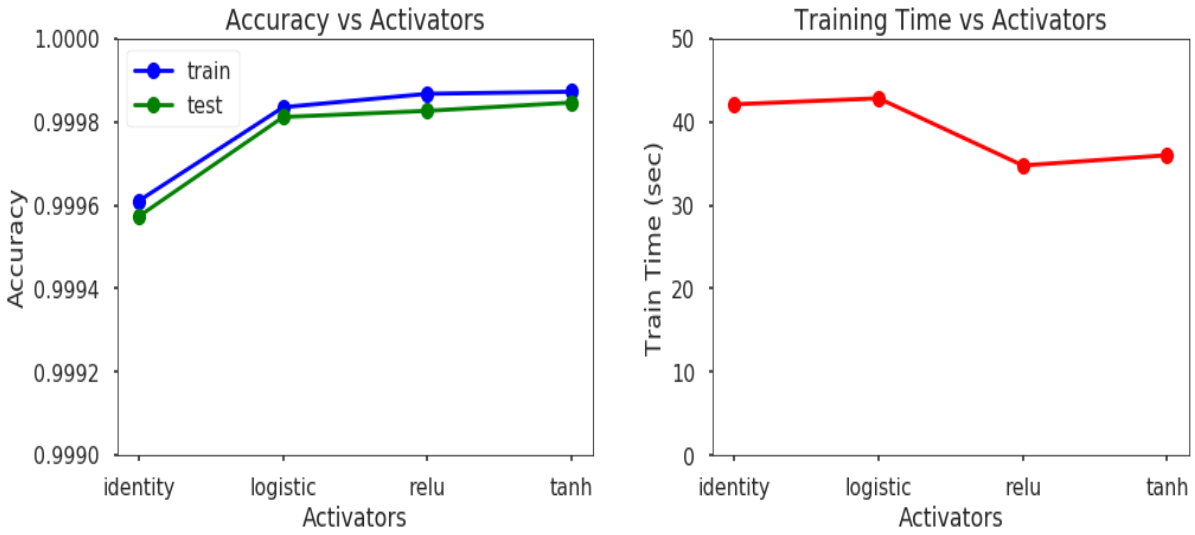


Figure 5.5 Accuracy and training time effect with activation function

Number of hidden layers

Hidden layers take as input the weighted values from the previous layer and use activation function to produce output. Largely hidden layers can capture complex relations between input features. The number of hidden parameters is another hyperparameter which has a greater influence on the accuracy of the model and as such needs to be tuned to get the optimum result. With the increase in the hidden layers, the accuracy of the model increases leading to a similar increase in training time. However, accuracy reaches its optimum when the number of hidden layers reaches 5. The effect of the number of hidden layers versus accuracy is shown in Figure 5.6.

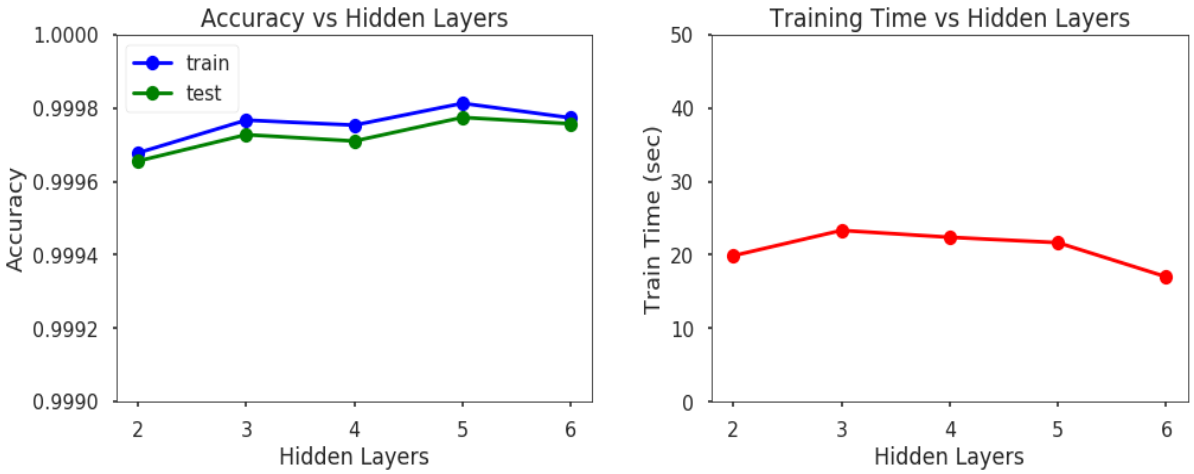


Figure 5.6 Accuracy and training time effect with the number of hidden layers

Maximum Iterations

Maximum iterations are the highest number of times the batch of input data values are used by the algorithm to train the model. A larger value may lead to overfitting of data which in turn leads to higher error values. The maximum number of iterations is a parameter that can be fine-tuned to improve the accuracy of the model. However, this parameter has a serious increasing effect on the training time. As the number of iterations increases, the accuracy of the system increases considerably until a point is reached where no further increase is found. The effect of a number of iterations versus accuracy is shown in Figure 5.7.

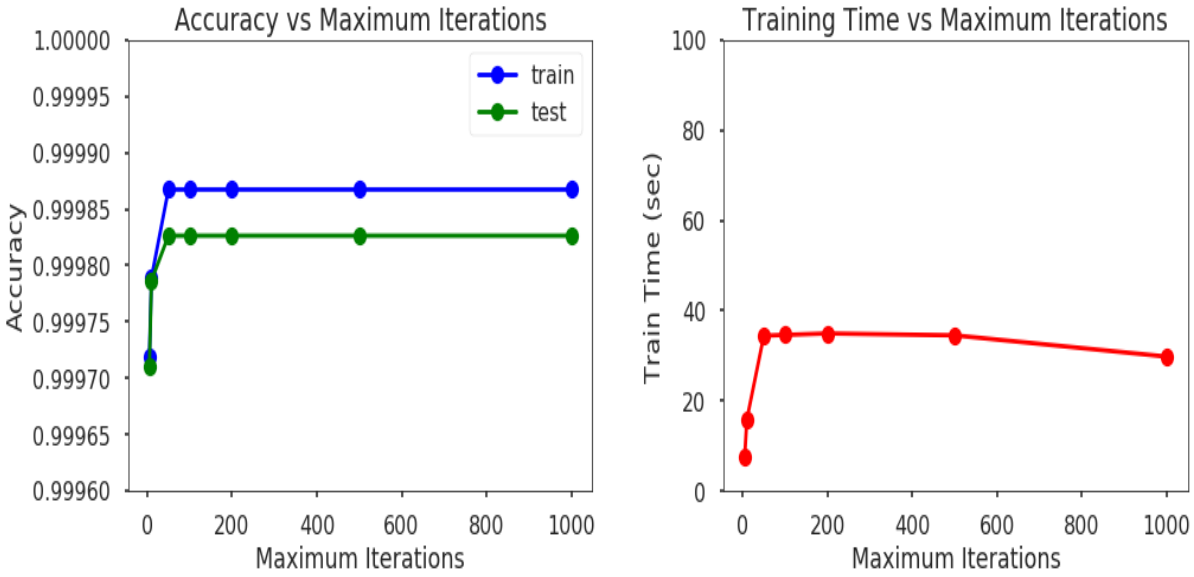


Figure 5.7 Accuracy and training time effect with maximum iterations

Confusion matrix for the model having the highest accuracy using scikit-learn is shown in Figure 5.8.

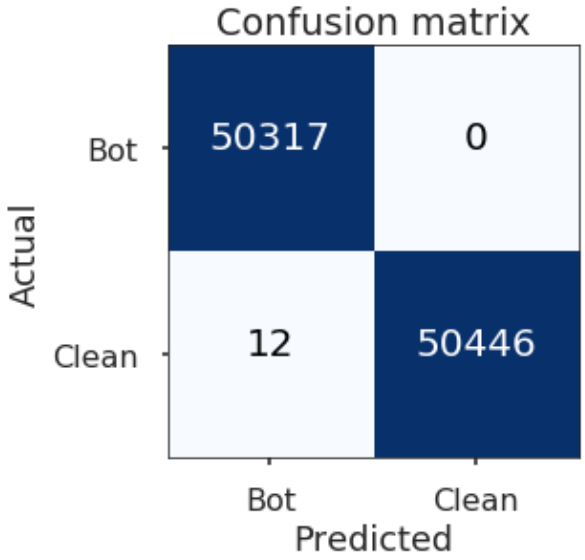


Figure 5.8 Confusion matrix for the best model using Scikit-learn

Discussion

Increasing the number of hidden layers way beyond a certain limit can lead to overfitting which in turns leads to an increase in the difference between training error and test error. Similarly, a

deep neural network with a single hidden layer can lead to underfitting wherein the model is unable to get small training error. Another important factor which can significantly reduce the error is the dataset volume. A large dataset can help train complex models which largely hidden layers thereby yielding a minimal error. However, smaller datasets are unable to train complex models.

5.6.2. Tensorflow

Tensorflow is another popular and powerful machine learning platform. Colaboratory [152] is a cloud-based jupyter notebook environment accessible from a browser that requires no installation on part of the user. Tensorflow and Colaboratory is an ideal combination for developing machine learning applications that require large computational power generally unavailable to common researchers. Hidden layers take as input the weighted values from the previous layer and use activation function to produce output. Largely hidden layers can capture complex relations between input features which are generally difficult to relate. Following hyperparameters were tuned to check the effect on accuracy.

- **Accuracy v/s Number of Neurons in layer one**

In order to test the effect of several layers and the number of neurons, accuracy was calculated while varying number of neurons in each layer. The effect of a varying number of neurons in layer one on sensitivity, specificity, and accuracy is shown in Figure 5.9. The test was performed with three layers having fixed 10 neurons in layer two and 2 neurons in layer three while the varying number of neurons in layer one. Neurons in layer one and two used a *sigmoid* activation function while layer 3 used Softmax activation function. It has been observed that varying the number of neurons in layer one has little effect on accuracy. Layer one with two neurons showed better results and as such was selected for testing layer two. Confusion matrix for the best model using 128 neurons in layer one is shown in Figure 5.10.

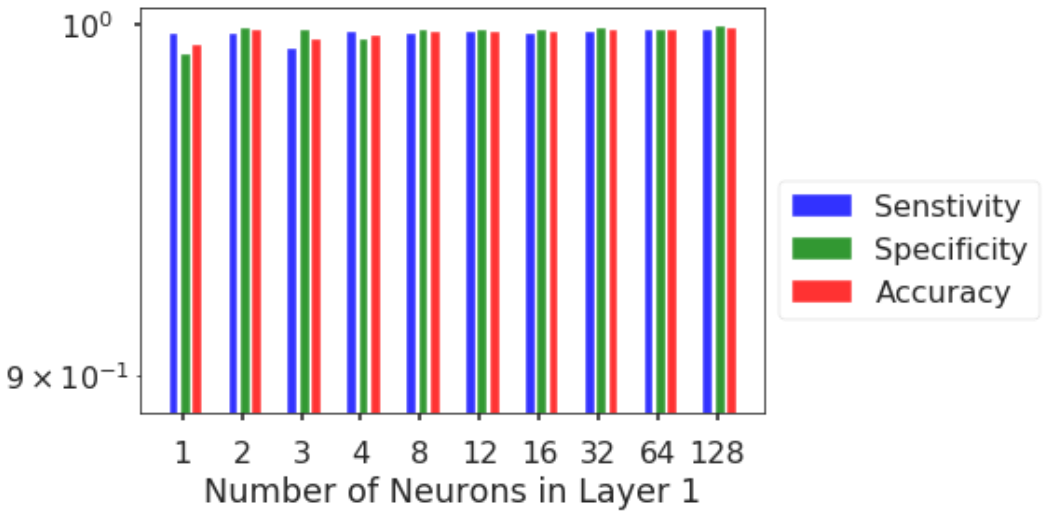


Figure 5.9 Effect of number of neurons in layer 1 on evaluation metrics

Configuration

Layer 2: *Units=32 and Activation='sigmoid'.*

Layer 3: *Units=32 and Activation='sigmoid'.*

Layer 4: *Units=2 and Activation='softmax'.*

Hyperparameters:

Optimizer='adam',

Loss='sparse_categorical_crossentropy' and

Metrics='accuracy'.

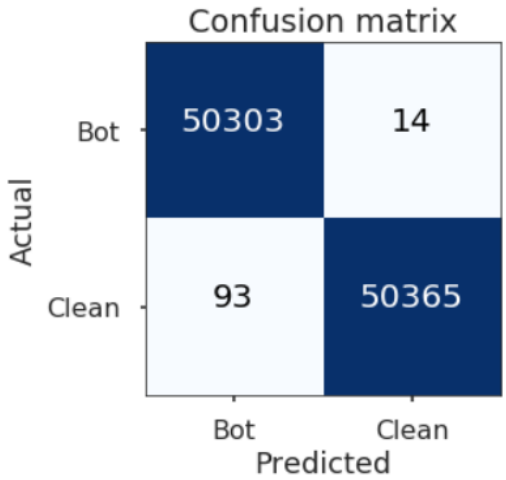


Figure 5.10 Confusion matrix for the best model

- **Accuracy v/s Number of Neurons in layer two**

Like the previous experiment, accuracy was calculated while varying number of neurons in the second layer. The effect of a varying number of neurons in layer two on accuracy is shown in Figure 5.11. The test was performed with three layers having fixed 2 neurons in layer one and 2 neurons in layer three while the varying number of neurons in layer two. Layer one and two neurons used a *sigmoid* activation function while layer 3 used Softmax activation function. As evident from the figure below, accuracy increases significantly with the increase in the number of neurons in layer two. However, it reaches its optimum at around 12 and then decreases significantly. Confusion matrix for the best model using 64 neurons in layer two is shown in Figure 5.12.

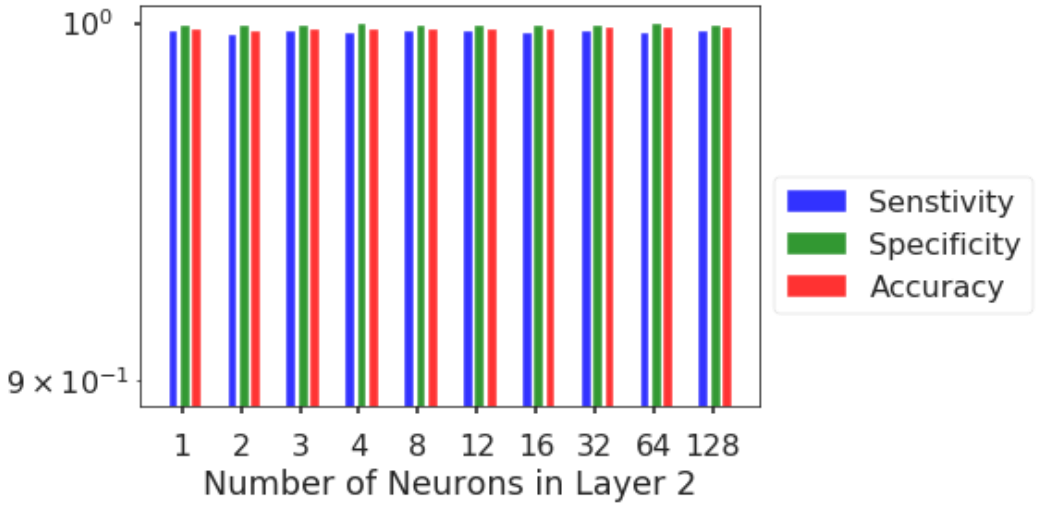


Figure 5.11 Effect of number of neurons in Layer 2 on evaluation metrics

Configuration

Layer 1: Units=32 and Activation='sigmoid'.

Layer 3: Units=32 and Activation='sigmoid'.

Layer 4: Units=2 and Activation='softmax'.

Hyperparameters:

Optimizer='adam',

Loss='sparse_categorical_crossentropy' and

Metrics='accuracy'.

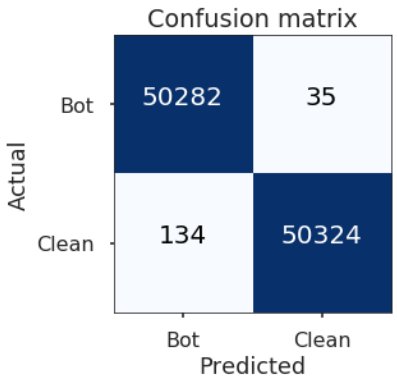


Figure 5.12 Confusion matrix for the best model

Discussion

Increasing the number of neurons in a hidden layer can result in overfitting which in turn results in a higher difference between training error and test error. Similarly, a hidden layer with a single neuron can lead to underfitting wherein the model produces large training error. Another important factor which can significantly reduce the error is the dataset volume. A large dataset can help train complex models with a higher number of neurons in each layer.

5.7. Comparison with BotDAD

To measure the effectiveness of the detection system, the following measures were calculated.

- $True\ Positive\ Rate(TPR)\ also\ called\ as\ Sensitivity = \frac{TP}{TP+FN}$
- $True\ Negative\ Rate\ (TNR)\ also\ known\ as\ Specificity = \frac{TN}{TN+FP}$
- $False\ Positive\ Rate\ (FPR) = \frac{FP}{FP+TN}$
- $False\ Negative\ Rate\ (FNR) = \frac{FN}{FN+TP}$
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

In our previous work [67] using Random Forest Classifier, a TPR of 0.996 and TNR of 0.9996 was obtained. The accuracy of the BotDAD was found to be 0.9978. Accuracy and

TPR of DeepDAD using *Scikit-Learn* were 0.9998 and 0.9997 respectively. Accuracy and TPR of DeepDAD using *Tensorflow* were 0.9989 and 0.9981 respectively. Comparison of DeepDAD using *Scikit-Learn* and *tensorflow* with BotDAD is shown in Figure 5.13.

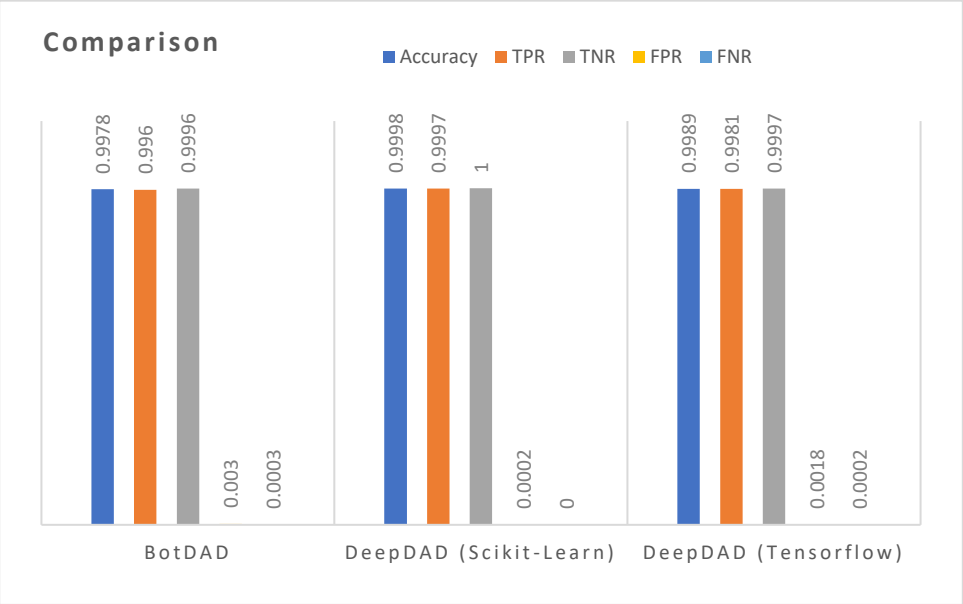


Figure 5.13 Comparison of deep learning frameworks with earlier work

5.8. DeepDAD Graphical User Interface

One of the limitations of our earlier work was the command line interface which presented a limited representation and exploration of data. In order to provide ease of access and minimal keyboard usage, an easy to use Graphical User Interface for DeepDAD tool was developed as shown in Figure 5.14. It overcomes the following limitations in the earlier version:

- Instead of entering long pcap filename and path, file open dialog can be used to select the file with ease.
- Instead of typing hostnames and domain, a list control is used which enables easy access and selection.
- Domains can be checked in DGArchive [138] for the presence/absence of DGA.

- Request and response data can be easily viewed for a given DNS Query along with the necessary details like timestamp, IP address of the DNS Server, etc.
- All infected hosts are listed in a new list to inspect the query pattern.
- Graph plot of query pattern for a given host can be accessed with the click of a button.
- Hosts FQDN-IP mapping can be exported to Gephi [63] for analysis and advanced plotting.

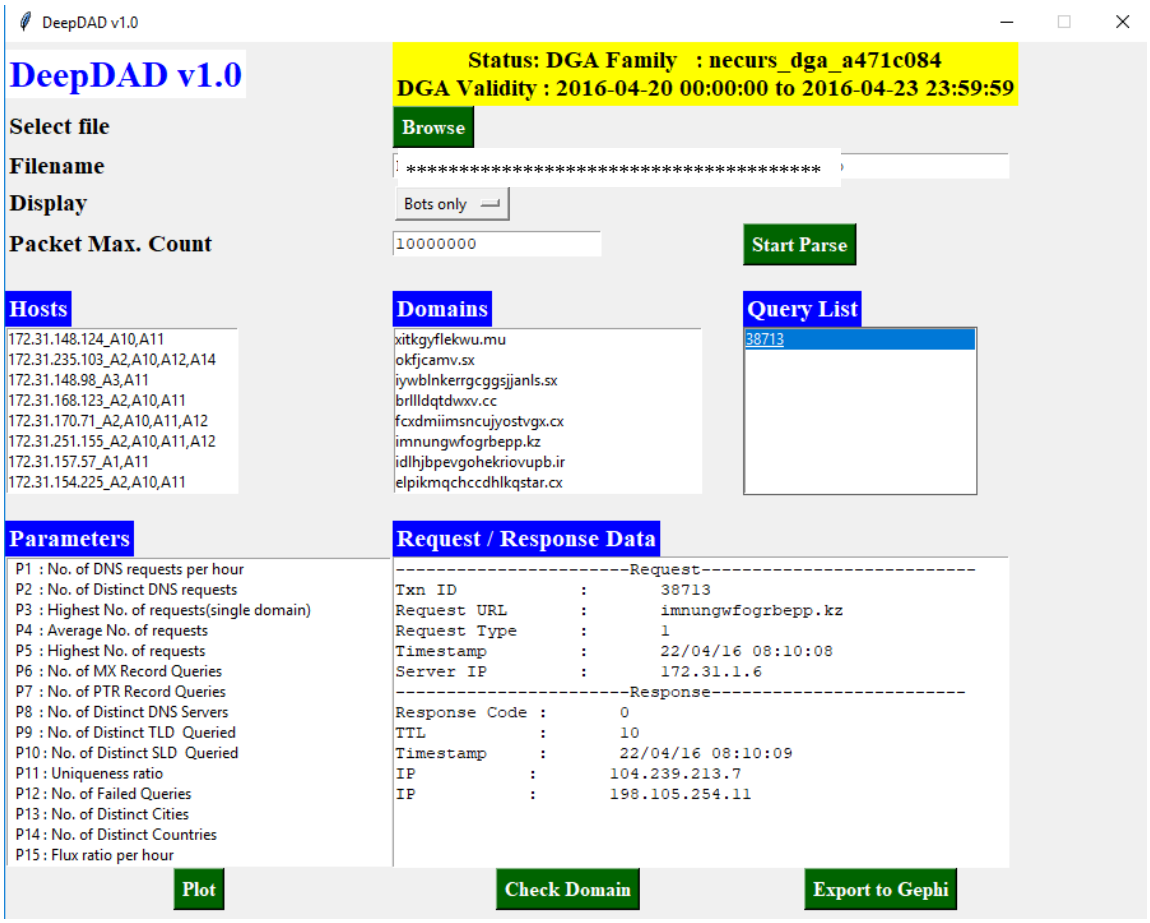


Figure 5.14 DeepDAD graphical user interface

The tool is part of the Github repository available online at [153]. Setup and installation instructions are available in the readme.md file. The malicious DNS-IP mapping can be

exported to Gephi using Gephistreamer [62] for python. Malicious DNS-IP mapping is shown in Figure 5.15.

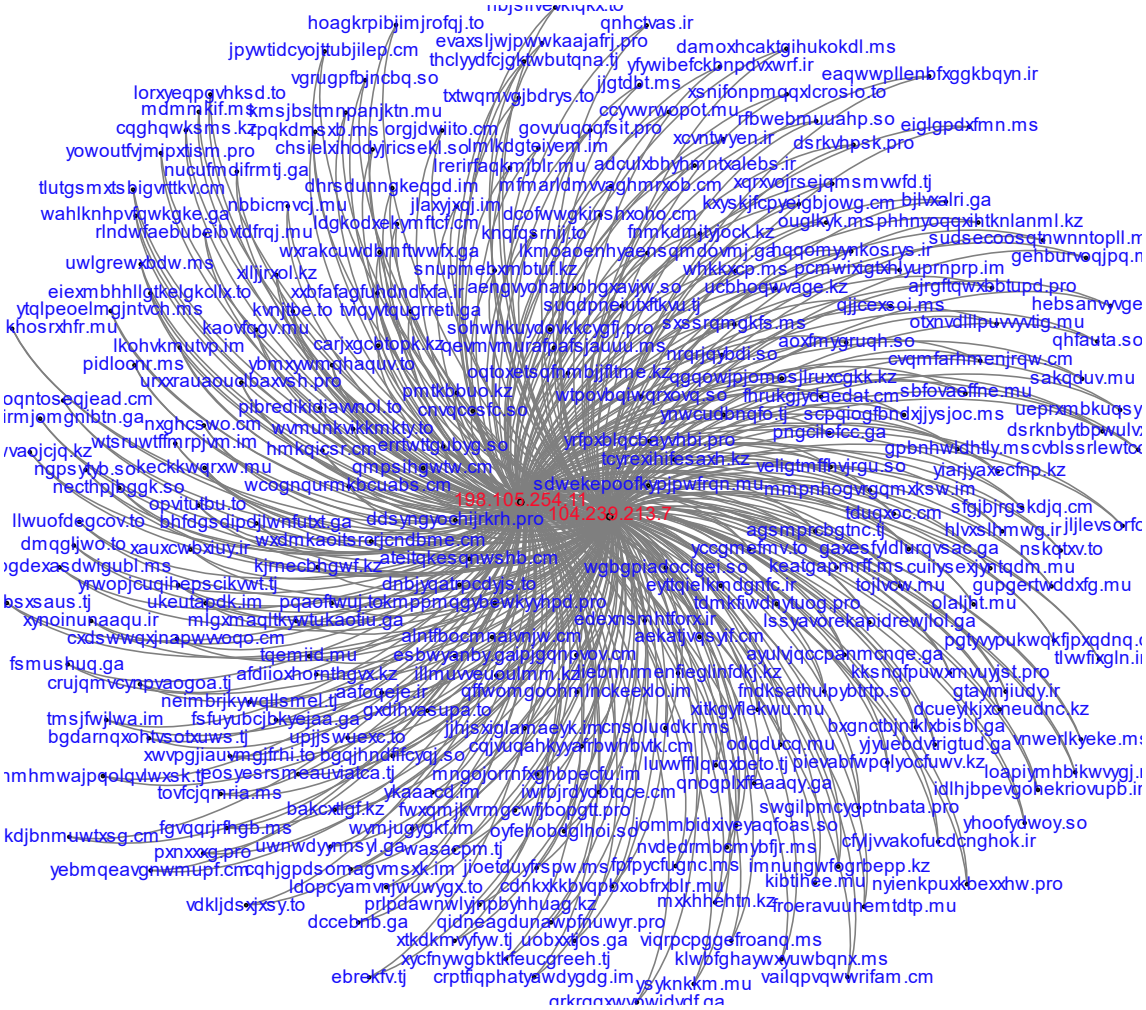


Figure 5.15 Gephi plot of malicious DNS-IP mappings

5.9. Summary

This chapter presented an improvement technique named DeepDAD which is a Deep Learning based DNS Anomaly Detection system which considers multipoint anomaly detection and uses

deep learning algorithms for machine learning. Two machine learning frameworks namely Scikit-learn and TensorFlow were used to train and test the model showing significant improvement over the results obtained using BotDAD. Finally, a graphical user interface for easy testing and comparison is presented.

6. Conclusions and Future Directions

A botnet is one of the most serious threat on the Internet. Despite many collaborative attempts by governments and law enforcement agencies, Botnet still exists and grab top headline around the world. A botnet is an open challenge for security researchers. There are many techniques available for Botnet detection but there are only a few of these are available for bot infection detection. This research presents one such technique for bot infection detection in a network. Following conclusions can be drawn from BotDAD.

- It is possible to detect Bot infection in a network by monitoring DNS traffic and looking for an anomalous pattern by creating a DNS fingerprint for each host.
- Signature-based detection is holding limited significance as far as botnet detection is concerned in view of dynamicity of botnet communication. Behavioral-based detection as such is the most viable method for detecting new and undetected bots.
- Use of whitelists and blacklist for curbing the menace of the botnet is not an alternative as DGA tend to use the domain for a limited duration and discarded thereafter.

One of the main conclusions that can be drawn from BotDAD & DeepDAD research is that multipoint anomaly detection is a better measure in detecting malicious hosts in a network. It was observed that DNS fingerprints belonging to infected hosts had more than one anomaly. Result indicate that Deep neural networks have improved results as compared to random forest. Deep neural networks tend to perform better when the number of records increases. Similarly, Deep neural networks with many layers and the number of neurons per layer have a significant improvement in the overall model when compared with traditional machine learning techniques. Our experimentation was based on random 10 days of traffic taken over a period of two months. Based on the results thus obtained, testing our model with network traffic comprising of more than 6 months of data with deep neural networks will further significantly improve results.

6.1. Future Directions

An important parameter that has a significant impact on our research is the number of active hosts in the network. For the purpose of this work, campus network traffic consisting of around 4000 hosts in peak load hours was used. It will be interesting to see how BotDAD and DeepDAD perform for ISP and corporates with more than 10,000 users in peak load hours.

Recent events have shown the use of IoT devices as bots is a very serious threat keeping in mind some of these devices like CCTV camera and wireless routers are online almost all the time and have greater bandwidth. Detecting infected IoT devices in a network is an open challenge and is bound to get broader research focus in the coming years.

Likewise, the use of a social media platform like Facebook and Twitter for botnet communication [32], [33], [154] is making it difficult to curb the menace of a botnet. Furthermore, silent or sophisticated bots which generally are designed to produce less noise and mimic normal behavior are extremely difficult to detect.

Our detection system uses anomaly-based detection technique which considers hourly hosts DNS fingerprint and attempts to find anomalous behavior which is quite different from normal machine behavior. System being passive in nature can help network administrators in detecting infected machines in the network. BotDAD and DeepDAD being a purely anomaly-based detection technique do not rely on any signature database/Blacklist/Whitelist and could detect unknown botnets. BotDAD and DeepDAD can be enhanced to include a prevention system like CleanDNS [155] in the future.

6.1.1. Additional features

This thesis work has analyzed 15 parameters while calculating DNS fingerprint. Apart from these features the following features can be analyzed in the future work.

1. “Number of TXT record queries per hour”: Some malwares use DNS TXT type queries to communicate. Number of TXT record queries can help detect such hosts using DNS TXT type queries.
2. “Average entropy of domains queried by hosts per hour”: Domain entropy represent the degree of randomness of a domain. Hosts querying DGA domains will have higher average entropy than normal hosts. Calculating average entropy of domains queried by each host will be a time-consuming process.
3. “Average Time-To-Live value in DNS response records”: Average TTL value is the average of TTL values received in DNS response records. Malicious domains have very small TTL value in DNS response records.

6.1.2. Improving ground truth

One of the challenges in using a real-world dataset is the problem of ground truth. While ground truth for dataset captured in virtual laboratory setup is easily obtained since the virtual hosts running malware is already known to the experimenter. However, when dealing with real world datasets, obtaining the ground truth is a cumbersome task. To obtain the ground truth, DGArchive was used in our study. We strongly believe that using other blacklists for obtaining the ground truth can help detect non-DGA based botnets. Other blacklists like malwaredomains.com, malwaredomainlist.com, abuse.ch, etc. can be used in future work to unearth botnets not using DGA for C&C communication.

7. Bibliography

[1] “Meaning of ‘bot’ in the English Dictionary.” [Online]. Available: http://dictionary.cambridge.org/dictionary/english/bot. [Accessed: 15-Aug-2017].

[2] “Meaning of ‘crossword puzzle’ in the English Dictionary.” [Online]. Available: https://dictionary.cambridge.org/dictionary/english/crossword-puzzle. [Accessed: 15-Aug-2017].

[3] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, “Spamming botnets: Signatures and characteristics,” *Computer Communication Review*, vol. 38, no. 4, pp. 171–182, Aug. 2008.

[4] J. Fu, P. Lin, and S. Lee, “Detecting spamming activities in a campus network using incremental learning,” *Journal of Network and Computer Applications*, vol. 43, pp. 56–65, 2014.

[5] A. West and I. Lee, “Towards the effective temporal association mining of spam blacklists,” in *ACM International Conference Proceeding Series*, 2011, no. September, pp. 73–82.

[6] P. A. R. Kumar and S. Selvakumar, “Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems,” *Computer Communications*, vol. 36, no. 3, pp. 303–319, 2013.

[7] S. McGregory, “Preparing for the next {DDoS} attack,” *Network Security*, vol. 2013, no. 5, pp. 5–6, 2013.

[8] G. Pellegrino, C. Rossow, F. J. Ryba, T. C. Schmidt, and M. Wählisch, “Cashing Out the Great Cannon? On Browser-Based DDoS Attacks and Economics,” *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, no. 1, pp. 1–36, 2015.

[9] B. Alshehry and W. Allen, “Proactive approach for the prevention of DDoS attacks in cloud computing environments,” in *Studies in Computational Intelligence*, vol. 695, 2017, pp. 119–133.

[10] H. Haddadi, “Fighting Online Click-Fraud Using Bluff Ads,” *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 2, pp. 21–25, Apr. 2010.

[11] H. Shahriar and V. K. Devendran, “Classification of Clickjacking Attacks and Detection Techniques,” *Information Security Journal: A Global Perspective*, vol. 23, no. 4–6, pp.

137–147, 2014.

[12] S. A. Alrwais, A. Gerber, C. W. Dunn, O. Spatscheck, M. Gupta, and E. Osterweil, "Dissecting ghost clicks," in *Proceedings of the 28th Annual Computer Security Applications Conference on - ACSAC '12*, 2012, p. 21.

[13] C. M. R. Haider, A. Iqbal, A. H. Rahman, and M. S. Rahman, "An ensemble learning based approach for impression fraud detection in mobile advertising," *Journal of Network and Computer Applications*, vol. 112, pp. 126–141, Jun. 2018.

[14] G. S. Oreku and F. J. Mtenzi, "Cybercrime: Concerns, challenges and opportunities," in *Studies in Computational Intelligence*, vol. 691, 2017, pp. 129–153.

[15] J. Armin, B. Thompson, and P. Kijewski, "Cybercrime Economic Costs: No Measure No Solution," in *Akhgar B., Brewster B. (eds) Combatting Cybercrime and Cyberterrorism. Advanced Sciences and Technologies for Security Applications. Springer, Cham*, 2016, pp. 135–155.

[16] R. Anderson *et al.*, "Measuring the cost of cybercrime," in *The Economics of Information Security and Privacy*, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 265–300.

[17] B. Akhgar *et al.*, "Consolidated Taxonomy and Research Roadmap for Cybercrime and Cyberterrorism," in *Akhgar B., Brewster B. (eds) Combatting Cybercrime and Cyberterrorism. Advanced Sciences and Technologies for Security Applications. Springer, Cham*, 2016, pp. 295–321.

[18] S. Furnell, D. Emm, and M. Papadaki, "The challenge of measuring cyber-dependent crimes," *Computer Fraud and Security*, vol. 2015, no. 10, pp. 5–12, 2015.

[19] T. Moore, "Phishing and the economics of e-crime," *Infosecurity*, vol. 4, no. 6, pp. 34–37, 2007.

[20] "Biggest-ever DDoS attack takes down high-profile web services," *Computer Fraud & Security*, vol. 2016, no. 11. pp. 1–3, 2016.

[21] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, and G. Vigna, "Understanding fraudulent activities in online ad exchanges," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, 2011, pp. 279–294.

[22] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, Feb. 2013.

[23] A. Karim, R. Bin Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet

detection techniques: review, future trends, and issues," *Journal of Zhejiang University: Science C*, vol. 15, no. 11, pp. 943–983, Nov. 2014.

[24] "mIRC: Internet Relay Chat Client." [Online]. Available: http://www.mirc.com/. [Accessed: 01-Jan-2016].

[25] C. Gañán, O. Cetin, and M. van Eeten, "An Empirical Analysis of ZeuS C&C Lifetime," *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '15*, pp. 97–108, 2015.

[26] "EVGENIY MIKHAILOVICH BOGACHEV - FBI." [Online]. Available: https://www.fbi.gov/wanted/cyber/evgeniy-mikhailovich-bogachev. [Accessed: 01-Jan-2016].

[27] C. Gañán, O. Cetin, and M. van Eeten, "An Empirical Analysis of ZeuS C&C Lifetime," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '15*, 2015, pp. 97–108.

[28] H. Binsalleeh *et al.*, "On the analysis of the Zeus botnet crimeware toolkit," in *PST 2010: 2010 8th International Conference on Privacy, Security and Trust*, 2010, pp. 31–38.

[29] "Major ISPs targeted in Internet of Things botnet attacks," *Network Security*, vol. 2016, no. 12. pp. 1–2, 2016.

[30] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[31] Y. Ji, Y. He, X. Jiang, J. Cao, and Q. Li, "Combating the evasion mechanisms of social bots," *Computers & Security*, vol. 58, pp. 230–249, 2016.

[32] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The Socialbot Network: When bots socialize for fame and money," in *ACM International Conference Proceeding Series*, 2011, pp. 93–102.

[33] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, "The rise of social botnets: Attacks and countermeasures," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1068–1082, Mar. 2018.

[34] N. Pantic and M. I. Husain, "Covert botnet command and control using twitter," in *ACM International Conference Proceeding Series*, 2015, vol. 7-11-Decem, pp. 171–180.

[35] P. Burghouwt, M. Spruit, and H. Sips, "Towards detection of botnet communication through social media by monitoring user activity," in *Lecture Notes in Computer Science*

(including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2011, vol. 7093 LNCS, pp. 131–143.

[36] J. Echeverria, C. Besel, and S. Zhou, "Discovery of the Twitter Bursty Botnet," pp. 145–159, Sep. 2018.

[37] R. Singel, "Hackers Use Twitter to Control Botnet," 2009. [Online]. Available: https://www.wired.com/2009/08/botnet-tweets/. [Accessed: 09-Aug-2018].

[38] C. Lumezanu and N. Feamster, "Observing common spam in tweets and email," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, 2012, pp. 441–466.

[39] N. Abokhodair, D. Yoo, and D. W. McDonald, "Dissecting a Social Botnet," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*, 2015, pp. 839–851.

[40] M. Khosroshahy, M. K. Mehmet Ali, and D. Qiu, "The SIC botnet lifecycle model: A step beyond traditional epidemiological models," *Computer Networks*, vol. 57, no. 2, pp. 404–421, 2013.

[41] R. A. Rodriguez-Gomez, G. Macia-Fernandez, and P. Garcia-Teodoro, "Survey and taxonomy of botnet research through life-cycle," *ACM Computing Surveys*, vol. 45, no. 4, pp. 1–33, Aug. 2013.

[42] McAfee Labs, "McAfee Labs Threats Report: December 2018," *Computer Fraud & Security*, 2019. [Online]. Available: www.mcafee.com/us/mcafee-labs.aspx. [Accessed: 04-Apr-2019].

[43] "The Spamhaus Project." [Online]. Available: https://www.spamhaus.org/statistics/botnet-cc/. [Accessed: 04-Apr-2019].

[44] P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 2, pp. 113–127, 2010.

[45] "Domain Names - Concepts and Facilities." [Online]. Available: https://www.ietf.org/rfc/rfc1034.txt. [Accessed: 19-May-2016].

[46] "Domain Names - Implementation and Specification." [Online]. Available: https://www.ietf.org/rfc/rfc1035.txt. [Accessed: 19-May-2016].

[47] S. Shenker, S. Alspaugh, I. Ganichev, and P. Narula, "DNS : Domain Name System," 2010. [Online]. Available: http://www.networksorcery.com/enp/protocol/dns.htm.

[Accessed: 20-Jun-2019].

[48] "The TCP/IP Guide - DNS Name Server Data Storage: Resource Records and Classes." [Online]. Available: http://www.tcpipguide.com/free/t_DNSNameServerDataStorageResourceRecordsandClasses-3.htm. [Accessed: 20-Jun-2019].

[49] N. M. Hands, B. Yang, and R. A. Hansen, "A study on botnets utilizing DNS," in *RIIT 2015 - Proceedings of the 4th Annual ACM Conference on Research in Information Technology*, 2015, pp. 23–28.

[50] Y. Fu *et al.*, "Stealthy Domain Generation Algorithms," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6. pp. 1430–1443, 2017.

[51] J. Bader, "The DGAs of Necurs," 2015. [Online]. Available: https://www.johannesbader.ch/2015/02/the-dgas-of-necurs/. [Accessed: 09-Aug-2018].

[52] "Domain Generation Algorithm (DGA)." [Online]. Available: https://resources.infosecinstitute.com/domain-generation-algorithm-dga/#gref. [Accessed: 13-Aug-2018].

[53] Millman Rene, "Russian hackers used Britney Spears' Instagram posts to control malware," *SC Media*, 2017. [Online]. Available: https://www.scmagazineuk.com/russian-hackers-used-britney-spears-instagram-posts-to-control-malware/article/667180/. [Accessed: 09-Aug-2018].

[54] "Malware finds unwitting ally in GitHub | InfoWorld." [Online]. Available: https://www.infoworld.com/article/3184399/security/malware-finds-unwitting-ally-in-github.html. [Accessed: 09-Aug-2018].

[55] "Cyber Swachhta Kendra: Home." [Online]. Available: https://www.cyberswachhtakendra.gov.in/index.html. [Accessed: 25-Jul-2019].

[56] "GitHub - mannirulz/BotDAD: Anomaly detection based on DNS traffic analysis." [Online]. Available: https://github.com/mannirulz/BotDAD. [Accessed: 15-Nov-2018].

[57] "scikit-learn: machine learning in Python," 2018. [Online]. Available: http://scikit-learn.org/stable/%0Ahttp://scikit-learn.org/stable/index.html. [Accessed: 14-Jan-2018].

[58] "TensorFlow." [Online]. Available: https://www.tensorflow.org/. [Accessed: 06-Dec-2018].

[59] "dpkt · PyPI." [Online]. Available: https://pypi.org/project/dpkt/. [Accessed: 25-Jul-

2019].

[60] “GeoIP2 Databases | MaxMind.” [Online]. Available: https://www.maxmind.com/en/geoip2-databases?pkit_lang=en. [Accessed: 05-Dec-2017].

[61] matplotlib, “Matplotlib: Python plotting — Matplotlib 3.0.3 documentation,” 2019. [Online]. Available: https://matplotlib.org/. [Accessed: 07-Mar-2019].

[62] “GephiStreamer 2.0.3 : Python Package Index.” [Online]. Available: https://pypi.python.org/pypi/GephiStreamer. [Accessed: 04-Feb-2018].

[63] Gephi.org, “Gephi - The Open Graph Viz Platform,” *Gephi*, 2017. [Online]. Available: https://gephi.org/. [Accessed: 04-Feb-2018].

[64] NumPy, “NumPy — NumPy,” *NumPy Website*, 2017. [Online]. Available: http://www.numpy.org/. [Accessed: 07-Mar-2019].

[65] SciPy developers, “SciPy.org — SciPy.org,” 2019. [Online]. Available: https://www.scipy.org/. [Accessed: 07-Mar-2019].

[66] M. Singh, M. Singh, and S. Kaur, “Issues and challenges in DNS based botnet detection: A survey,” *Computers & Security*, vol. 86, pp. 28–52, Sep. 2019.

[67] M. Singh, M. Singh, and S. Kaur, “Detecting bot-infected machines using DNS fingerprinting,” *Digital Investigation*, vol. 28, pp. 14–33, 2018.

[68] M. Feily, A. Shahrestani, and S. Ramadass, “A survey of botnet and botnet detection,” *Proceedings - 2009 3rd International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2009*, pp. 268–273, 2009.

[69] S. Soltani and S. Seno, “A survey on real world botnets and detection mechanisms,” *International Journal of …*, vol. 3, no. 2, pp. 116–127, 2014.

[70] A. H. Lashkari, S. G. Ghalebandi, and M. Reza Moradhaseli, “A Wide Survey on Botnet,” in *Communications in Computer and Information Science*, vol. 166 CCIS, no. PART 1, 2011, pp. 445–454.

[71] D. Acarali, M. Rajarajan, N. Komninos, and I. Herwono, “Survey of approaches and features for the identification of HTTP-based botnet traffic,” *Journal of Network and Computer Applications*, vol. 76, pp. 1–15, Dec. 2016.

[72] H. R. Zeidanloo, Mohammad Jorjor Zadeh Shooshtari, Payam Vahdani Amoli, M. Safari, and M. Zamani, “A taxonomy of Botnet detection techniques,” in *2010 3rd*

International Conference on Computer Science and Information Technology, 2010, pp. 158–162.

[73] G. Vormayr, T. Zseby, and J. Fabini, "Botnet Communication Patterns," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2768–2796, 2017.

[74] K. Alieyan, A. Almomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on DNS," *Neural Computing and Applications*, vol. 28, no. 7, pp. 1541–1558, Dec. 2017.

[75] X. Li, J. Wang, and X. Zhang, "Botnet detection technology based on DNS," *Future Internet*, vol. 9, no. 4, pp. 1–12, 2017.

[76] S. Asha, T. Harsha, and B. Soniya, "Analysis on botnet detection techniques," *International Conference on Research Advances in Integrated Navigation Systems, RAINS 2016*. pp. 1–4, 2016.

[77] C. A. Schiller *et al.*, "Botnet Detection: Tools and Techniques," in *Botnets*, Elsevier, 2007, pp. 133–215.

[78] K. Wang, C.-Y. Y. Huang, S.-J. J. Lin, and Y.-D. D. Lin, "A fuzzy pattern-based filtering algorithm for botnet detection," *Computer Networks*, vol. 55, no. 15, pp. 3275–3286, 2011.

[79] A. Rahim and F. T. Bin Muhaya, "Discovering the Botnet detection techniques," in *Communications in Computer and Information Science*, 2010, vol. 122 CCIS, pp. 231–235.

[80] H. S. Nair and V. E. S. E, "A Study on Botnet Detection Techniques," *International Journal of Scientific and Research Publications*, vol. 2, no. 4, pp. 2–4, 2012.

[81] M. Yahyazadeh, M. Abadi, and F. G. Marmol, "BotGrab: A negative reputation system for botnet detection," *Computers and Electrical Engineering*, vol. 41, no. C, pp. 68–85, 2015.

[82] J. Goebel and T. Holz, "Rishi: identify bot contaminated hosts by IRC nickname evaluation," *HotBots'07 Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, vol. 7, p. 8, 2007.

[83] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, Jan. 2016.

[84] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[85] E. Stinson and J. C. Mitchell, “Characterizing bots’ remote control behavior,” *Advances in Information Security*, vol. 36, pp. 45–64, 2008.

[86] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, “Active botnet probing to identify obscure command and control channels,” *Proceedings - Annual Computer Security Applications Conference, ACSAC*, pp. 241–253, 2009.

[87] Q. Yan, Y. Zheng, T. Jiang, W. Lou, and Y. T. Hou, “PeerClean: Unveiling peer-to-peer botnets through dynamic group behavior analysis,” *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pp. 316–324, 2015.

[88] B. Stone-Gross *et al.*, “Your botnet is my botnet: Analysis of a botnet takeover,” *Proceedings of the ACM Conference on Computer and Communications Security*, vol. 97, no. 3, pp. 635–647, 2009.

[89] Cisco Systems, “Snort Network Intrusion Detection & Prevention System,” *Cisco Systems*, 2019. [Online]. Available: https://www.snort.org/#get-started. [Accessed: 15-May-2018].

[90] A. Ramachandran, N. Feamster, and D. Dagon, “Revealing Botnet Membership Using DNSBL Counter-Intelligence,” in *Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2*, 2005.

[91] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, “Detecting botnets with tight command and control,” *Proceedings - Conference on Local Computer Networks, LCN*, pp. 195–202, 2006.

[92] “CRAWDAD: A Community Resource for Archiving Wireless Data At Dartmouth.” [Online]. Available: http://www.crawdad.org/. [Accessed: 01-Jan-2016].

[93] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, “BotHunter: detecting malware infection through IDS-driven dialog correlation,” *USENIX Security ’07 Proceedings of the 16th USENIX Security Symposium*, p. 12, 2007.

[94] G. Gu, J. Zhang, and W. Lee, “BotSniffer : Detecting Botnet Command and Control Channels in Network Traffic,” *Proceedings of the 15th Annual Network and Distributed System Security Symposium.*, vol. 53, no. 1, pp. 1–13, 2008.

[95] G. Gu, R. Perdisci, J. Zhang, and W. Lee, “BotMiner: Clustering Analysis of Network

Traffic for Protocol- and Structure-Independent Botnet Detection," *Proceedings of the 17th conference on Security symposium*, pp. 139–154, 2008.

[96] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," *Proceedings of the 19th USENIX Security Symposium*, pp. 273–289, 2010.

[97] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting malware domains at the upper DNS hierarchy," *Proceedings of the 20th USENIX Security Symposium*, pp. 411–426, 2011.

[98] I. Prieto, E. Magaña, D. Morató, and M. Izal, "Botnet detection based on DNS records and active probing," *SECRYPT 2011 - Proceedings of the International Conference on Security and Cryptography*. pp. 307–316, 2011.

[99] M. Stevanovic and J. M. Pedersen, "An analysis of network traffic classification for botnet detection," in *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, 2015, pp. 1–8.

[100] M. Stevanovic, J. M. Pedersen, A. D'Alconzo, S. Ruehrup, and A. Berger, "On the ground truth problem of malicious DNS traffic analysis," *Computers and Security*, vol. 55, pp. 142–158, 2015.

[101] D. Zhao *et al.*, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[102] R. Villamarín-Salomón and J. C. Brustoloni, "Identifying botnets using anomaly detection techniques applied to DNS traffic," *2008 5th IEEE Consumer Communications and Networking Conference, CCNC 2008*, no. 1, pp. 476–481, 2008.

[103] N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z. L. Zhang, "Identifying suspicious activities through DNS failure graph analysis," *Proceedings - International Conference on Network Protocols, ICNP*, pp. 144–153, 2010.

[104] S. Yadav and A. L. N. Reddy, "Winning with DNS failures: Strategies for faster botnet detection," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, vol. 96 LNICST, 2012, pp. 446–459.

[105] A. Berger, A. D'Alconzo, W. N. Gansterer, and A. Pescapé, "Mining agile DNS traffic using graph analysis for cybercrime detection," *Computer Networks*, vol. 100, pp. 28–44, 2016.

[106] H. Choi, H. Lee, and H. Kim, “BotGAD: detecting botnets by capturing group activities in network traffic,” *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, pp. 1–8, 2009.

[107] H. Choi and H. Lee, “Identifying botnets by capturing group activities in DNS traffic,” *Computer Networks*, vol. 56, no. 1, Elsevier B.V., pp. 20–33, 2012.

[108] H. Choi, H. H. Lee, H. H. Lee, and H. Kim, “Botnet detection by monitoring group activities in DNS traffic,” *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, pp. 715–720, 2007.

[109] R. Perdisci, I. Corona, D. Dagon, and W. Lee, “Detecting malicious flux service networks through passive analysis of recursive DNS traces,” *Proceedings - Annual Computer Security Applications Conference, ACSAC*, pp. 311–320, 2009.

[110] S. Y. Huang, C. H. Mao, and H. M. Lee, “Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection,” in *Proceedings of the 5th International Symposium on Information, Computer and Communications Security, ASIACCS 2010*, 2010, pp. 101–111.

[111] C. M. Chen, M. Z. Huang, and Y. H. Ou, “Detecting Web-Based Botnets with Fast-Flux Domains,” in *Smart Innovation, Systems and Technologies*, vol. 21, 2013, pp. 79–89.

[112] S. Garcia, “Malware Capture Facility Project,” 2013. [Online]. Available: https://mcfp.felk.cvut.cz/. [Accessed: 06-Sep-2017].

[113] Alexa, “Alexa Top 500 Global Ranking,” 2016. [Online]. Available: http://www.alexa.com/topsites. [Accessed: 09-Jul-2017].

[114] M. Antonakakis and R. Perdisci, “From throw-away traffic to bots: detecting the rise of DGA-based malware,” *Proceedings of the 21st USENIX Security Symposium*, p. 16, 2012.

[115] Y.-L. Zhou, Q.-S. Li, Q. Miao, and K. Yim, “DGA-Based Botnet Detection Using DNS Traffic,” *Journal of Internet Services and Information …*, vol. 3, no. 11, pp. 116–123, 2013.

[116] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, “EXPOSURE: a passive DNS analysis service to detect and report malicious domains,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, p. 14, 2014.

[117] G. Bottazzi and G. F. Italiano, “Fast Mining of Large-Scale Logs for Botnet Detection:

A Field Study," *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 1989–1996, 2015.

[118] R. Sharifnya and M. Abadi, "DFBotKiller : Domain- fl ux botnet detection based on the history of group activities and failures in DNS traf fi c," *Digital Investigation*, vol. 12, pp. 15–26, 2015.

[119] T. D. Nguyen, T. C. A. O. Dung, and L. G. Nguyen, "DGA botnet detection using collaborative filtering and density-based clustering," in *ACM International Conference Proceeding Series*, 2015, vol. 03-04-Dece, pp. 203–209.

[120] M. J. Erquiaga, C. Catania, and S. García, "Detecting DGA malware traffic through behavioral models," *2016 IEEE Biennial Congress of Argentina, ARGENCON 2016*. pp. 1–6, 2016.

[121] V. Tong and G. Nguyen, "A method for detecting DGA botnet based on semantic and cluster analysis," in *ACM International Conference Proceeding Series*, 2016, vol. 08-09-Dece, pp. 272–277.

[122] J. Kwon, J. Kim, J. Lee, H. Lee, and A. Perrig, "PsyBoG: Power spectral density analysis for detecting botnet groups," *Proceedings of the 9th IEEE International Conference on Malicious and Unwanted Software, MALCON 2014*. pp. 85–92, 2014.

[123] T. S. Wang, H. T. Lin, W. T. Cheng, and C. Y. Chen, "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis," *Computers and Security*, vol. 64, pp. 1–15, 2017.

[124] T. D. Tu, C. Guang, and L. Y. Xin, "Detecting bot-infected machines based on analyzing the similar periodic DNS queries," in *2015 International Conference on Communications, Management and Telecommunications (ComManTel)*, 2015, pp. 35–40.

[125] M. Stevanovic, J. M. Pedersen, A. D'Alconzo, and S. Ruehrup, "A method for identifying compromised clients based on DNS traffic analysis," *International Journal of Information Security*, vol. 16, no. 2, pp. 115–132, Apr. 2017.

[126] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1520–1549, 2014.

[127] M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, "The rise of 'malware':

Bibliometric analysis of malware study," *Journal of Network and Computer Applications*, vol. 75, pp. 58–76, Nov. 2016.

[128] "Stantinko botnet was undetected for at least 5 years while infecting half a million systemsSecurity Affairs." [Online]. Available: http://securityaffairs.co/wordpress/61250/malware/stantinko-botnet.html. [Accessed: 20-Aug-2017].

[129] S. K. Das *et al.*, "Chapter 4 – Evolution of Widely Spreading Worms and Countermeasures: Epidemic Theory and Application," in *Handbook on Securing Cyber-Physical Critical Infrastructure*, 2012, pp. 73–93.

[130] A. D. Rayome, "Mirai variant botnet launches IoT DDoS attacks on financial sector - TechRepublic." [Online]. Available: https://www.techrepublic.com/article/mirai-variant-botnet-launches-iot-ddos-attacks-on-financial-sector/. [Accessed: 01-Feb-2019].

[131] L. Vu, D. S. Turaga, and S. Parthasarathy, "Impact of DHCP Churn on Network Characterization," *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems*, vol. 42, no. 1, pp. 587–588, Jun. 2014.

[132] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "FluXOR: Detecting and monitoring fast-flux service networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5137 LNCS, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 186–206.

[133] M. Stevanovic, J. M. Pedersen, A. D'Alconzo, and S. Ruehrup, "A method for identifying compromised clients based on DNS traffic analysis," *International Journal of Information Security*, vol. 16, no. 2, pp. 115–132, Apr. 2017.

[134] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[135] "sklearn.preprocessing.LabelEncoder — scikit-learn 0.21.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. [Accessed: 26-Jul-2019].

[136] "sklearn.model_selection.GridSearchCV — scikit-learn 0.21.2 documentation." [Online]. Available: https://scikit-

learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. [Accessed: 26-Jul-2019].

[137] "Intel® Distribution for Python* | Intel® Software." [Online]. Available: https://software.intel.com/en-us/distribution-for-python. [Accessed: 14-Jan-2018].

[138] "DGArchive - Fraunhofer FKIE." [Online]. Available: https://dgarchive.caad.fkie.fraunhofer.de/site/. [Accessed: 22-May-2018].

[139] C. Strobl, A. L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, "Conditional variable importance for random forests," *BMC Bioinformatics*, vol. 9, no. 1, p. 307, Dec. 2008.

[140] "imblearn.over_sampling.SMOTE — imbalanced-learn 0.3.0 documentation." [Online]. Available: http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.over_sampling.SMOTE.html. [Accessed: 20-Jun-2018].

[141] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002.

[142] S. Garcia, "Malware Capture Facility Project," 2013. [Online]. Available: https://mcfp.felk.cvut.cz/publicDatasets/. [Accessed: 29-May-2018].

[143] "Index of /publicDatasets/CTU-Malware-Capture-Botnet-158-1." [Online]. Available: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-158-1/. [Accessed: 29-May-2018].

[144] "Index of /publicDatasets/CTU-Malware-Capture-Botnet-166-1." [Online]. Available: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-166-1/. [Accessed: 29-May-2018].

[145] Wireshark, "tshark - The Wireshark Network Analyzer 2.0.0," 2016. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html. [Accessed: 29-May-2018].

[146] "BotDAD/README.md at master · mannirulz/BotDAD · GitHub." [Online]. Available: https://github.com/mannirulz/BotDAD/blob/master/README.md. [Accessed: 15-Nov-2018].

[147] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," *2015 10th International Conference on Malicious*

and Unwanted Software, MALWARE 2015, pp. 11–20, 2016.

[148] C. D. McDermott, F. Majdani, and A. V. Petrovski, "Botnet Detection in the Internet of Things using Deep Learning Approaches," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2018-July, no. August, 2018.

[149] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[150] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information (Switzerland)*, vol. 10, no. 4, p. 122, Apr. 2019.

[151] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[152] Google, "Welcome to Colaboratory!" [Online]. Available: https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=5fCEDCU_qrC0. [Accessed: 07-Mar-2019].

[153] "GitHub - mannirulz/DeepDAD: DeepDAD is a GUI based Bot DNS Anomaly detection tool." [Online]. Available: https://github.com/mannirulz/DeepDAD. [Accessed: 22-Apr-2019].

[154] D. Décary-Hétu and B. Dupont, "The social network of hackers," *Global Crime*, vol. 13, no. 3, pp. 160–175, 2012.

[155] "CleanDNS Appliance download | SourceForge.net." [Online]. Available: https://sourceforge.net/projects/cleandns/. [Accessed: 19-Nov-2018].

8. List of Publications

International Journal (SCI/SCIE Indexed)

1. Singh M, Singh M, Kaur S (2018), "*Detecting bot-infected machines using DNS fingerprinting*", Digital Investigation (Elsevier), ISSN: 1742-2876, vol. 28, pp. 14–33, 2018, doi: 10.1016/j.diin.2018.12.005.
   [Impact Factor = 1.66] [**Status: Published**]

2. Singh M, Singh M, Kaur S (2019), "*Issues and Challenges in DNS based Botnet Detection: A Survey*", Computers & Security (Elsevier), ISSN: 0167-4048, vol. 86, pp. 28–52, Sep. 2019, doi: 10.1016/j.cose.2019.05.019
   [Impact Factor = 3.062] [**Status: Published**]