Spring 2017

Analysis of Periodicity in Botnets

Prathiba Nagarajan
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Information Security Commons](#)

Recommended Citation

Nagarajan, Prathiba, "Analysis of Periodicity in Botnets" (2017). *Master's Projects*. 544.
DOI: <https://doi.org/10.31979/etd.cmyz-7qw7>
https://scholarworks.sjsu.edu/etd_projects/544

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Analysis of Periodicity in Botnets

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Prathiba Nagarajan

May 2017

© 2017

Prathiba Nagarajan

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Analysis of Periodicity in Botnets

by

Prathiba Nagarajan

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2017

Dr. Mark Stamp Department of Computer Science

Dr. Thomas Austin Department of Computer Science

Dr. Melody Moh Department of Computer Science

ABSTRACT

Analysis of Periodicity in Botnets

by Prathiba Nagarajan

A botnet consists of a network of infected computers which are controlled remotely via a command and control (C&C) server. A typical botnet requires frequent communication between the C&C server and the infected nodes. Previous approaches to detecting botnets have employed various machine learning techniques, based on features extracted from network traffic. In this research, we carefully analyze the periodicity of traffic as a means for detecting a variety of botnets by applying machine learning to publicly available datasets.

ACKNOWLEDGMENTS

I would like to express gratitude to my advisor Prof. Mark Stamp for his patience, motivation and knowledge to guide this thesis and project. My thanks to the committee members Prof. Thomas Austin and Prof. Melody Moh for their encouragement. I would also like to thank Mr. Kevin Ross for sharing his knowledge about network concepts. I will forever be thankful to the Almighty, my parents, my husband, and friends for the immense support and guidance received throughout my journey.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
2	Background	3
2.1	Introduction to Botnets	3
2.1.1	Botnet Lifecycle	3
2.1.2	Command and Control	4
2.2	Network Protocols and Botnet Communication	5
2.2.1	Glossary of Terms	5
2.2.2	Network Layers and Protocols	6
2.2.3	Botnet Communication	7
2.2.4	Network Attacks Overview	8
3	Related Work	11
3.1	Static Analysis	11
3.2	Dynamic Analysis	13
3.3	Network Analysis	13
3.3.1	Malicious Botnet Detection	14
3.3.2	Non-Botnet Malware Detection	15
4	Experiments and Results	18
4.1	Protocols Used	18
4.2	Feature Selection	18
4.2.1	DNS Features	18

4.2.2	HTTP Features	21
4.2.3	TCP Features	22
4.3	Techniques	22
4.3.1	Autocorrelation	22
4.3.2	Rolling Statistics	23
4.3.3	Support Vector Machines (SVM)	23
4.3.4	Clustering	24
4.4	Tools and Languages Used	25
4.5	Dataset	26
4.6	Procedure	26
4.7	Autocorrelation Results	27
4.7.1	Normal Activity	28
4.7.2	Financial Botnets	28
4.7.3	Ransomware	32
4.7.4	DDoS and Spam Botnets	41
4.8	Rolling Statistics and SVM results	44
4.9	RFE Results	49
4.10	Clustering Results	54
4.11	Discussion	54
5	Conclusions and Future Work	62
5.1	Future Work	62
	LIST OF REFERENCES	64

LIST OF TABLES

1	Introduction to Static Analysis Literature	12
2	Introduction to Dynamic Analysis Literature	14
3	Literature of network analysis in botnets	16
4	Literature of network analysis in other malware	16
5	Feature set	20

LIST OF FIGURES

1	Lifecycle of a botnet	4
2	Various Protocols in the layers of the OSI Model	6
3	Protocol Hierarchy of Normal Activity	19
4	Protocol Hierarchy Zeus bot	19
5	Autocorrelation of a periodic signal	24
6	Autocorrelation of a aperiodic/random signal	25
7	SVM Classification of two classes of data	26
8	Autocorrelation plots of DNS Features - Normal Activity	28
9	Autocorrelation plots of HTTP Features - Normal Activity	28
10	Autocorrelation plots of TCP Features - Normal Activity	29
11	Autocorrelation plots of DNS Features - Citadel	29
12	Autocorrelation plots of HTTP Features - Citadel	30
13	Autocorrelation plots of TCP Features - Citadel	30
14	Autocorrelation plots of DNS Features - SpyEye	31
15	Autocorrelation plots of HTTP Features - SpyEye	32
16	Autocorrelation plots of TCP Features - SpyEye	33
17	Autocorrelation plots of DNS Features - Zeus	33
18	Autocorrelation plots of HTTP Features - Zeus	34
19	Autocorrelation plots of TCP Features - Zeus	35
20	Autocorrelation plots of DNS Features - Tinba	35
21	Autocorrelation plots of HTTP Features - Tinba	36

22	Autocorrelation plots of TCP Features - Tinba	36
23	Autocorrelation plots of DNS Features - Cerber	37
24	Autocorrelation plots of HTTP Features - Cerber	38
25	Autocorrelation plots of TCP Features - Cerber	39
26	Autocorrelation plots of DNS Features - Cryptolocker	39
27	Autocorrelation plots of TCP Features - Cryptolocker	40
28	Autocorrelation plots of DNS Features - Locky	40
29	Autocorrelation plots of HTTP Features - Locky	41
30	Autocorrelation plots of TCP Features - Locky	42
31	Autocorrelation plots of DNS Features - Agobot	43
32	Autocorrelation plots of HTTP Features - Agobot	44
33	Autocorrelation plots of TCP Features - Agobot	45
34	Autocorrelation plots of DNS Features - Avzhan	45
35	Autocorrelation plots of TCP Features - Avzhan	46
36	Autocorrelation plots of DNS Features - Cutwail-Pushdo	46
37	Autocorrelation plots of HTTP Features - Cutwail-Pushdo	47
38	Autocorrelation plots of TCP Features - Cutwail-Pushdo	48
39	Autocorrelation plots of DNS Features - Rbot	49
40	Autocorrelation plots of TCP Features - Rbot	50
41	Autocorrelation plots of DNS Features - Storm Variant	50
42	Autocorrelation plots of HTTP Features - Storm Variant	51
43	Autocorrelation plots of TCP Features - Storm Variant	51
44	Accuracy and AUC (DNS and HTTP) - Locky	52

45	Accuracy and AUC (TCP) - Locky	53
46	Accuracy and AUC - Tinba	56
47	Accuracy and AUC - Rbot	57
48	RFE results of Tinba	58
49	RFE results of Locky	59
50	RFE results of Rbot	60
51	Clustering results statistics	61

CHAPTER 1

Introduction

Malware is defined as a “malicious software program” [1]. It started as an annoying nuisance to computer users and has advanced to its current position of a very high security threat worldwide. Malware detectors, which are software utilities used to detect and remove malware, are the primary tools used in defense against malware. Their techniques have also grown with the growing complexity of malware and involve the combination of static and dynamic analysis for detection.

The prospect of malware as a security threat was little known when the first major Internet worm became popular in 1988 [2]. Malware writing has evolved now from being a hobby to a profession. One of the malicious security threats at present, are the Botnets. Malicious botnets are a network of infected computers which are controlled remotely, by Command and Control (C&C) servers, without the knowledge of the user. The most recent Distributed Denial of Service (DDoS) attack shut down half of the infrastructure of the Internet [3] and disrupted people’s lives to a large extent. Some longstanding botnets such as Windigo [4] have only been recently detected. Some of the other popular botnets have been taken down and others are expanding. As we have increased our dependence on the Internet from social media to the Internet of Things, our expectations of the malware detection techniques have grown. Since botnets expand and communicate using the network as the main propagation medium [5], it is a popular research subject in the field of network analysis.

Network analysis is a branch of dynamic analysis which involves detection of malware by inspecting the features of packet traffic. Previous research on network analysis as an independent feature for malware detection has shown positive results. The basic working of the botnets requires frequent communication between the C&C server(s) and the infected computers. This project inspects this frequent communi-

cation from the perspective of periodicity.

Periodicity is defined as the “recurrence of similar observations at regular intervals”. Periodic patterns could shed light on the characteristics of the underlying processes. For instance, in the case of network traffic, periodicity can be an indicator of network congestion [6]. The focus of this project is on the periodicity features of the various network protocols and their payloads of the botnet traffic across various botnets.

The report is organized as follows: Chapter 2 introduces botnets and their working, followed by a brief description of network protocols and their usage by the botnets. Chapter 3 introduces the related work to network analysis with an insight into the static and dynamic analysis. Chapter 4 deals with the details of the experiments performed and the results obtained, followed by the conclusions in Chapter 5.

CHAPTER 2

Background

This chapter provides a brief insight into the working of botnets, before diving into the rest of the project. Also, since botnet communication depends only on the existence of a network, a cursory glance at the glossary of network terms followed by the various layers and protocols is given. The chapter concludes with the preferred protocols used by the botnets and a brief description of the various types of network attacks.

2.1 Introduction to Botnets

Also known as ‘zombie army’ [7], the infected nodes are controlled by a main set of server(s) called Command and Control (C&C). The ‘controller’ servers send commands to direct the infected terminals to help with attacks such as collecting sensitive information from the user’s computers to demand a ransom, for financial theft and so on.

2.1.1 Botnet Lifecycle

Botnets follows similar steps to ‘recruit’ a vulnerable system and use it for malicious purposes, and hence is said to have a lifecycle. The components in the above lifecycle are repeated since a larger botnet results in a bigger malicious effect. The main components of the lifecycle [7] are given in Figure 1.

The C&C servers propagate the malicious software via emails, software vulnerabilities, unsecured file sharing etc. to various nodes. The nodes which are infected start communicating with the C&C server without the user’s knowledge. After infection, the botmaster may send commands to the infected computers or may be polled regularly by the bots. These commands are issued to execute tasks such as expanding the botnet by propagating the malware further or erasing all evidence and abandoning the node. The infected computers are then used for malicious coordinated attacks

Figure 1: Lifecycle of a botnet

when commanded by the C&C server.

The C&C server provides software updates to the infected systems as and when required.

2.1.2 Command and Control

The C&C servers are the main component in the survival of botnets [8]. If they are taken down or the communication channel is interrupted, the botnet army is presumed useless for malicious activities. The botnet adopts the centralized and distributed network models for its activities. The centralized model is implemented using hierarchical or star (single or multi server at the core) topologies. There is no latency in the communication to the infected terminal which helps in a more coordinated attack. But this has the single point-of-failure disadvantage. The distributed model is a peer to peer topology and constitutes multiple C&Cs which are harder to take down. But, the message delivery to other infected terminals is a complex process.

2.2 Network Protocols and Botnet Communication

This section will provide a basic overview of the terms generally used in the networking field and the protocols used by botnets for communication.

2.2.1 Glossary of Terms

Mentioned below are some of the common networking terms used in this documentation and their brief description.

- **Connection** — Connection refers to the necessary arrangements made in the form of status and related information exchange so that data transfer can take place between two nodes.
- **Packet** — Packets are the most basic units that are transferred over the network. They contain the header and payload information. Header information consists of the source and destination addresses, time stamps, and other data depending upon the network protocol. Payload consists of the data or pieces of the data that needs to be transferred, which will be combined by the protocol at the destination.
- **Protocol** — Protocols are sets of rules/standards used to define the communication method between devices. There are many different protocols used for different purposes, which will be elaborated in the next section.
- **Port** — Ports are specific addresses in the node where a logical connection takes place. Some commonly used protocols have preassigned port numbers (such as port 80 for Hypertext Transfer Protocol) where the server and client can exchange data using the defined protocol in the defined ports. The other connections are assigned ports dynamically for communication.
- **Firewall** — Firewall is a software program which inspects the traffic coming into a node at its ports. The program is set with rules which is used to check the

packets/connection and reject or accept it.

2.2.2 Network Layers and Protocols

The different layers of network communication can be explained by the historic Open Systems Interconnection (OSI) Model or Transmission Control Protocol/ Internet Protocol (TCP/IP) model. The OSI model and the various protocols in the network stack, as shown in Figure 2.

Figure 2: Various Protocols in the layers of the OSI Model

The TCP/IP model defines network layers that overlap with the OSI model. Since it has fewer layers than the OSI Model, it is more flexible and also used as a model for the internet. Hence, it is also called the Internet Suite. The *Application Layer* directly interacts with the end user and provides many services like Simple Mail Transfer Protocol (SMTP), Internet Relay Chat (IRC) etc. The *Presentation Layer*, present below the application layer, is responsible for integrating formats and presenting data in a standardized format via encryption/decryption, compression/decompression etc. The establishment and control of sessions between

end user applications is taken care of by the *Session Layer*. It uses protocols such as Remote Procedure Call (RPC) to do the needful.

Transport Layer ensures reliable communication via flow control, segmentation and error control. It uses either connection oriented protocol (TCP) or connection-less User Datagram Protocol (UDP). The *Network Layer* takes care of routing and packet forwarding. As a consequence, it manages the Quality of Service (QoS). It supports protocols such as Internet Control Message Protocol (ICMP) and Datagram Delivery Protocol (DDP) *Data Link Layer* looks into forwarding data to the adjacent nodes within a Local Area Network (LAN) or Wide Area Network (WAN). It is responsible for logical link control and media access control. *Physical Layer* is the first and lowest layer which looks into transmitting data as raw bits over the network hardware.

2.2.3 Botnet Communication

The IRC and HTTP protocols are popular among the botnets for communication. The IRC protocol facilitates communication between clients in the form of text. The chat servers act as intermediary to transfer messages between clients. Due to the scalability and flexibility in the protocol [9], it can be extended to suit the needs of the bot application. They are easy to set up and maintain but are focused to specific servers.

The HTTP protocol is also commonly used by botnets, which use HTML to communicate. The inherent simplicity of its usage comes from the fact that it can easily blend into the normal HTTP traffic. The bots exploit few tags in certain compromised sites to communicate with the infected nodes, when visited [7].

Some use TCP/IP and ICMP for their decentralized communications. Some bots use instant messaging (IM) which is similar to IRC, but contains difficulties in creating new accounts when the botnet is expanding.

There are two different ways to initiate connections, namely, Push and Pull [7]. In the Push method, the bot controller pushes the required commands via broadcast methods to the entire botnet. The advantage of this is that it is easier to coordinate attacks, but the botmaster needs a broadcast method to reach the entire network successfully. In the Pull method, the infected nodes periodically poll the botmaster for updates and commands. This can be done with the botmaster sending a query to the infected computers to start polling or a schedule with regular intervals in the botnet.

The malicious applications that the botnets were built for decide the directionality of communications required. Some botnets have bidirectional communication where information is returned for the query sent by the botmaster. Unidirectional communication have the botmasters sending commands for a task to be executed by the infected computer (such as extending the botnet).

2.2.4 Network Attacks Overview

Network attacks can be defined as an intrusion into the network infrastructure [10]. Sometimes the attack intends to silently collect data without harming the network's resources. This is called a passive attack. Active attacks have a main intention of destroying or tampering with the data and resources with or without the help of malware.

The types of potential threats to the networks is always evolving, due to which its security is an important part of the design. Some of the common network attacks overview is mentioned below.

- *Packet sniffing* works at the Ethernet frame level and is a passive attack. The packet sniffer software is generally used by network administrators to detect faults in the network infrastructure. By the hackers, it is used to spy on conver-

sations between two terminals, steal passwords, etc. by analyzing the decrypted traffic originating from the terminals. Depending on the network infrastructure, the packet sniffers may have access to the entire network or only a certain segment of it.

- *Port scanning* is used by network administrators to check the security policies. By the attackers, it is used to see the type of services running at the port and look for vulnerabilities. Port scanning is done in various ways such as FTP Bounce scan, which is using File Transfer Protocol (FTP) as a mask for scanning in order to disguise the attacker's location, and using fragmented packets to get through the filters in the firewall.
- *Spoofing* is falsifying data in order to imitate a person or software for malicious purposes. There are different areas in which this technique is used and a few of them are given below:
 - DNS spoofing — False data inserted into the DNS Server Cache to redirect users to the attacker's address of choice.
 - IP address spoofing — Creating and sending packets of data with the source IP address of a legitimate system in order to bypass firewalls.
 - Email spoofing — Falsifying the 'From:' field of an email in order to hide the actual origin of the email.
- *Denial of Service* (DoS) is used to flood the target server with a huge amount of spam and useless communications, so that the service becomes unavailable to legitimate requests. When DoS is achieved using multiple infected systems to target a particular service, it is called *Distributed Denial of Service* (DDoS) attack. One of the common types of DoS is called SYN flood attack, wherein multiple TCP requests with SYN flag set are sent to the host to enable the ports for the connection. But neither any packets with FIN flag set nor any data is

sent, due to which the ports are occupied and unavailable to other connections.

The ICMP flood attack works in a similar way with ICMP requests.

- For terminals that do not use a protocol to secure their connections, the attacker can intercept the conversations and tamper with them. In this way, the attacker can successfully impersonate one of the end points so as to disguise the tampered data as legitimate replies. This is called *Man-in-the-middle* attack.
- *Session hijacking* attack, also known as cookie hijacking, the attacker captures the cookies of an unsecured conversation between terminals, using which the attacker can gain unauthorized access into secure servers.

CHAPTER 3

Related Work

Malware detection techniques can be broadly classified into static and dynamic analysis. Static analysis explores the external features of the malicious executable, without executing it. The examples include code analysis, code or string fingerprint detection, etc. Dynamic analysis examines the malware behavior when executed in a controlled environment, such as monitoring the registry and network processes.

Network traffic analysis is part of dynamic analysis and has been used as one of the many features in machine learning for malware detection, until recently. It consists of examining the header and payload of various traffic protocols of a particular process, for any discrepancies or anomalies.

This chapter will give a brief overview of the literature using static and dynamic analysis for malware detection, followed by network analysis for botnet and other malware detection.

3.1 Static Analysis

Malware detection techniques started with static analysis. Schultz et al. [11] used the Naive Bayes algorithm on the features of Portable Executable (PE) metadata to achieve a detection of 97%, and were considered the pioneers in the usage of machine learning techniques for malware detection using static analysis. The study used the Darpa 1999 dataset, and there have been many advancements since then.

Leder et al. [12] used a procedure known as Value Set Analysis (VSA) which tracked the propagation and changes in the values of executables for detection. They aimed their study at metamorphic malware, and were able to achieve 100% accuracy. Metamorphic malware are a malware variant which have specific characteristics of replication. Austin et al. [13] identified similarities in different viruses using dueling Hidden Markov Models (HMM). They built multiple sets of models on benign and

malware files, and achieved an accuracy of 90%. The study looked at compiler and construction kits, with focus on metamorphic malware.

Using techniques from other domains, such as image processing, also broadened the knowledge of the underlying structure of malware. Nataraj et al. [14] produced a similarity measure based on texture and layout of malware files using image processing techniques. They achieved a classification accuracy of 98%. This method has resilience to obfuscation, but is very sensitive to code relocation or redundant data, which may distort the results. Rad et al. [15] used a ‘histogram of opcodes’ method to achieve 100% classification accuracy. The study used disassembled malware binaries of a small dataset of PE files, to classify malware from benign files. As the dataset increases, there may be a blur in the distinction.

A summary of the above mentioned research is given in Table 1. Moser et al. [16] showed that static analysis alone is insufficient for classification. They used simple obfuscation techniques to evade advanced signature based detection. This gave way to the research and popularization of dynamic analysis.

Table 1: Introduction to Static Analysis Literature

Authors	Method	Accuracy	Comments
Schultz et al.	Naive Bayes	97%	Pioneers of using machine learning for malware detection.
Leder et al.	VSA	100%	Focus on metamorphic malware.
Austin et al.	HMM	90%	Focus on metamorphic malware, compiler, and construction kits.
Nataraj et al.	Image Processing	98%	Sensitive to location of code.
Rad et al.	Histogram of opcodes	100%	Tests run on a small dataset.

3.2 Dynamic Analysis

Dynamic analysis helped give a peek into the malware behavior and its effect on the system processes. Table 2 summarizes this work. Bailey et al. [17] used the effect of malware behavior on the system, in terms of creation of processes and network activity, to detect malware. The study looked at all types of malware, but each of the malware was executed for less than 5 minutes. Generally, malware behavior has certain dependence on the characteristics of the external environment. They may have specific behaviors that manifest after a particular period of time, and hence time is a limitation.

Firdausi et al. [18] analyzed the behavior of malware using sparse vector models and achieved an accuracy of 93.6% with the J48 Decision Tree algorithm. Okazaki et al. [19] modeled normal behavior by ranking the frequency of system calls. They achieved improved detected over previous techniques. Blokhin et al. [20] used the similarity of system call subsequences to achieve a high degree of precision in identifying known malware family labels. The authors of the papers mentioned above, used a Virtual Machine (VM) to isolate the malware behavior, which is also a disadvantage since the malware may not be exhaustive in their behavior.

Nari et al. [21] applied the J48 decision tree algorithm on a framework which utilized the network behavior of a malware to create a behavior graph. They achieved a high classification accuracy using the network traces of the malware out in the wild, which gives a more accurate representation, with certain specific protocols only.

3.3 Network Analysis

The section is divided into malicious botnet detection and non-botnet malware detection and is summarized in Table 3 and Table 4 respectively.

Table 2: Introduction to Dynamic Analysis Literature

Authors	Method	Accuracy	Comments
Bailey et al.	Clustering of malware behavior	91%	Clustering according to the change in system processes.
Firdausi et al.	J48 decision tree on vector models	96.8%	VM to isolate malware behavior.
Nari et al.	J48 decision tree on behavior graph	57-98%	Better performance than anti virus programs compared.

3.3.1 Malicious Botnet Detection

Botnets utilize Peer-to-Peer (P2P) networks, open file sharing frameworks, and even “hit lists” to detect vulnerable IP addresses for infection among many other network propagation methods [22]. Tyagi et al. [23] focused on detecting periodicity of similar contents in a particular flow using N-Gram analysis and Deep Packet Inspection (DPI). The study clustered the similar flows and checked the similarity score using a distance metric. They also used a timing analyzer which checked the time of similar communication flows. The authors generated a synthetic dataset by setting up the Command and Control (C&C) server for a Zeus Bot, with fast flux, and achieved a 98% detection accuracy. The study uses only the HTTP protocol, and the result of application to a larger dataset is yet to be determined.

Stevanovic and Pederson [24] used more than 20 features of DNS, TCP and UDP protocols to classify traffic from 40 botnets extracted from public datasets such as ISCX [25] and ISOT botnet traffic dataset. They applied random forests classifier with the three set of features on network traces found in the wild to achieve an detection accuracy of 99%. Jin et al. [26] created a database of authoritative name server record (DNS TXT record) and IP address from the server, and used them to differentiate the traffic. They achieved a ‘hit’ of 19% on the IP address per day. The

study uses IP addresses, which may not stay constant.

Beigi et al. [25] applied a decision tree classifier on the byte based, duration based, behavior based and packet based features on approximately three botnet traces. They achieved a detection range of 75 to 99%. A larger dataset could have revealed more information regarding the accuracy. Adamov et al. [27] conducted a study about the type of files downloaded, the protocol, and encryption information along with the commands and unique characteristics of popular bots. Their focus on the anomaly details in the traffic can be used to increase the detection accuracy. This research uses a different perspective, and focuses on the details, but the generality of the method is unknown.

The strongly periodic, weakly periodic and not periodic characteristics of network features in botnets was discussed by Garcia. [4]. Depending on the flow size, he set thresholds to easily categorize the data. Using Markov chains and the transport layer protocols, they were able to achieve a F-measure of 93%.

Eslahi et al. [1] use the metrics such as periodic factor, range of absolute frequencies, and time sequence factor to determine periodicity in HTTP botnets. They define periodic factor to be a metric of time in which the corresponding activities of the botnet take place. A decision tree classification on their dataset based on the above metrics shows Neris and Zeus to be strongly periodic.

3.3.2 Non-Botnet Malware Detection

Malwares such as virus, worms, DoS malware, scanners, etc. utilize the network infrastructure for propagation and attacks. Celik et al. [28] generated more than 40 data features from the header and payload details, such as count of payload, min/mean/variance of bytes, etc. They used four types of algorithms: clustering (K-means), density (OCSVM) and least-square cost (LSAD) algorithms to generate

Table 3: Literature of network analysis in botnets

Authors	Method	Accuracy	Comments
Tyagi et al.	Cluster similar flows according to periodicity	98%	Tested on synthetic data for Zeus with fast flux.
Stevanovic and Pederson	Random Forests classification on 20 features of TCP,UDP and DNS	99%	Tested on public dataset.
Jin et al.	Database of name server records and IP addresses	19% ‘hit’	Data may not stay constant.
Beigi et al.	Decision tree classifier	75-99%	Three traces of botnets tested.

results on 16 malware families taken from public datasets. They noted a substantial decrease in detection in the recent malware families which may be connected with some traffic protocols disguised as other protocols. The number of features and techniques in the study, sheds light on the effectiveness of each.

Table 4: Literature of network analysis in other malware

Authors	Method	Results	Comments
Celik et al.	Four types of algorithms over 40 protocol features	AUC varied from 0.75 to 0.98	Disguise of one protocol as another reduced accuracy.
Li et al.	Clustering network activity of Trojans	90% accuracy	Generality of method unknown.

Iorliam et al. [29] applied Benford’s Law on TCP traffic flow size and flow difference in the malware to examine the degree of variation. They tested on multiple public datasets and observed that the flows followed the law to a satisfactory degree. One of the recent datasets they tested on had significantly lower TCP flows and hence was less representative. This is a new and focused perspective and can be explored further.

Heard et al. [30] experimented with Fourier analysis of traffic conducted to model

non-malicious polling traffic, and detect malicious traffic using a periodogram. They used the peaks in the periodogram (polling frequency) and also mapped the “time of day” distribution. They mapped the user activity and the periodicity generated by the applications, but the types of periodic user network activity were limited. Li et al. [31] focused on clustering the network activity of Trojan malware communication and achieved an accuracy of 90% with less than 3.5% false positive rate. Since this study is focused on Trojan malware, the generality of the method is unknown.

CHAPTER 4

Experiments and Results

4.1 Protocols Used

The sample protocol hierarchy of the benign (normal) and botnet packet captures are shown in Figure 3 and Figure 4 respectively. The network communication in the TCP and DNS protocols are common across both. Furthermore, the HTTP protocol under TCP has a special significance—it occupies a significant portion of the TCP communications and many botnets use HTTP as the protocol of choice. Hence, this project focuses on the features of HTTP, TCP, and DNS.

The TCP Features encompass the data of all the protocols other than HTTP under the TCP umbrella. This is to account for the botnets which use Voice over Internet Protocol (VoIP) and other such protocols [6] for their malicious activities, which fall under TCP.

4.2 Feature Selection

The features have been selected keeping techniques such as Domain Generation Algorithms (DGA) and Tunneling in mind. DGAs are used to generate different domain names periodically so that taking down the main C&Cs is difficult. Tunneling technique is used to piggyback one protocol data over another in order to avoid firewalls. One example of this is DNS tunneling. The summary of the features is given in Figure 5.

4.2.1 DNS Features

DNS is the identification system for resources connected to the internet. A DNS query translates a domain name to a IP address. There are various kinds of query and response types. Types such as ‘A’ or ‘AAAA’ are used when a new domain name needs to be translated to its IP address. Among the other types, the ‘TXT’ record is used by the server to add a non-formatted string, which can be exploited by botnets

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▼ Frame	100.0	541043	100.0	418949266	44 k	0	0	0
▼ Ethernet	100.0	541043	1.8	7574602	810	0	0	0
▼ Internet Protocol Version 4	98.8	534520	2.6	10690432	1143	0	0	0
▼ Transmission Control Protocol	89.1	482313	93.8	392802012	42 k	385500	303854393	32 k
Secure Sockets Layer	16.0	86719	43.2	181182521	19 k	83291	149457913	15 k
> Hypertext Transfer Protocol	1.2	6689	26.8	112347052	12 k	3675	2671301	285
Data	1.2	6371	0.5	2271970	243	6371	2271970	243
WebSocket	0.1	382	0.0	1524	0	381	1522	0
Malformed Packet	0.0	94	0.0	0	0	94	0	0
eXtensible Markup Language	0.0	18	0.0	1124	0	18	1124	0
High bandwidth Digital Content Protection version 2	0.0	2	0.0	26	0	2	26	0
Binary Floor Control Protocol	0.0	2	0.0	97	0	1	13	0
SPDY	0.0	1	0.0	201027	21	0	0	0
▼ User Datagram Protocol	9.5	51629	0.1	413032	44	8	0	0
Domain Name System	3.1	16826	0.3	1197545	128	16826	1197545	128
> QUIC (Quick UDP Internet Connections)	2.0	10718	0.6	2304663	246	10687	2303969	246
SKYPE	1.6	8437	0.4	1500060	160	8437	1500060	160
Simple Service Discovery Protocol	1.2	6418	0.3	1283312	137	6418	1283312	137
Dropbox LAN sync Discovery Protocol	0.8	4062	0.1	422448	45	4062	422448	45
Data	0.5	2574	0.1	261366	27	2574	261366	27
NetBIOS Name Service	0.2	1032	0.0	51600	5	1032	51600	5
Link-local Multicast Name Resolution	0.1	680	0.0	15084	1	680	15084	1
Bootstrap Protocol	0.1	414	0.0	124200	13	414	124200	13
Session Traversal Utilities for NAT	0.1	324	0.0	35372	3	324	35372	3
Multicast Domain Name System	0.0	78	0.0	14516	1	78	14516	1
Malformed Packet	0.0	32	0.0	0	0	32	0	0
▼ Wireless Access Station Session Protocol	0.0	16	0.0	13437	1	9	12195	1
Malformed Packet	0.0	7	0.0	0	0	7	0	0
▼ Real-Time Transport Protocol	0.0	6	0.0	3627	0	0	0	0
RFC 2833 RTP Event	0.0	6	0.0	3555	0	6	3555	0
NAT Port Mapping Protocol	0.0	4	0.0	48	0	4	48	0
Internet Control Message Protocol	0.1	570	0.0	58877	6	570	58877	6
Internet Group Management Protocol	0.0	8	0.0	128	0	8	128	0
Address Resolution Protocol	1.2	6523	0.0	182644	19	6523	182644	19
Text item	0.8	4411	0.0	0	0	1	0	0

Figure 3: Protocol Hierarchy of Normal Activity

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▼ Frame	100.0	2201308	100.0	891472775	1423	0	0	0
▼ Ethernet	100.0	2201307	3.5	30818298	49	0	0	0
▼ Internet Protocol Version 4	94.1	2071627	4.6	41432540	66	0	0	0
▼ Transmission Control Protocol	70.3	1548132	84.2	750497760	1198	976268	136815041	218
Secure Sockets Layer	19.5	429863	62.5	557609669	890	423531	543939991	868
> Hypertext Transfer Protocol	5.6	124204	13.4	119573116	190	62590	26318019	42
Data	1.1	24094	1.2	10632763	16	24094	10632763	16
Virtual Network Computing	0.0	23	0.0	18606	0	23	18606	0
eXtensible Markup Language	0.0	17	0.0	52681	0	17	53557	0
> Binary Floor Control Protocol	0.0	8	0.0	4092	0	0	0	0
SPDY	0.0	1	0.0	28315	0	0	0	0
Malformed Packet	0.0	1	0.0	0	0	1	0	0
High bandwidth Digital Content Protection version 2	0.0	1	0.0	524	0	1	524	0
▼ User Datagram Protocol	19.8	436490	0.4	3491920	5	141	0	0
Domain Name System	8.7	190561	0.7	6527079	10	190561	6527079	10
FP	4.2	93195	1.9	17213473	27	93195	17213473	27
NetBIOS Name Service	2.1	46590	0.3	2329500	3	46590	2329500	3
SKYPE	1.7	36571	0.8	6924860	11	36571	6924860	11
> DOF Protocol Stack	1.6	35638	0.8	6877481	10	0	0	0
Data	1.3	28640	0.6	5502936	8	28640	5502936	8
> Real-Time Transport Protocol	0.1	2655	0.0	400744	0	1236	189286	0
> uTorrent Transport Protocol	0.1	1454	0.0	283659	0	238	47387	0
AX/4000 Test Block	0.0	378	0.0	59824	0	378	59824	0
> Wireless Access Station Session Protocol	0.0	296	0.0	53617	0	1	112	0
> Real-time Transport Control Protocol	0.0	213	0.0	35156	0	58	0	0

Figure 4: Protocol Hierarchy Zeus bot

Table 5: Feature set

Protocol	Feature	Description
DNS	Query Type	The type of the query
	Response Type	The type of the response
	Frame Length	The length of the data frame
	Frame Delta Time	Time elapsed since the previous data frame
	Query Name	The second level domain (SLD)
	Response TTL	The response time to live
	Answer Count	The number of answers for the query
	Reply (Response) Code	The response code
	Response Length	The length of the response
	Length of Domain	The length of the SLD
HTTP	Count of Digits in Domain	The number of digits present in the SLD
	Bigram Query Score	The bigram score of the domain
	Request Method	The method used for request
	Content Type	The type of content
	Response Code	The status code received
	Content Length	The length of the content
	URL Length	The length of URLs
	Cache Time	The time of cache
	Cache Type	The type of cache
	Header elements	The number of header elements
	Bigram Score	The bigram score of data
	Request Interval	The interval between similar requests
TCP	Time since request	The time elapsed since the request
	Keep Alive status	The status of keep-alive flag
	Segment Length	The TCP segment length
	Flags status	The number of flags set
	iRTT	The initially estimated round trip time
	Length of connection	The connection duration
	Port	The port number used

to bypass firewalls [8]. In techniques such as tunneling, the length of the frame will be higher than normal, owing to the extra protocol data. If the length of the response is unusually high, it may point to a DDoS attack. The communication between the botmaster and the bot happens frequently and hence the ‘delta time’ feature is used to observe how often the requests and responses are being sent.

The Second Level Domain (SLD) points to the owner of the domain. Taking an example of 'google.com', 'google' is the SLD to the '.com' Top Level Domain (TLD). This feature is used to extract other features such as length of the domain, number of digits in the domain and bigram query score. The bigram score of the domain w.r.t the English language, without the digits, is calculated to measure the human readability of the domain. The score points towards the commonality of usage of the bigrams, which may not be a factor for domains supporting malware. The response code describes the success or failure of the query. These features are helpful in the case of DGAs where the length of the domain is usually longer, requires many permutations for flexibility and all of the generated domains are not generally registered,

The time to live (TTL) is a technique used to limit the validity of the DNS Record. The record is cached until the validity expires, after which it is looked up again. The TTL is used to achieve a balance between fast access to websites (high TTL) and quick changes to the domain (low TTL). The number of answers received for the query is generally the number of IP addresses through which the domain can be accessed.

4.2.2 HTTP Features

HTTP is an application level protocol that is a popular choice of communication among botnets. The request methods such as GET, POST, PUT, etc. are retrieved for the HTTP request. The type of content received may be an image, file or a HTML web page in text form. This is extracted in a numerical form.

The status code received for HTTP request, which may be a 'Success' or a 'Server not found', is important and may be higher in the case of DGAs. The length of the content received is also extracted since abnormally large content size may point to tunneling or attacks.

The time the resources could be cached and the type of cache is usually used for performance in a scenario where a user frequents a web page. The number of standard header elements being set in a response is also extracted. The interval between similar requests will support the concept of periodicity since the botnets may have similar requests for repeating activities at regular intervals.

4.2.3 TCP Features

TCP is a connection oriented protocol. The ‘keep alive’ flag is used to determine if the connection is still valid. Some malware also use TCP segments with flags set and empty payloads to keep the connection valid. The botnets set these flags in order to keep the socket busy, which helps in DDoS and zero day attacks.

The initially estimated RTT determines the base latency of the connection. The duration that the TCP connection lasts is also extracted. This is the difference between the time the ‘SYN’ and ‘SYN/‘FIN’ flags are sent.

The port number being used for the connection is used to correlate between the type of traffic generally seen at the port and traffic seen when techniques such as tunneling are used.

4.3 Techniques

There are various methods to identify the periodicity of a time domain signal. Discrete Fourier Transform (DFT) finds the periodicity of a signal by transforming it into the frequency domain. The other method, followed in this project, is called Autocorrelation, which is basically cross-correlation of a signal with itself to identify the periodicity.

4.3.1 Autocorrelation

In auto-correlation, the signal is correlated with itself and this helps to verify the presence of cycles and determine their duration. In this method, the auto-correlation

result shows peaks at various instants of time. Typically, a signal can have short and long period cycles. So we could use a peak level threshold which can be iteratively varied to identify long and short periods.

The main reason for choosing to use autocorrelation is the kind of dataset that is available for analysis. The datasets are a mixture of periodic and aperiodic samples. Implementation of autocorrelation function is straight-forward irrespective of the periodic or aperiodic nature of samples. So the results obtained are complete. On the other hand, DFT relies on a fixed sampling rate for determining the periodicity. For aperiodic sampling, DFT cannot be applied. In this case, non-uniform DFT also have to be implemented. This approximately doubles the processing time of computation. Given below, in Figures 5 and 6 are the plots of a periodic and aperiodic signal, and their autocorrelation trends respectively.

4.3.2 Rolling Statistics

The rolling statistics consists of the rolling average/mean, sum, standard deviation, and variance in this project. These statistics are applied on specific ‘windows’, and the windows are then shifted over the progress of the data. These statistics are used to smooth out the short-term noise and bring forward the trend present, if any. These statistics have been applied over the dataset in the hope that the insignificant requests/packets are smoothed out giving a clearer picture of the underlying periodicity/randomness.

4.3.3 Support Vector Machines (SVM)

In machine learning, a SVM is a supervised learning algorithm which is generally used for binary classification [32]. The algorithm uses a hyperplane to separate the data with maximum margins in a high-dimension space, if required. It uses a labeled training dataset to create a model, which is used on the test dataset to generate the

Figure 5: Autocorrelation of a periodic signal

estimated labels.

4.3.4 Clustering

Clustering is the process of grouping data that are similar or have some common characteristics. Cluster analysis falls under the category of unsupervised machine learning wherein the inherent structure of data is deduced from unlabeled data. This project has also been analyzed from the perspective of clustering to infer similarity in the grouping of botnet data in terms of periodicity. Two algorithms were looked into: k-means and Expectation-maximization (EM). Both k-means and EM work by grouping the data points according to a centralized vector. EM assigns a probability of a data point belonging to a particular cluster and produces spherical as well as

Figure 6: Autocorrelation of a aperiodic/random signal

elliptical clusters. *k*-means is a variant of the EM algorithm, assigns a data point specifically to clusters and produces spherical clusters.

4.4 Tools and Languages Used

Wireshark is a free and open source network packet analyzer for Windows and Unix. *Tshark* is its command line companion, using which the features were extracted on Windows. The version used is 2.2.1. *Microsoft Excel* is used to store the data and generate plots. *Lua* is used for writing packet dissectors, which dissect packet details. In this project, the dissectors were used to extract data and return it as features of a custom protocol. *Python* is used to manipulate data stored in the excel to generate autocorrelation and rolling statistics data. The ‘numpy’ built-in libraries

Figure 7: SVM Classification of two classes of data

were used to generate the rolling statistics, ‘sklearn’ package was used for executing SVM, generating accuracy data and Area Under the Curve (AUC) statistics.

4.5 Dataset

The datasets used are all available to the public and have been used for both botnet and benign data. The *Contagio Malware Dump* [33] consists of many malware and botnet packet captures and binaries. The *Stratosphere IPS dataset* [34] is a sister project of the Malware Capture Facility Project (MCFP), it is dedicated to the packet capture and binaries of more than 200 traces of botnets. The *ISCX dataset* [25] was created by the Information Security Centre for Excellence (ISCX) as an amalgam of its 2012 Intrusion Detection System (IDS) dataset, ISOT dataset and the MCFP dataset containing 13 traces of botnet packet captures and benign data.

4.6 Procedure

Three categories of botnets, according to their end application, were chosen taking into account the dataset availability. This was done to detect any existence of a relationship between the periodicity and their end application. The categories and

botnets chosen were:

- Financial – Bots which look into stealing credentials, banking fraud, targeting financial institutions, etc. were categorized under this label. The bots chosen were Zeus, Tinba, SpyEye, and Citadel.
- Spam and DDoS – This category comprised of botnets used to generate spam or disrupt services or both. The bots looked into were Rbot, a Storm variant, Agobot, Avzhan, and Cutwail-Pushdo.
- Ransomware – The bots which held data hostage or threatened to publish the user’s data unless a ransom was paid to a specific source fell under this category. The bots analyzed were Cerber, Locky, and Cryptolocker.

The dataset packet capture (pcap) data for the botnets and normal activity was taken from the datasets and the respective features were extracted using protocol dissectors written in Lua and executed using tshark. Protocol dissectors are used to analyze the protocol data. When a dissector is registered, if an incoming packet matches the type of the dissector, it is analyzed. In this project, the dissector is used to extract the data, manipulate it, and produce the data feature as part of a custom protocol. The custom protocols defined were ‘deepdns’, ‘deephttp’ and ‘deeptcp’, and the features extracted were defined as protocol attributes.

These data features were extracted to an excel sheet, which was then used as input to the python scripts. The python scripts were written to generate the autocorrelation, rolling statistics over 75 ‘windows’ and perform SVM on the normal as well as the malware data.

4.7 Autocorrelation Results

In the case of a large dataset, the DNS and TCP plots are shown for randomly selected specific connections with significant packet data. A basic SVM with no

options set, was used for detection.

4.7.1 Normal Activity

The data for normal traces was taken from the Stratosphere IPS dataset. Figures 8, 9, and 10 show the network analysis of normal/benign activity. As expected, the working does not show any periodicity.

Figure 8: Autocorrelation plots of DNS Features - Normal Activity

Figure 9: Autocorrelation plots of HTTP Features - Normal Activity

4.7.2 Financial Botnets

Financial botnets mostly aim to steal credentials, capture keystrokes etc. in order to inflict financial harm. The threat countered, design and working of botnets have been specifically studied previously [35] [36]. The periodicity analysis is given below.

Figure 10: Autocorrelation plots of TCP Features - Normal Activity

4.7.2.1 Citadel

The botnet traces were taken from the Contagio dataset. The DNS and TCP autocorrelation shown in Figure 11 and Figure 13 seem to have a strong periodicity, with a short term periodicity of about 11 and a long term periodicity of around 350, shown by the peaks. The HTTP plot shows a periodicity of 2 – The packet capture shows similar requests (with different data) and responses going around repeatedly.

Figure 11: Autocorrelation plots of DNS Features - Citadel

Figure 12: Autocorrelation plots of HTTP Features - Citadel

Figure 13: Autocorrelation plots of TCP Features - Citadel

4.7.2.2 SpyEye

Like Citadel, SpyEye also shows a periodicity of 2 as shown in Figure 14. The domain length seems to have a weak periodicity of around 13, but without more data, it cannot be confirmed. This trace is taken from the Contagio dataset. The Figure 15 also shows a weak periodicity of around 23 across all features, and is surrounded by noise.

Figure 14: Autocorrelation plots of DNS Features - SpyEye

Most of the TCP features in Figure 16 seem to have a periodicity of around 81 packets.

4.7.2.3 Zeus

The data acquired from the Stratosphere IPS dataset shows packet data of the infected computer with two C&Cs, which is shown in Figure 18. Both plots are slightly similar and seem to have a periodicity of 2. The DNS trend in Figure 17 and TCP trend in Figure 19 also does not seem to show any periodicity.

Figure 15: Autocorrelation plots of HTTP Features - SpyEye

4.7.2.4 Tinba

Tinba was the world's smallest malware, until Andromeda took over, at 13 Kb. The analysis of the traces taken from the Stratosphere IPS dataset is shown in Figure 20, Figure 21 and Figure 22. The DNS plots look like noise, but the domain length seems to have a undefinable trend. There seem to be peaks at around 5000 in both DNS and TCP, but nothing definite. The HTTP data was taken from a trace that has sinkholed domains (DNS servers are sometimes configured to return false IP addresses to malicious URLs, thereby preventing the connection establishment). But, the data score feature seems to have a consistent trend, but without more data, it cannot be confirmed.

4.7.3 Ransomware

Ransomware is a malware that installs stealthily on a user's computer and holds the user data hostage until a sum of money is paid. The analysis of Cerber, Cryp-

Figure 16: Autocorrelation plots of TCP Features - SpyEye

Figure 17: Autocorrelation plots of DNS Features - Zeus

tolocker and Locky is given below.

4.7.3.1 Cerber

Cerber's trace is taken from the Stratosphere IPS dataset. The DNS and TCP data shown in Figures 23 and 25 is inconclusive. However, most of the general HTTP features in Figure 24 show an almost ideal trend and periodicity of about 46.

Figure 18: Autocorrelation plots of HTTP Features - Zeus

Figure 19: Autocorrelation plots of TCP Features - Zeus

Figure 20: Autocorrelation plots of DNS Features - Tinba

Figure 21: Autocorrelation plots of HTTP Features - Tinba

Figure 22: Autocorrelation plots of TCP Features - Tinba

Figure 23: Autocorrelation plots of DNS Features - Cerber

Figure 24: Autocorrelation plots of HTTP Features - Cerber

4.7.3.2 Cryptolocker

Taken from the Contagio dataset, cryptolocker also shows a strong trend in most of the DNS features shown in Figure 26 with a short term periodicity of around 9 and

Figure 25: Autocorrelation plots of TCP Features - Cerber

a long term periodicity around 50. The TCP plot in Figure 27 trend shows a peak around 11 for most of the features, but more data will provide better confirmation.

Figure 26: Autocorrelation plots of DNS Features - Cryptolocker

Figure 27: Autocorrelation plots of TCP Features - Cryptolocker

4.7.3.3 Locky

Locky's DNS trend shown in Figure 28 and HTTP trend in Figure 29 show a periodicity of 2 in some features. The TCP plot in Figure 30 contains a weak trend at around a lag of 250, shown by a consistent peak across many of its features. The traces analyzed are taken from Contagio dataset and Stratosphere IPS dataset.

Figure 28: Autocorrelation plots of DNS Features - Locky

Figure 29: Autocorrelation plots of HTTP Features - Locky

4.7.4 DDoS and Spam Botnets

Below are mentioned the analysis of the botnets that occur in DDoS and spam application scenarios.

4.7.4.1 Agobot

Few of the features in Agobot DNS plot, shown in Figure 31 has a strong periodicity around 30. The HTTP (Figure 32) and TCP features (Figure 33) reveal weak to no periodic information. The analyzed trace was taken from the Stratosphere IPS dataset.

Figure 30: Autocorrelation plots of TCP Features - Locky

4.7.4.2 Avzhan

The network analysis of the Avzhan trace, taken from the Stratosphere IPS dataset, shows little to no periodicity in Figures 34 and 35.

4.7.4.3 Cutwail-Pushdo

The analysis of Cutwail-Pushdo (from the Contagio dataset) is shown in Figures 36, 38 and 37. Some features show a weak periodicity between 60 and 80. The general HTTP features show almost no periodicity but the features of specific connections do.

4.7.4.4 Rbot

Two traces of Rbot were used for analysis from the ISCX dataset. Both were controlled by the dataset creators, with one being labeled as for DDoS and another being applied to Port Scanning (PS). The plots differ in a few ways - some of the features in the PS application show consistent peaks and periodicity, while in the

Figure 31: Autocorrelation plots of DNS Features - Agobot

DDoS trace, there is none seen. The features in the TCP plots seem similar. This seems intuitive since port scanning is a frequent and repetitive process. We can decipher that some features hold their trends irrespective of the end application.

(2).png

Figure 32: Autocorrelation plots of HTTP Features - Agobot

4.7.4.5 Storm Variant

The network analysis of Storm revealed no information about its periodicity in DNS and TCP. HTTP in Figure 42 showed a good periodicity similar to Tinba.

4.8 Rolling Statistics and SVM results

The benign dataset used was taken from Stratosphere IPS dataset, except for Rbot, which was taken from the ISCX dataset. Since rolling statistics smooth out the noise, we would expect the AUC/Accuracy to increase. The number of malware

Figure 33: Autocorrelation plots of TCP Features - Agobot

Figure 34: Autocorrelation plots of DNS Features - Avzhan

Figure 35: Autocorrelation plots of TCP Features - Avzhan

Figure 36: Autocorrelation plots of DNS Features - Cutwail-Pushdo

and benign samples varied in each of the protocols across the botnets.

The results of Locky are given in the Figures 44 and 45. In DNS, many features maintained a 0.5 AUC and 100% accuracy over all the windows. Response length (min) sum and mean showed an improvement in AUC between the 20 and 30 window sizes, while maintaining the 100% accuracy. The accuracy of query type fell to 0 after 10 windows. Response type (standard deviation and variance) showed

Figure 37: Autocorrelation plots of HTTP Features - Cutwail-Pushdo

an improvement in AUC between the 40 and 50 windows, while maintaining 100% accuracy.

Similarly in HTTP, many features maintained a 0.5 AUC and 100% accuracy over all the windows. Header Elements (sum and standard deviation) after 45 windows and URI Length (sum and mean) after 25 windows approximately showed an improvement in AUC while maintaining the accuracy.

In TCP, the port and flags feature stand out among all the other features maintaining a 0.5 AUC and 100% accuracy. The flags feature deteriorates while the port

Figure 38: Autocorrelation plots of TCP Features - Cutwail-Pushdo

feature improves almost linearly over as the window sizes increase.

The HTTP and TCP graphs of Tinba are given in the Figure 46. In HTTP, the features such as cache time, content type, URI length, and data score improve both in accuracy and AUC to reach 100% after 10 windows. The header elements (sum and mean) AUC and accuracy maintain 100% accuracy and AUC throughout all windows. In TCP, the flags and segment length feature improve to reach 100% accuracy and AUC after 10 windows and 30 windows respectively.

The Rbot results are given in the Figure 47. In TCP, the flags (variance) increases in AUC and accuracy constantly as the windows progress. The TCP length AUC sum and mean reach 1.0 within 2nd and 8th windows, and after 10 windows respectively, while maintaining accuracy at 100%.

Figure 39: Autocorrelation plots of DNS Features - Rbot

4.9 RFE Results

Recursive Feature Elimination (RFE) is a technique used to determine which features contribute the most to the detection of the data samples. Figures show the results of RFE on the raw data feature set of Tinba, Locky and Rbot. In the case of

Figure 40: Autocorrelation plots of TCP Features - Rbot

Figure 41: Autocorrelation plots of DNS Features - Storm Variant

Figure 42: Autocorrelation plots of HTTP Features - Storm Variant

Figure 43: Autocorrelation plots of TCP Features - Storm Variant

Tinba, all the features maintained an AUC between 0.4 and 0.6 in DNS and TCP, as shown in Figure 48. In HTTP, the highest accuracy and AUC of 0.96 each is achieved with 10 and 4 features respectively. There, 7 of the 10 and 3 of the 4 features are periodic.

In Locky, for DNS, the features which had same or similar data points had the same impact as the periodic features. In HTTP and TCP, one weak periodic feature

Figure 44: Accuracy and AUC (DNS and HTTP) - Locky

Figure 45: Accuracy and AUC (TCP) - Locky

looked enough to generate 99% accuracy. But in all the cases AUC remained close to 0.5, as shown in Figure 49.

Figure 50 shows the DNS and TCP results for Rbot. In DNS, one periodic feature (query type) which was periodic in training but not in testing, maintained the accuracy and AUC. In TCP, surprisingly, all features were required to achieve the highest accuracy and AUC.

4.10 Clustering Results

In clustering, 11, 8, and 11 botnets of approximately similar data sample size were clustered in DNS, HTTP, and TCP respectively. EM showed better results when compared to k-means. In EM, as seen from the results, 16 clusters showed the best results of all the cluster sizes in DNS and TCP. In HTTP, 20 EM clusters for 8 botnets showed good results.

In DNS, all but SpyEye were analyzed, due to the limited number of data samples. The data samples of Citadel, Tinba and Cerber were equally distributed across two clusters each. Half of Tinba's samples have been clustered with the Storm variant, while Citadel and Cerber have two dedicated clusters to them. This seems to imply that Tinba may have some characteristics of a spam botnet, while Citadel and Cerber have two types of operations running under the hood, which are not clustered with the other botnets. Similar analysis can be inferred from TCP and HTTP, the results of which are shown in Figure 51.

4.11 Discussion

The concept of periodicity in the financial botnets that are chosen, ranged from strong to weak. Some features in the DNS and TCP protocols were strong with regular intervals in Citadel and weak with possible intervals in SpyEye. The HTTP features in all the botnet data displayed a periodicity of 2, irrespective of whether the same URL was accessed or not. This points to the thought that the content of the requests and responses may be correlated. When SVM was used for the detection of the Tinba botnet, the AUC score of some of the HTTP features improved in the early windows itself, which could be associated to periodicity.

In ransomware, either DNS or HTTP data showed strong trends, while TCP was either weak or inconclusive due to limited data. The results showed a strong trend

of DNS and HTTP due to a string of ‘No such name’ responses for query resolution and bad requests in HTTP. SVM was used for the detection of the Locky botnet, and some of the DNS and HTTP features showed an improvement in the AUC in a range of specific windows, while some TCP features deteriorated.

In DDoS and spam botnets, all the protocols ranged from strong to inconclusive. The Rbot botnet, which is used for different end applications, showed starkly different periodicity features. This may point to the fact that either the similarity in the internal implementation may not necessarily depict periodicity or the inherent periodicity can be masked. During the detection of Rbot using SVM, there was some improvement in a few TCP features. The end application of the traces were different, and mostly impacted the detection.

Figure 46: Accuracy and AUC - Tinba

Figure 47: Accuracy and AUC - Rbot

Figure 48: RFE results of Tinba

Figure 49: RFE results of Locky

Figure 50: RFE results of Rbot

Protocol	Type	Clusters	Homogeneity	V-Measure	Completeness	F-Measure	Purity
DNS	K-means	6	0.34	0.45	0.65	0.017	0.36
		8	0.38	0.45	0.57	0.124	0.41
		12	0.41	0.45	0.49	0.03	0.43
		16	0.43	0.46	0.5	0.04	0.45
		20	0.44	0.46	0.47	0.019	0.47
	EM	6	0.51	0.60	0.73	0.0007	0.45
		8	0.61	0.67	0.76	0.045	0.53
		12	0.79	0.8	0.82	0.078	0.76
		16	0.82	0.8	0.78	0.003	0.82
		20	0.80	0.79	0.78	0.04	0.77
Protocol	Type	Clusters	Homogeneity	V-Measure	Completeness	F-Measure	Purity
HTTP	K-means	6	0.57	0.59	0.61	0.005	0.62
		8	0.70	0.68	0.66	0.06	0.77
		12	0.81	0.74	0.67	0.09	0.87
		16	0.84	0.74	0.67	0.08	0.88
		20	0.86	0.73	0.63	0.0	0.88
	EM	6	0.59	0.61	0.64	0.13	0.62
		8	0.75	0.72	0.69	0.20	0.80
		12	0.88	0.79	0.73	0.05	0.91
		16	0.94	0.80	0.70	0.06	0.96
		20	0.96	0.81	0.70	0.0	0.98
Protocol	Type	Clusters	Homogeneity	V-Measure	Completeness	F-Measure	Purity
TCP	K-means	6	0.15	0.22	0.39	0.02	0.25
		8	0.22	0.29	0.43	0.1	0.29
		12	0.32	0.39	0.50	0.09	0.38
		16	0.37	0.43	0.50	0.05	0.43
		20	0.41	0.43	0.47	0.03	0.45
	EM	6	0.40	0.49	0.65	0.02	0.38
		8	0.47	0.54	0.65	0.004	0.47
		12	0.65	0.69	0.75	0.088	0.64
		16	0.69	0.71	0.73	0.036	0.66
		20	0.69	0.69	0.70	0.021	0.66

Figure 51: Clustering results statistics

CHAPTER 5

Conclusions and Future Work

We looked at four financial bots, three ransomwares and five DDoS and spam bots, taken from public datasets. In order to observe the periodicity trends, we looked at the autocorrelation plots of their HTTP, TCP, and DNS data, and used rolling statistics on the raw data as input to the SVM machine learning algorithm. This was done with the goal of the underlying periodic trends improving the accuracy rate. The botnets were chosen for analysis according to their end applications, in the hope that similarities in their implementation may give rise to certain similarities in periodicities.

Periodicity was observed in many features of the bots in either/all of the three protocols. The periodicity of two stood out, with different content in some bots. In many bots, a strong periodicity stood out while in other bots, it was inconclusive. In the case of a particular bot where the end application was changed, one showed strong periodicity while the other was inconclusive. This also showed that periodicity can be masked even if the underlying implementation is similar.

The SVM detection results showed many features improving in AUC as the windows progressed, while maintaining a good accuracy while a few features deteriorated in the detection. This can be attributed to the reduction in noise.

5.1 Future Work

An entity like a bot which has a specific lifecycle will have a certain set of patterns. And periodicity analysis is a step towards it. Mentioned below are some of the areas where the analysis can be extended.

- Linux based botnets are steadily rising [37] due to the many vulnerabilities without a reliable security solution. This project used public datasets which were mainly collected on a Windows OS. The analysis of periodicity can be

extended in Linux, where the protocols are similar.

- Many botnets use encryption/encoding/compression of their data to avoid detection while using techniques such as tunneling. In this project, in the bots which used encoding/encryption, the data score feature was not considered. The periodicity in the commands and communication can be analyzed and understood better when decrypted/decoded.
- Masking of the periodicity trends in a bot with different end applications can be studied. The methods of masking and unmasking the periodicity can be looked at, which may give further insight into the vulnerabilities of a botnet.

LIST OF REFERENCES

- [1] Symantec, “What is malware and how can we prevent it?” *pctools*, accessed 2017-28-04.
- [2] G. Lawton, “Virus wars: fewer attacks, new threats,” *Computer*, vol. 35, no. 12, pp. 22–24, 2002.
- [3] W. Turton, “This is why half the internet shut down today,” *Gizmodo*, 2016, accessed 2016-11-20.
- [4] M. Epstein, “Unix botnet operation windigo steals your credentials and sends tons of spam,” *Digital Trends*, 2014, accessed 2017-28-04.
- [5] X. l. Wang, L. Y. Chen, F. Liu, and Z. m. Lei, “Analysis and modeling of the botnet propagation characteristics,” in *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, Sept 2010, pp. 1–4.
- [6] X. He, C. Papadopoulos, J. Heidemann, U. Mitra, and U. Riaz, “Remote detection of bottleneck links using spectral and statistical methods,” *Computer Networks*, vol. 53, no. 3, pp. 279–298, Feb. 2009.
- [7] N. Hachem, Y. B. Mustapha, G. G. Granadillo, and H. Debar, “Botnets: Life-cycle and taxonomy,” in *2011 Conference on Network and Information Systems Security*, May 2011, pp. 1–8.
- [8] H. Tiirmaa-Klaar, J. Gassen, E. Gerhards-Padilla, and P. Martini, *Botnets*, ser. SpringerBriefs in Cybersecurity. Springer London, 2013. [Online]. Available: https://books.google.com/books?id=mbQ_AAAAQBAJ
- [9] J. Butts and S. Shenoi, *Critical Infrastructure Protection V: 5th IFIP WG 11.10 International Conference on Critical Infrastructure Protection, Revised Selected Papers*. Springer Berlin Heidelberg, March 2011. [Online]. Available: <https://books.google.com/books?id=72WrCAAAQBAJ>
- [10] S. Z., “Welcome to the security 1:1 - part 3,” *Symantec Connect*, 2013, accessed 2017-18-04.
- [11] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, “Data mining methods for detection of new malicious executables,” in *Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001*, 2001, pp. 38–49.

- [12] F. Leder, B. Steinbock, and P. Martini, "Classification and detection of metamorphic malware using value set analysis," in *2009 4th International Conference on Malicious and Unwanted Software (MALWARE)*, Oct 2009, pp. 39–46.
- [13] T. H. Austin, E. Filiol, S. Josse, and M. Stamp, "Exploring hidden markov models for virus analysis: A semantic approach," in *2013 46th Hawaii International Conference on System Sciences*, Jan 2013, pp. 5039–5048.
- [14] Y. Bai, X. Sun, G. Sun, X. Deng, and X. Zhou, "Malware images: visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security (VizSec '11)*, 2011.
- [15] B. B. Rad, M. Masrom, and S. Ibrahim, "Opcodes histogram for classifying metamorphic portable executables malware," in *2012 International Conference on E-Learning and E-Technologies in Education (ICEEE)*, Sept 2012, pp. 209–213.
- [16] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, Dec 2007, pp. 421–430.
- [17] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in *Proceedings of the 10th international conference on recent advances in intrusion detection (RAID'07)*, 2007, pp. 178–197.
- [18] I. Firdausi, C. lim, A. Erwin, and A. S. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection," in *2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*, Dec 2010, pp. 201–203.
- [19] Y. Okazaki, I. Sato, and S. Goto, "A new intrusion detection method based on process profiling," in *Proceedings 2002 Symposium on Applications and the Internet (SAINT 2002)*, 2002, pp. 82–90.
- [20] K. Blokhin, J. Saxe, and D. Mentis, "Malware similarity identification using call graph based system call subsequence features," in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, July 2013, pp. 6–10.
- [21] S. Nari and A. A. Ghorbani, "Automated malware classification based on network behavior," in *International Conference on Computing, Networking and Communications (ICNC)*, Jan 2013.
- [22] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, and M. Karir, "A survey of botnet technology and defenses," in *2009 Cybersecurity Applications Technology Conference for Homeland Security*, March 2009, pp. 299–304.

- [23] R. Tyagi, T. Paul, B. S. Manoj, and B. Thanudas, "A novel http botnet traffic detection method," in *2015 Annual IEEE India Conference (INDICON)*, Dec 2015, pp. 1–6.
- [24] 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)*, June 2015, pp. 1–8.
- [25] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*, Oct 2014, pp. 247–255.
- [26] Y. Jin, H. Ichise, and K. Iida, "Design of detecting botnet communication by monitoring direct outbound dns queries," in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*, Nov 2015, pp. 37–41.
- [27] A. Adamov, V. Hahanov, and A. Carlsson, "Discovering new indicators for botnet traffic detection," in *Proceedings of IEEE East-West Design Test Symposium (EWDTS 2014)*, Sept 2014, pp. 1–5.
- [28] Z. B. Celik, R. J. Walls, P. McDaniel, and A. Swami, "Malware traffic detection using tamper resistant features," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, Oct 2015, pp. 330–335.
- [29] A. Iorliam, S. Tirunagari, A. T. Ho, S. Li, A. Waller, and N. Poh, "'flow size difference' can make a difference: Detecting malicious tcp network flows based on benford's law," *arXiv preprint arXiv:1609.04214*, 2016.
- [30] N. Heard, P. Rubin-Delanchy, and D. J. Lawson, "Filtering automated polling traffic in computer network flow data," in *2014 IEEE Joint Intelligence and Security Informatics Conference*, Sept 2014, pp. 268–271.
- [31] S. Li, X. Yun, Y. Zhang, Y. Pang, and T. Yin, "A novel approach of detecting trojan based on network behavior analysis," in *2012 IEEE 14th International Conference on Communication Technology*, Nov 2012, pp. 513–518.
- [32] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: <http://dx.doi.org/10.1007/BF00994018>
- [33] M. Parkour, "Collection of pcap files from malware," *Contagio Malware Dump*, 2015, accessed 2016-11-20.
- [34] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

- [35] A. K. Sood, S. Zeadally, and R. J. Enbody, “An empirical study of http-based financial botnets,” *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 236–251, March 2016.
- [36] G. Bottazzi and G. Me, *A Survey on Financial Botnets Threat*, H. Jahankhani, A. Carlile, B. Akhgar, A. Taal, A. G. Hessami, and A. Hosseini-Far, Eds. Cham: Springer International Publishing, 2015. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23276-8_15
- [37] I. Barker, “The rise of the linux botnet,” *betanews*, 2016, accessed 2017-20-04.