A COMPREHENSIVE STUDY ON ONE-WAY BACKSCATTER TRAFFIC ANALYSIS

by

Eray Balkanli

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2015

I dedicate this thesis to my loving parents, Gulay and A. Erdal Balkanli, who have encouraged me to study Master of Computer Science and supported me throughout the process. I also dedicate this study to the most special person in my life, Emel Uras, who have kept me motivated for all the time and never left my side.

Table of Contents

List of Tables

List of Figures

Abstract

Since the occurrence and variety of Distributed Denial of Service (DDoS) has dramatically increased, the discovery of DDoS signatures (rules) become very difficult for current intrusion detection mechanisms. Darknets, which refer to unallocated Internet Protocol (IP) addresses in a network, are used to collect attack traffic to reveal the potential signatures. Backscatter, a behaviour observed in darknets, is a side effect of DDoS attacks generated by victim responds to the spoofed IP addresses. This thesis explores general backscatter patterns mostly based on the major transport, network and application layer protocols and ports. A detailed evaluation expressing the performances of five different signature-based network traffic monitoring systems, namely Snort, Bro, Iatmon, Corsaro and Cisco's Adaptive Security Appliance (ASA) 5515-X, over backscatter traffic is also presented. Moreover, this thesis analyzes the performances of three machine learning techniques, namely C4.5 Decision Tree, Naive Bayes and AdaBoost.M1, in terms of the detection rate, false alarm rate, computational cost and ease of use of these techniques. Additionally, different training sets with different sizes and different feature sets are used to study the effects of training datasets and data pre-processing, respectively. Five different feature sets depending on the two well-known feature selection approaches, namely Chi-Square and Symmetrical Uncertainty, as well as the most commonly used features in the literature are included in these studies. All of the evaluations are performed on six different publicly available one-way darknet datasets collected between 2004 and 2012 by CAIDA. The results show that the attack trends in the employed datasets are important to understand the nature of DDoS traffic. Furthermore, the signatures generated by a machine learning system are robust in detecting DDoS traffic even when the training set is small and the attack trends are changing over time.

List of Abbreviations and Symbols Used

ASA Adaptive Security Appliance.

ASDM Adaptive Security Device Manager.

DDoS Distributed Denial of Service.

DNS Domain Name Server.

FN False Negative.

FNR False Negative Rate.

FP False Positive.

FPR False Positive Rate.

FTP File Transfer Protocol.

HTTP HyperText Transfer Protocol.

IANA Internet Assigned Numbers Authority.

ICMP Internet Control Message Protocol.

IDS Intrusion Detection System.

IMAP4 Internet Mail Access Protocol v4.

IP Internet Protocol.

IPsec Internet Protocol security.

IRC Internet Relay Chat.

L2TP Layer Two Tunneling Protocol.

LDAP Lightweight Directory Access Protocol.

MIB Management Information Base.

MMS Multimedia Messaging Service.

NNTP Network News Transfer Protocol.

P2P Peer-to-peer.

POP Post Office Protocol.

PPTP Point-to-Point Tunneling Protocol.

RPC Remote Procedure Call.

SMB Server Message Block.

SMTP Simple Mail Transfer Protocol.

SNMP Simple network management protocol.

SSH Secure SHell.

SSL Secure Sockets Layer.

SSP Secure Server Protocol.

TCP Transmission Control Protocol.

TN True Negative.

TP True Positive.

TTL Time-to-Live.

UDP User Datagram Protocol.

Acknowledgements

First of all, I would like to thank my supervisor, A. Nur Zincir-Heywood, for her infinitive support, guidance and encouragement. Every single comment she made and advice she gave at our progress meetings has a profound effect on this research.

Moreover, there are no words to express how grateful I am to my family and fiancee, Emel Uras, for always supporting, helping and motivating me to study hard and finalize this research.

Also, I would like to show my gratitude to my friends, Vahid Foroushani and Fariba Haddadi, for always helping me when I got stuck in any technical issues.

In addition, I appreciate my friends, Baran Tatar and Ozkan Anil Toral, who were always there to provide me their endless support and motivation.

Finally, this research is supported by the Canadian Safety and Security Program(CSSP) E-Security grant. The CSSP is led by the Defense Research and Development Canada, Centre for Security Science (CSS) on behalf of the Government of Canada and its partners across all levels of government, response and emergency management organizations, nongovernmental agencies, industry and academia. This research is conducted as a part of the Dalhousie NIMS Lab.

Chapter 1

Introduction

Distributed Denial of Service (DDoS) is a well-known network attack aiming to prevent legitimate user access to a network system by overwhelming the related server(s) with a huge volume of traffic. The increase in DDoS traffic has turned out a considerable problem since DDoS attempts have drastically increased and the attack trends become elusive. According to the report by NSFOCUS, 168,459 DDoS attacks were observed in the only first half of 2013 [43]. More recently, Sony Playstation and Xbox live gaming networks were exposed to a DDoS attack by the hacktivist group "Lizard Squad" in January, 2014 [31]. It is interesting that this group also offers to generate unlimited DDoS attacks to any service on behalf of their customers who pay 500$ for a month [26]. This indicates how easy it has become to play a role in a network attack today, even as an attacker or a victim.

Given the malicious activities such as the ones above, availability, confidentiality and integrity of network systems become significant issues. To this end, network traffic monitoring tools including integrated intrusion detection systems (IDS) are used to inspect the transmitting network packets passively to reveal the potential threats. The main purpose of an IDS is to assist a network/security analyst to obtain any suspicious packets over the network. Those tools generally use pre-defined signatures (rules) or events to identify suspicious any suspicious activity. If the pattern of an inspected packet matches with a pre-defined signature, this packet causes an alarm and the IDS alerts the human expert. Therefore, the rule sets such systems include are very critical for detecting malicious traffic while monitoring a real-time traffic. At this point, network telescopes (darknets) are used to collect malicious data to reveal the potential attack patterns.

A darknet refers to such networks consisting of valid but unused IP addresses where there is no device configured. Since no legitimate network traffic is supposed to send to these IP addresses, all the packets routed to darknets are considered as

hostile. Even though misconfiguration-based packets may be observed, malicious activities such as scans to look for vulnerable devices or DDoS packets to block a system constitute the majority of attacks in a darknet. The main goal of darknets is to enable one to explore the general attack trends by analyzing the collected malicious traffic. The revealed trends are used in developing IDSs to provide reduced occurrence of false positives while identifying suspicious network packets. Note that the number of IP addresses separated for a darknet is very important since the resolution of a darknet depends on the size of its darkspace. For instance, a darknet monitoring a /8 network (16,277,216 addresses) is likely to collect more malicious traffic than a darknet monitoring a /16 network (65,536 addresses).

A darknet can work passively or actively based on its configuration. While a passive darknet only stores all the packets routed to one of the unallocated IP addresses in the range of the darknet, active darknets also respond to these packets. Therefore, active darknets are likely to reach the real attack packets including viruses or malwares since the one who generated scans would think that a vulnerable host is found. Thus, more information about the attacker who scans for vulnerable hosts before attacking can be gathered and the attacking technique can be found out by using active darknets.

Backscatter, a side of DDoS attack, is one of the potential behaviours observed in passive darknets. In such attacks, the source address of the network packets targeted to the victim are spoofed. Thus, the victim responds to these packets coming from a spoofed IP addresses as if they are normal packets. Such traffic generated by these responses is called as the backscatter traffic. Since normal packets sent to spoofed IP addresses produce backscatter traffic, which means scanning traffic are not required to be generated before generating backscatter traffic, passive darknets are able to gather backscatter packets.

In this thesis, three publicly available backscatter datasets collected in 2004, 2006 and 2008, have been employed to reveal the common attack patterns of the backscatter traffic. Moreover, two more publicly available darknet datasets from 2008 and 2012, where one of them includes mostly backscatter traffic and the other one includes both scanning and backscatter traffic, have been employed to evaluate the performances of four well-known open source and one commercial network traffic monitoring tools,

namely Snort, Bro, Iatmon, Corsaro and Cisco ASA 5515-X. In doing so, it is aimed to study the capability of these systems in detecting attack behaviours in these real life datasets, and reveal their drawbacks. Moreover, one DDoS-based darknet traffic from 2007 and two normal anonymous network traffic traces from 2008 and 2014 in addition to the aforementioned five darknet datasets have been employed in training/testing three machine learning classifiers. These classifiers include: C4.5 Decision Tree, Naive Bayes and AdaBoost. They are used to measure their capabilities in detecting backscatter traffic and to observe if they are able to overcome the challenges faced by the network traffic monitoring tools employed. Finally, different sizes of training sets and five different feature sets are used to observe how the properties of a training set affect (if at all) the performance of a machine learning based classifier. To this end, I have employed the following five different features sets: (i) including all available 25 features, (ii) one generated by the feature extraction technique called Chi-Square [60], (iii) one generated by the feature extraction technique called Symmetrical Uncertainty [17]; (iv) one feature set based on features reported in the literature [48][30][45][33][38]; and (v) one on the patterns of the backscatter traffic identified by myself.

In short, the main contributions of this thesis include:

- What are the main characteristics of backscatter traffic and do they change over years?

- What are the performances of the current network traffic monitoring tools in analyzing network traffic against potential threats? What type of signatures (rules) they use to define suspicious packets? What are the main challenges of such tools in detecting malicious activity?

- What are the performances of the supervised machine learning approaches in detecting malicious network traffic? What is the importance of the employed training set size and the instances it includes, and the selected features while building a machine learning based classifier? How does a machine learning based classifier deal with the drawbacks of the current network traffic monitoring tools? How robust the auto-generated rules by such tools are against time-based changed attack patterns? Is this possible to detect recent attack traces

by building a machine learning based system trained by old attack traces?

Note that all the darknet traffic employed in this thesis was captured by UCSD Network Telescope [25], which is known as a passive darknet configured by CAIDA. Moreover, to the best of my knowledge these datasets are the only and the most up to date datasets available that include darknet traffic. It should also be noted here that all the performance measurements are based on the detection rates, false alarm rates, computational costs and ease of usages.

The remainder of this thesis is organized as follows. Chapter 2 presents the existing works in the literature. Chapter 3 discusses the main characteristics of the employed datasets, descriptions of the used network traffic monitoring tools and machine learning approaches as well as the selected features and the used performance metrics in detail. Chapter 4 explains my evaluations and experiences with these tools and techniques on the employed datasets. Finally, Chapter 5 draws conclusions and discusses the future work.

Chapter 2

Literature Review

There are many researches explored darknet traffic analysis. These researches can be classified into two main categories. The first category aims to monitor and analyze the darknet data using open source, in house or commercial tools to discover the major attack patterns. The second category focuses on employing machine learning classifiers to analyze the darknet data to detect attack patterns.

For the first category, Bailey et al. studied on configuring passive and active darknets and clustering a large darknet dataset collected by Internet Monitor Sensor based on unique source IP addresses [19]. Eto et al. monitored various sizes of darknets, such as /8, /16 and /24, to develop a network analysis center for tactical emergency response [37]. They also observed the correlation between the attack packets seen in the monitored darknets to explore which of them fluctuate together. Wang et al. measured host infection times and reconstruct worm infection sequences by employing well-known statistical techniques on a worm dataset provided by CAIDA via /8 network telescopes [56]. Fachkha et al. focused on studying the characteristics of the darknet traffic collected via /8 network blocks over 9 months by analyzing the used transport, network and application layer protocols [38]. They also presented 30 types of threats seen in these network blocks by using association approaches to reveal the patterns of such darknet data to help building highly accurate classification models. Pang et al. focused on analyzing the nature of darknet data collected by /8 and /19 iSinks [59] in 2004 by measuring the major application-level responders and filtering the traffic by source-connection, source-destination, source-port and source-payload based features [51]. Wustrow et al. reengineered Pang et al.s research [51] by using more recent darknet datasets to analyze how the nature of darknet traffic changed in time [58]. They analyzed protocol/port selections, traffic types, sizes etc. of recent traffic over unused /8 blocks and explored critical changes in misconfiguration traffic rate and source locations. Moore et al. analyzed different darknet traces collected over

3 years, between 2001 and 2004, to classify DDoS attacks without using the packets while performing their experiments [48]: transmitting over User Datagram Protocol (UDP), transmitting over Transmission Control Protocol (TCP) with an active RST flag, and response packets. Their approach was based on observing the successive packets generated by the same source IP addresses. They used the features "packet threshold", "attack duration" and "packet rate" during classification. Fachkha et al. extended Moore et al.'s research [48] classifying Domain Name Server (DNS) based internet-scale DDoS attacks by employing 720 GB of real darknet data collected over 3 months based on the number of packets and scanned hosts, DNS query types and requested domains [30]. Finally, Dainotti et al. focused on identifying the sources of IP-spoofed internet attacks by using source/destination IP addresses, protocols, Time-To-Live (TTL) values and the least significant bytes of source IP addresses as the most informative features [33].

For the second category, Cui-Mei presented a comprehensive system to detect DDoS attacks via Support Vector Machine technique by employing Management Information Base (MIB) statistical data collected over ten days via Simple Network Management Protocol (SNMP) agents [20]. He used the features "IPReceives", "IPDelivers" and "IPOutRequests" and reached up to 99% detection rate with an SVM model trained by 30% of the traffic and tested the unseen 70% of the remaining traffic. Noh et al. studied on filtering network traffic based on TCP flags within the TCP header of the packets to detect DDoS attacks [50]. They built three machine learning models, namely Decision Tree, Naive Bayes and CN2, by employing such data collected via web servers, and achieved 99% detection rate with their Naive Bayes model. Saad et al. achieved 97% detection rate by classifying peer-to-peer (P2P) botnet traffic via Linear Support Vector Machine technique [35]. Even thought this ratio is high, the processing time of this model is approximately 20 times slower than the other approaches in this field. Livadas et al. employed real botnet packets and chat traffic collected over Dartmouth′s wireless campus network over four months to train Decision Tree, Naive Bayes and Bayesian Network models to detect Internet Relay Chat (IRC) based attacks [34]. According to their study, the Naive Bayes achieved the highest detection rate for IRC-based attacks. Fachkha et al. used TCP flags (RST, ACK, SYN+ACK and RST+ACK) to classify DDoS traffic and session

flows to categorize scanning activities to highlight their patterns and impact features. Then, they employed Moving Average, Weighted Moving Average, Exponentional Smoothing and Linear Regression models to build a model to detect DDoS traffic over a network [29]. Lee et al. explored the drawbacks of network IDSs by employing malicious sendmail system call data, then proposed a data mining based framework that uses association techniques and audit data logs to build machine learning classifiers [57]. Their study demonstrates that the rules generated by RIPPER method achieve to detect 99% of the malicious traffic. Finally, Farid et al. targeted to decrease false positives of network IDSs by developing a preliminary system by combining Decision Tree and Naive Bayes techniques to overcome the challenges of network IDSs [44]. They employed KDD'99 dataset, which includes various attacks simulated in a military network environment, and reached up to 99.7% accuracy by using 19 features.

One of the most important drawbacks current network IDSs have is the high possibility of incorrectly categorizing legitimate traffic as attack traffic. Aforementioned researches focus on decreasing the false alarm rate of network IDSs by analyzing darknet traffic to discover its main characteristics, mostly depending on packet features, such as IP addresses, ports and protocols, in order to generate signatures for these IDSs to detect malicious traffic. These studies also focus on developing machine learning based models that automatically generate signatures (rules) to detect such traffic. The challenges that are faced by these studies can be summarized as the following:

- The employed datasets mostly include two-way traffic with payload.
- Commonly transport, network and application layer protocol usages were studied to discover the attack patterns.
- For the researches related to building machine learning classifiers to detect malicious traffic, the robustness and generalization of such classifiers were not evaluated by employing such data collected in different time periods.

This thesis is complementary to these researches in the aspects of analyzing darknet traffic using well-known open source and commercial tools and studying machine learning classifiers to understand if they can be used to overcome the aforementioned

challenges (if at all). The main differences between the previous researches and this thesis are:

- This research focuses on analyzing publicly available backscatter data sets including no payload information. In doing so, it is aimed to explore common attack patterns seen in such traffic. Thus, the attack signatures revealed by previous researches focusing on two-way malicious traffic are extended by those patterns belonging to one-way malicious traffic.

- Besides studying the usage of the transport, network and application protocols, packet-based features including packet length, TTL, source location etc. (explained in detail in the Chapter 3.4) are also employed to find out general attack patterns. In doing so, it is aimed to observe if using packet-based features in addition to the features related to the transport, network and application layers, commonly employed by the previous researches, are sufficient to detect malicious activity.

- The robustness and generalization abilities of a C4.5 classifier trained by old darknet datasets are analyzed by running this classifier on recent datasets, which potentially include different attack patterns. The main purpose by doing so is to observe how much the auto-generated rules by the C4.5 classifier successful to detect such attack packets having different trends.

Chapter 3

Methodology

This chapter presents the main characteristics of the employed traffic traces, the description of the network traffic monitoring systems and the focused supervised learning systems used as well as the feature selection methods and performance metrics employed in this thesis.

3.1 The Characteristics of Employed Datasets

Six publicly available darknet traffic traces (datasets) from CAIDA's archives are employed in this thesis. These traces are captured by UCSD Network Telescope [25], which is located in San Diego, consisting of a globally routed /8 network that rarely carries legitimate traffic. Since this network telescope is a type of a passive darknet, the traffic captured by this telescope include only one-way traffic, i.e. the response traffic from the attack victim to other IP addresses. According to CAIDA's explanations [6][5][4][1], all the employed darknet datasets except "April-2012" include mostly backscatter attacks, whereas "April-2012" [7] may involve scanning data in addition to backscatter data. Note that the destination IP addresses and payload information are hidden in these datasets for the sake of privacy.

Table 3.1: Employed Datasets

	May-28 2004	**Feb-23 2006**	**Feb-22 2008**	**Nov 2008**	**Apr 2012**	**DDoS 2007**	**Norm 2008**	**Norm 2014**
Size (GB)	4.4	7.5	6.9	102.7	11.2	0.1	0.15	5
# of packets	57,641,141	85,547,065	81,606,489	1,317,888,867	15,817,479	1,000,000	1,000,000	50,000,000
DDoS Pattern Analysis	✓	✓	✓	×	×	×	×	×
Snort	✓	✓	✓	×	×	×	×	×
Bro IDS	✓	✓	✓	✓	×	×	×	×
Iatmon	×	×	×	✓	×	×	×	×
Corsaro	×	×	×	✓	✓	×	×	×
Cisco ASA	×	×	×	✓	✓	×	×	×
Machine Learning Classifiers	×	×	×	✓	✓	✓	✓	✓

Table 3.1 shows all the employed datasets in this thesis. These datasets contain

a large number of instances. It should also be noted here that TCP and Port 80 are the major protocol and the major port used in these datasets, respectively.

The earliest days from the captured 2004, 2006 and 2008 backscatter datasets provided by CAIDA were selected as one-day datasets to employ to study both the different backscatter behaviours over these years (2004 - 2008) and the performance of the tools Snort and Bro IDS. Also, November-2008 (the latest available backscatter dataset from year 2008) and April-2012 (the latest available darknet dataset) datasets were employed to evaluate the performances of Bro, Corsaro and Cisco ASA 5515-X systems as well as the machine learning classifiers. Note that two additional datasets including normal network traffic from CAIDA's archives collected in 2008 [2] and 2014 [3] via CAIDA's "equinix-chicago" and "equinix-sanjose" monitors on OC-192 Internet backbone links and one more darknet dataset including only DDoS traffic captured in 2007 via UCSD Network Telescope were also employed in order to study the best practices to form training datasets for the machine learning classifiers. In doing so, it is aimed to have more consistent training sets containing both labelled attack and normal traffic with different patterns to build a discriminative classifier. Table 3.2 summarizes my efforts to form different training data sets using the aforementioned normal and attack datasets, where:

- D1 includes 1,000,000 records from the dataset November-2008.
- D2 is a combination of the November-2008 and DDoS-2007 datasets including 1,000,000 records from each of them.
- D3 is a combination of the November-2008 and Normal-2008 datasets including 1,000,000 records from each of them.
- D4 is a combination of the November-2008, DDoS-2007 and Normal-2008 data sets including 1,000,000 records from each of them.
- D5 is a combination of the April-2012 and Normal-2014 datasets including 50,000,000 records from each of them.

Note that while the datasets D1, D2, D3 and D4 are used in the training phases, D5 is used in only testing phases. The reason behind this is to explore if it is possible to detect recent attack traffic via a machine learning based classifier trained in the

past. To the best of my knowledge, this is the first research which focuses on building a robust classifier by employing such combinations of backscatter and normal traffic from the past to analyze today's attack traffic.

Table 3.2: Formation of Training Datasets

Dataset	Datasets for Training (1,000,000 records from each "1")				Dataset for Testing (50,000,000 records from each "1")
	D1	D2	D3	D4	D5
November-2008	✓	✓	✓	✓	×
DDoS-2007	×	✓	×	✓	×
Normal-2008	×	×	✓	✓	×
April-2012	×	×	×	×	✓
Normal-2014	×	×	×	×	✓

3.2 Systems Employed

This section describes the network traffic monitoring systems employed in this thesis to explore their performances against backscatter traffic.

3.2.1 Wireshark and Tshark

Wireshark [16] and Tshark [14] are publicly available network traffic analyzers capable of capturing network packets and displaying them as detailed as possible. These tools mostly help one to troubleshoot network problems and handle security problems. While Wireshark has a graphical front-end, Tshark can only be used in Terminal since it does not have a user interface. Figure 3.1 and 3.2 represent an example display from November-2008 dataset in Wireshark and Tshark, respectively. Note that while Wireshark v1.10.6 was used for visualizing and examining the packets, Tshark v1.10.6 was used to extract the information under specific features (explained in the section 3.4) of the packets in the datasets employed. Since both of the tools do not have any system for intrusion detection, these tools only monitor the traffic over a network and enable one to analyze the traffic.

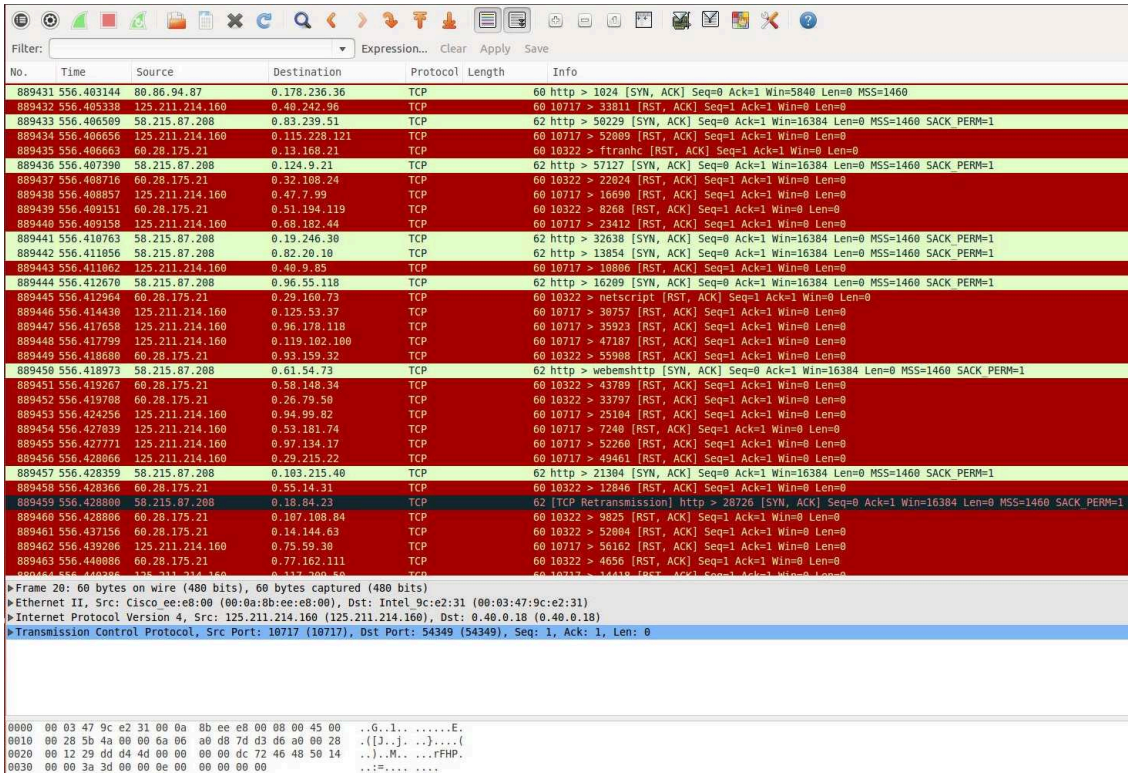

Figure 3.1: Wireshark presentation

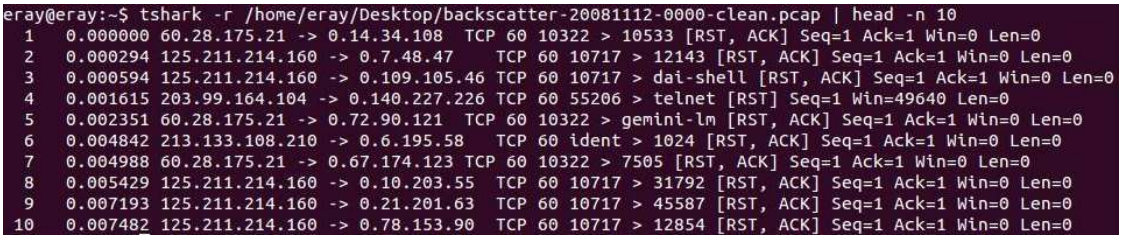

Figure 3.2: Tshark presentation

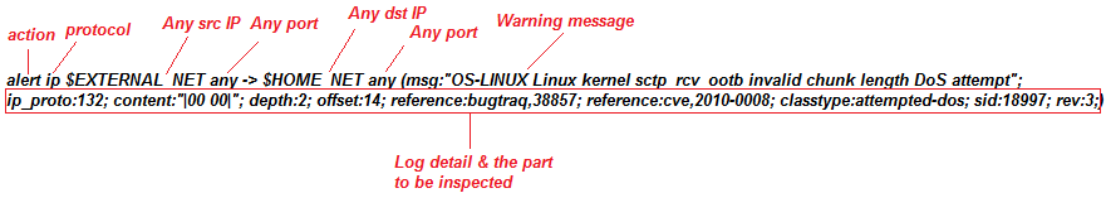

Figure 3.3: Snort Rule sample

3.2.2 Snort

Snort is a well-known rule-based open-source network IDS which is capable of monitoring real time network traffic and logging the transmitting packets on IP networks. This system is developed to perform protocol analysis, content searching, and to prevent a network against many type of attacks and probes such as buffer overflows, port scans and Server Message Block (SMB) probes. Each rule in Snort consists of a header and a tail part. The header part includes the information about the rule's action, protocol, source and destination IP addresses, netmasks and port numbers, whereas the tail part contains the content of the packets supposed to be inspected by the rules and the warning message to be logged. Figure 3.3 represents a sample rule of Snort and shows its partitions.

The versions 2.9.1 and 2.9.6.0, released in 2011 and 2014 respectively, were employed to perform the experiments on Snort in this thesis. The version 2.9.1 has 3,111 rules in total and they are classified into the following 8 main categories to detect malicious traffic activity:

- Indicator - includes the rule sets used for detecting compromised systems, encrypted contents, shell-script based attacks.
- Malware - includes the rule sets used for detecting botnets, harmful softwares and tools, and backdoors.
- Policy - includes the rule sets used for detecting spams and potential violations of social media and multimedia.
- Protocol - includes the rule sets used for detecting threats and vulnerabilities over network layer protocols.
- PuA - includes the rule sets used for detecting "Potentially Unwanted Applications" such as adwares and spywares.
- Server - includes the rule sets used for detecting vulnerabilities or attacks on a server, i.e. mail server.
- Web - includes the rule sets used for detecting web-based attacks.

- Other - includes the rule sets used for detecting attacks for other situations.

However, the newer version, v2.9.6.0, includes 21,838 rules under 11 main categories, the following three categories in addition to aforementioned ones:

- Browser - includes the rule sets used for detecting the vulnerabilities present in web browsers such as Chrome, Firefox etc.
- File - includes the rule sets used for detecting the vulnerabilities in executable, image, flash, multimedia and text files.
- Operating System - includes the rule sets used for detecting operating system based malicious activities.

The increase in the rules is expected to increase the performance of Snort since there is more chance for an attack packet to trigger a rule and to be detected. To find out which of those rules are effective in detecting backscatter packets, each of the rules and the categories were analyzed in this research and explained in detail in the Chapter 4.2.1.

3.2.3 Bro

Bro is an open-source network traffic analyzer that can be used to detect malicious activity, anomaly detection and semantic misuse on a network system passively based on pre-defined rules and events, or to increase its performance [52]. This tool records all the activities on a network in detail by using well-structured logging system to reveal the vulnerabilities of the network clearly. Figure 3.4 presents an example of a pre-defined event in Bro. This event is used for detecting client-side softwares using File Transfer Protocol (FTP) server.

```
event ftp_request(c: connection, command: string, arg: string) &priority=4
{
   if ( command == "CLNT" )
   {
      Software::found(c$id, [$unparsed_version=arg, $host=c$id$orig_h, $software_type=CLIENT]);
   }
}
```

Figure 3.4: Example of an event in Bro v2.2.

Bro has a layered structure mainly consisting of two different parts: event engine and interpreter. Event engine is used to reform the IP fragments of the packets

whose IP headers could not be verified by this layer, to run over the complete IP datagrams and to decrease the number of filtered events by eliminating lower-level network events. On the other hand, interpreter layer is used to provide real-time notifications and logging. Figure 3.5 reflects the structure of Bro IDS with its main components. The main purpose of this structure is to enable Bro to handle large volumes of traffic without dropping packets. Note that using libpcap formatted files to perform an offline analysis aids the tool to decrease its processing time in analyzing the traffic since such files do not contain any details regarding the Data Link Layer technology used. Moreover, if the operating system of the device that the libpcap files have been captured has a powerful kernel packet filter, these files provide special filters to decrease the traffic on the kernel, which allows Bro to deal with less amount of network traffic.

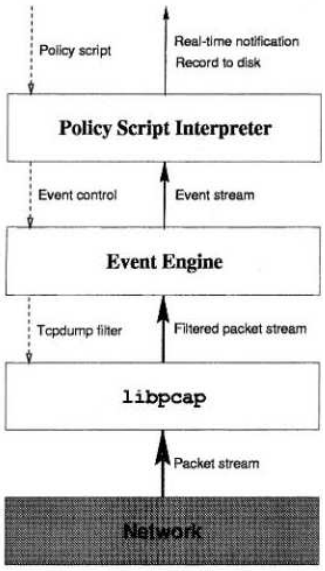

Figure 3.5: Structure of Bro IDS from [52]

3.2.4 Iatmon

Inter-Arrival Time Monitor (Iatmon) is an open source network traffic analyzer designed to monitor only one-way traffic by categorizing the traffic based on the source types, characteristics and inter-arrival time [24]. The main purpose of Iatmon to help one to explore unsolicited traffic on a network, such as the traffic destined to any unused address.

-----	Stealth & 3S mode	Stealth & spikes	Stealth Other	Left Skew	Even	Right Skew	Short-lived	High-rate	Dos	Ungrouped
Probe										
Vertical Scan										
Horizontal Scan										
Other										
Probe										
Vertical Scan										
Horizontal Scan										
Other										
ICMP only										
Backscatter										
TCP and UDP										
uTorrent										
Conflicker C										
Unkvped										

Figure 3.6: Defined subsets in Iatmon

Iatmon generates log files based on 14x10 matrices consisting of 14 types and 10 groups which indicates 140 subsets to categorize the network traffic and source hosts as shown in Figure 3.6. In this figure, the rows represent the types and the column represent the groups. To decide on type of packet, the information extracted from the packet, such as the protocol and the port number, is used. In this case, the extracted information includes: the packet size, the lifetime, the rate and the time interval between the consecutive packets. In the output matrix, the parts highlighted in yellow, Figure 3.6, represent the TCP packets, whereas the parts highlighted in green represent the UDP packets and the parts that are highlighted in red represent the other packets. On the other hand, the columns highlighted in purple represent the long-lived packets, the blue part represents 3-second packets (these are generated by such hosts that procures a new packet in every 3 seconds) and the red part represents the other packets for the columns. Note that while deciding the types and groups for source addresses, the packets generated by those addresses are considered as the same way explained above. It is also important to underline that the packets coming from such sources that have been idle for at least 120 seconds or have not sent more that two packets are not considered as suspicious by Iatmon.

Figure 3.7: Example of 8-tuple structure in Corsaro

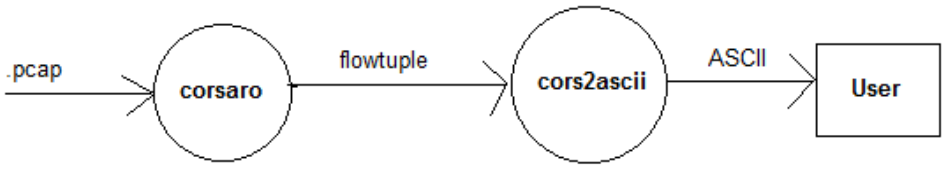

Figure 3.8: Data processing in Corsaro

3.2.5 Corsaro

Corsaro is a network monitoring tool specifically designed for I/O threads to provide significant functions such as high compression rate and faster packet inspection for offline analysis of large network traffic to increase the processing speed and efficiency [41]. The major advantage of Corsaro is that it enables network administrators to define time intervals between the inspected packets. In this way, one can gain information about the packets transmitting in a specific period of daytime.

Corsaro supports FlowTuple representation for logging the network traffic. These include the following 8 features for each packet: source IP, destination IP, source port, destination port, protocol, TTL, TCP flags and IP length. Figure 3.7 shows an example of a packet stored in 8-tuple format by Corsaro. Note that converting this format into human-readable ASCII format is necessary to display the log data to users. Figure 3.8 presents the data processing steps performed by Corsaro.

3.2.6 Cisco ASA 5515-X

It is required to ensure the security of a network system by inspecting all the incoming packets to the system and deciding the suspicious ones. A firewall is such a system used to decide whether to accept or drop packets based on the organization's security policy. The devices using Cisco's Adaptive Security Appliance (ASA) as an

operating system includes additional firewall capabilities such as integrated antivirus applications and intrusion detection systems to increase the security and performance of a network system [8].

These additional capabilities mentioned above change based on the version of the used ASA model. In this thesis, the model 5515-X [9] has been employed, which has the following additional capabilities:

- Integrated Intrusion Prevention System (I-IPS) is used to prevent network attacks to infect a network system. There are two options to configure this module. Inline mode is one of these options. This mode enables the ASA to send the original packets to the I-IPS module for inspection, whereas the promiscuous mode allows to copy the traffic for the I-IPS module. Therefore, while all the suspicious traffic is prevented to reach its destination in inline mode, the ASA keeps handling their routes until it gets a warning from the I-IPS module in the promiscious mode.

- Web Security Essentials (WSE) Service is used to ensure the reliability of specified web-based applications.

- Cisco Cloud Web Security (CCWS) is used to increase web security for cloud-based organizations.

- Application Visibility and Control (AVC) Service is used to monitor the traffic on specified applications.

Note that the I-IPS function is used in inline mode for this thesis since stream normalization techniques are only applied in inline mode by eliminating or reducing many of the network evasion capabilities to increase the speed of the I-IPS [27]. This module can drop a network packet when:

- The packet has a bad format,

- Suspicious Internet Control Message Protocol (ICMP) or scanning packet is detected,

- Connection limit is exceeded,

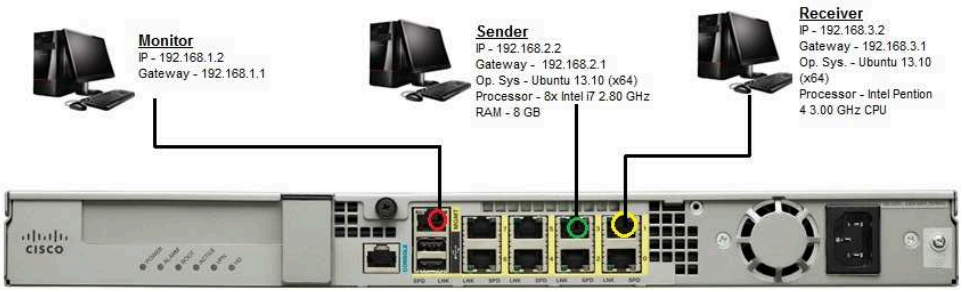

Figure 3.9: Configuration of the Cisco ASA 5515-X Router.

- Interface is overloaded,
- Generated by such hosts listed in user-specified black list,
- Packet is failed from firewall check or I-IPS inspection,
- Session is incompleted,
- Packets failed application inspection,

Three different computers are used to configure the Cisco ASA device, a sender, a receiver and a monitor are used as shown in Figure 3.9. A Unix-based tool, Tcpreplay v3.4.4, that enables to replay libpcap formatted network traffic files has been used to replay the employed traffic from the sender computer to the receiver one. The interface "Cisco Adaptive Security Device Manager (ASDM) v6.6", released in 2012, is used on the Cisco ASA 5515-X device.

3.3 Machine Learning Approaches Employed

Supervised learning is a popular machine learning technique used to create a model by analyzing training data which is labelled based on its ground-truth to classify any new instances. Decision Tree, Naive Bayes and AdaBoost are three well-known supervised machine learning classifiers employed for the machine learning based experiments. Note that Weka v3.6.10 [39] is used to perform these experiments using the default parameters.

3.3.1 C4.5 Decision Tree

Decision Tree classifier generates a tree model to show the relation between the features and to sort them from the most informative one to the least. To this end, it starts with using the entire training set as a root node. Then, the dataset is divided into child nodes after measuring all the possible split alternatives and their performances based on information gain ratio of each alternative. Thus, the tree representation helps to reveal the most informative features via displaying the parent-child relations between them based on its algorithm. Therefore, this approach is beneficial to explore a set of rules from a large amount of training data [36].

J48 algorithm has been employed to design C4.5 Decision Tree model to perform the related measurements in this thesis. This approach marks the features based on information gain ratio to choose the parent nodes which represent the most informative ones. To this end, it calculates *Entropy* value by employing the entire training set to find the root (top informative) node as shown in Eq. 3.1.

$$Entropy(S) = -\sum_{i=1}^{u} p_i log_2(p_i) \tag{3.1}$$

where u refers to the total number of different labels and p_i refers to the rate of the occurrence for the current label over the entire data set. After that, the algorithm measures the entropy for the each split of S as described in Eq. 3.2:

$$Entropy_F(S) = -\sum_{i=1}^{m} p(D_i) Entropy(D_i) \tag{3.2}$$

where F represents a feature and m refers to the total number of the different values F has. Then, the information gain value for the feature F is calculated by using the formula defined in Eq. 3.3:

$$GainRatio(F) = \frac{Entropy(S) - Entropy_F(S)}{SplitInfo_F(D)} \tag{3.3}$$

where $SplitInfo_F(D)$ is the value due to the split of D depending on the value of the feature F (Eq. 3.4).

$$SplitInfo_F(D) = -\sum_{i=1}^{n} p(D_i) log_2(p(D_i)) \tag{3.4}$$

After the root node is specified, all the different values for the root node are shown as branches. This algoritm keeps generating sub-nodes and branches by following the formulas above until it manages to complete a tree representation that includes all the necessary rules to categorize new instances based on the training set.

3.3.2 Naive Bayes

Naive Bayes is one of the simplest linear classifiers in the literature. In this case, each feature in the training set affects the final decision equally and independently since it does not include an integrated feature selection algorithm, unlike C4.5 Decision Tree. Although this presents an increased computational efficiency since it provides fast data analysis and decision making, it makes a Naive Bayes model incapable of using multiple features simultaneously while building the model, which may cause missing some important information for the classification process [32].

"NaiveBayes" algorithm, which calculates posterior probabilities of a class value based on the Bayesian Theorem shown in Eq. 3.5 has been employed to build Naive Bayes model to perform the related measurements in this thesis.

$$P(f|c) = \frac{P(c|f)P(f)}{P(c)} \tag{3.5}$$

where $P(f|c)$ refers to the posterior probability of a feature value f for the given c, $P(c|f)$ shows the likelihood of the feature value f, $P(f)$ represents the prior probability of the f and $P(c)$ presents the probability of c in the training set. Also, for such features that include continuous information, Gaussian distribution is used to calculate the probabilities as shown in Eq. 3.6.

$$F(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3.6}$$

where σ represents standard deviation and μ refers to the arithmetic mean of the values.

3.3.3 AdaBoost

AdaBoost is a machine learning technique that recursively calculates the weights of the classifiers by using a weak learning algorithm and iteratively combines the weak

learners to generate a strong learner (classifier). On each cycle, the algorithm employs a different sample from the training data. In the end, it presents a strong classifier by adding the weak classifiers that have performed higher accuracy in each cycle. Note that this approach is sensitive to outliers and noisy data [61].

AdaBoost.M1 algorithm has been employed to perform the related experiments in this thesis. This approach starts with initializing the weights of training samples as shown in Eq. 3.7.

$$w(c) = \frac{c}{m} \tag{3.7}$$

where $c = 1, 2, \ldots, m$ and m refers to the number of cycles. Then, a weak classifier, $F_c(s)$ is computed for each cycle as described in Eq. 3.8.

$$F_c(s) = \sum_{c=1}^{n} f_c(s) \tag{3.8}$$

where f_c denotes the current weak learner for the cycle c and s refers to the current cycle and input sample, respectively. After that, the error rate of the weak classifiers is measured as defined in Eq. 3.9.

$$E_c = \sum_{i}^{n} E[F_c(s_i) + f_c(s_i)] \tag{3.9}$$

where $E(F_c(s_i))$ shows the current error of the weak classifier F_c generated via current weak learner f_c for the current sample input s_i. It should be marked here that even if there is a single situation where E_c, the total error rates, is more than 50% for a weak classifier, AdaBoost.M1 fails and aborts the recursive loop. Then, β_c values are calculated for each cycle and the weights of the training samples are updated as defined in Eq. 3.10 and Eq. 3.11, respectively.

$$\beta_c = \frac{E_c}{1 - E_c} \tag{3.10}$$

$$W_{c+1}(s_i) = \frac{w_c(s_i)}{C_c} \begin{cases} \beta_c & \mathrm{F}_c(s_i) = y_i \in \{1, k\} \\ 1 & \text{Otherwise} \end{cases} \tag{3.11}$$

where C_c is a constant to normalize the weight of the training sample s_i, y_i represents the value of the weak classifier for the current training sample, $F_c(s_i)$, and k

is the finite cardinality. Finally, the final classifier is generated by the combination of linear weak classifiers as described in Eq. 3.12.

$$F_{fin}(X) = \arg \max_{y \in \{1,k\}} \sum_{F_c(x)=y} log \frac{1}{\beta_c} \tag{3.12}$$

In summary, Table 3.3 summarizes the employed machine learning techniques. As to be seen, each of them has its own advantages and disadvantages:

- C4.5 has the advantage of using information gain ratio as an integrated feature selection technique, which enables it to select the most informative features without user specification, and the disadvantage of learning slowly from the training data.

- Naive Bayes has the advantage of being fast while learning from the training data, which makes it easier to adapt new records and updating the model, and the disadvantage of considering all the features independently.

- AdaBoost.M1 has the advantage of allowing to select or modify the weak learner algorithm it uses, which makes it flexible to be employed for different areas, and has the disadvantage of being tough while modification is required since it learns from the training data even slower than C4.5 model. Moreover, if the error rate of a weak learner for even one iteration is more than 50%, the algorithm stops running. It should be also noted here that AdaBoost is the most sensitive one to noisy data and outliers.

- All the employed models have the advantage of being used for multi-class classification on such data that include continuous or discrete features, or both.

- All the employed models may have "overfitting" problems occurring in case of biased (in some way) training datasets.

3.4 Feature Selection

Feature selection is a data preprocessing technique aiming to employ the most informative features to construct a machine learning classifier. It helps increasing the

Table 3.3: Summary of Employed Machine Learning Approaches

C4.5 Decision Tree	**Naive Bayes**	**AdaBoost**
Handling both continuous and discrete attributes	Handling both continuous and discrete attributes	Handling both continuous and discrete attributes
Non-linear	Linear	Non-linear
Hard to update a model (Slow learning)	Easy to update a model (Fast learning)	Hard to update a model (Slow learning)
Integrated feature selection to reduce dimentionality	All features are considered independently	Fails if there is a weak learner with an error rate greater than 50%
Can overfit	Can overfit	Can overfit

prediction accuracy of a machine learning classifier as well as decreasing its computational cost and memory usage by eliminating irrelevant and redundant features used in a training set. Selecting the most informative features is one of the main challenges since it is hard to obtain them. Note that the selected features as well as the training instances are significant factors that can affect the performance of the classifiers. To this end, common attack patterns of malicious network traffic are explored in Chapter 4.1 to understand the informative features. Also, two well-known statistical measurement methods, namely chi-square and symmetrical uncertainty, have been used as evaluation criteria for Ranker Search, which returns a sorted list of features based on the defined evaluation criteria, to specify the top informative features. Table 3.4 shows all the informative features available in the employed datasets.

3.4.1 Chi-square Measurement

Chi-squre (X^2) is a statistical measurement technique used to individually evaluate features by computing the independence level between the co-occurrence of two different values, v and c, with respect to their classes via Eq. 3.13 [60].

$$X^2(v,c) = \frac{N((vc)(v'c') - (vc')(v'c))^2}{((vc)+(v'c))((vc')+(v'c'))((vc)+(vc'))((v'c)+(v'c'))} \tag{3.13}$$

Table 3.4: All available features in the Employed Datasets

Feature	Description
ip.src	IP address where a packet is sent from
ip.srccountry	Defines the country where a packet is sent from
port.src	Port number where a packet is sent from
port.dst	Port number where a packet is sent to
deltatime	Time interval between two consecutive packets
frame.len	Length
frame.caplen	Length stored into the capture file
offset	Fragment offset
ip.ttl	Time-to-live value of a packet
ip.proto	Protocol information of a packet
ip.checksum_good	Good checksum
ip.checksum_bad	Bad checksum
tcp.stream	Stream index
tcp.seq	Sequence number
ns flag	Nonce-sum flag
ecn flag	Explicit congestion notification flag
ack flag	Acknowledgment flag
psh flag	Push flag
res flag	Reset flag
syn flag	Synchronization flag
fin flag	Finish flag
icmp.type	Icmp type of a packet
icmp.code	Icmp code of a packet
icmp.checksum_bad	Bad checksum for an ICMP packet
alert	"yes" for backscatter packets, "no" for scanning and misconfiguration packets

where (vc) refers to the number of the times v and c co-occurred, $(v'c)$ represents the number of the times c occurred without v, (vc') shows the number of the times v occurred without c and $(v'c')$ denotes the number of the times that neither v nor c occurred at the same time. Note that the value of X^2 decreases when the independence level between the values increases.

3.4.2 Symmetrical Uncertainty Ranked Method

Symmetry is an important and desired factor while measuring the correlations between the features. Information Gain, which is also employed while building C4.5 Decision Tree models, is a symmetrical approach which gains the same amount of

information about feature F_1 after analyzing feature F_2 and vice versa. Symmetrical Uncertainty (SU) is a information gain based correlation measurement technique which calculates the mean of two uncertainty coefficients by normalizing information gain results to between 0 and 1 as shown in Eq. 3.14; where 0 shows the features are independent and 1 indicates that the features are dependent [17].

$$SU(F_1, F_2) = 2 * \frac{IG(F_1|F_2)}{Entropy(F_1)Entropy(F_2)} \tag{3.14}$$

Algorithm 1 presents how ranker search approach works with SU. For a given training data T, it begins with measuring the correlation value between the feature f and the class c for each feature $\{f_1, f_2, ..., f_t\}$ and places the features in an order based on their correlation values, starting from the highest to the lowest. Then, it compares the features from the first element to the last based on their correlations to select the predominant ones and eliminate the redundant ones. This process is kept until there is no redundant feature to be removed.

Algorithm 1 How Symmetrical Uncertainty Works

Data: Training data T with t features

Result: Optimal feature set

foreach *f in the training set* **do**
  Calculate $SU_{c,f}$ for the class c and feature f.
end

Place the features on a list in descending order based on the SU values.

Assign the first feature as $f_{current}$.

while *available* f_{next} **do**
  if $SU_{f_{current},f_{next}} >= SU_{f_{next},class}$ **then**
    remove f_{next}
  else
    $f_{current} \leftarrow f_{next}$
    continue;
  end
end

return *ordered feature list*

3.4.3 Features Based on the Experiments

The following 12 features (from Table 3.4) have been selected as the most informative ones based on the literature and the observations from data classification experiments explained in the Chapter 4.1: *ip.src, ip.srcountry, port.src, port.dst, ip.proto, syn flag, res flag, ack flag, ip.ttl, frame.len, deltatime and alert.* Note that while obtaining the geo-locations of the attack sources, GeoLite_Country [10] database has been employed. It should be also noted here that "deltatime" has been selected since the researchers manage to reach promising results by using packet rate to detect DDoS attacks in [48][30][45]. Finally, Jin et al. emphasized that checking TTL values of the network packets is important to detect DDoS attacks in [28], so this included in this research, too.

After extracting those features, two different feature sets were specified. While *Feature set-1* includes all the features above, *Feature set-2* includes all the features except the IP addresses and port numbers. The reason behind creating these feature sets is to understand if it is possible to detect backscatter traffic without checking the IP addresses and port numbers, which are the most common features used to detect malicious traffic in the literature since they can be easily spoofed. Note that since destination IP addresses are hidden by CAIDA, this feature could not be selected at all.

Table 3.5: Experiment-based Selected Features

Feature set-1	**Feature set-2**
ip.src	
ip.srccountry	
port.src	ip.srccountry
port.dst	ip.proto
ip.ttl	res flag
frame.len	ack flag
deltatime	syn flag
ip.proto	frame.len
res flag	deltatime
syn flag	ip.ttl
ack flag	
alert	

3.5 Performance Metrics

The values procured by contingency table (or confusion matrix), helps analyzers to measure the performance of a machine learning based classifier by providing the actual and predicted classes via true positive (TP), true negative (TN), false positive (FP) and false positive (FN) parameters. In this thesis, TP refers to the correctly categorized attack instances, TN indicates the correctly categorized non-attack instances, FN represents the incorrectly categorized actual attack instances and FP shows the incorrectly categorized actual non-attack instances. Assume that X is an attack packet and Y is a non-attack packet (i.e. misconfiguration). Then, the following examples present what the aforementioned parameters represent:

- TP - X is classified as an attack packet.
- TN - Y is classified as an non-attack packet.
- FP - Y is classified as an attack packet.
- FN - X is classified as an non-attack packet.

Accuracy, precision, recall (sensitivity), specificity and F-score values are measured via the aforementioned parameters from the contingency tables provided by the trained classifiers by using the Eq. 3.15, 3.16, 3.17, 3.18 and 3.19, respectively:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.15}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.16}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.17}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.18}$$

$$F - measure = 2\left(\frac{(Precision)(Recall)}{Precision + Recall}\right) \tag{3.19}$$

Accuracy refers to the fraction of the correct predictions, including both correctly classified attack and non-attack packets, over all the predictions. Precision represents the fraction of correctly predicted attack packets over all the predictions. Recall denotes to the fraction of the correctly predicted attack packets over the actual attack packets. Specificity shows the incorrectly predicted non-attack packets over the actual attack packets. Finally, F-measure is the harmonic mean of precision and recall values. Note that Precision and Recall are commonly used in information retrieval and data mining areas since they help to decrease FP and FN values. Therefore, it is aimed to maximize the precision and recall values to construct a discriminative classifier [55]. For all the parameters above, 1 (100%) is always the most expected result where 0 (0%) is the result for the worst cases.

One of the main goals of this thesis is to build an accurate machine learning based classifier with high precision, recall and specificity and low false negative rate (FNR) and false positive rate (FPR) values. FNR and FPR scores are calculated as shown in Eq. 3.20 and 3.21, respectively.

$$FNR = 1 - recall \tag{3.20}$$

$$FPR = 1 - specificity \tag{3.21}$$

To the best of my knowledge, this is the first reseach aiming to analyze the performance of the aforementioned classifiers with different sizes of training sets and different feature sets and compare them with the network traffic monitoring tools on backscatter (darknet) datasets. This research follows the performance metrics given above to evaluate the performances of different systems to compare their capabilities and complexities. Different sizes of training sets and different feature sets are employed to emphasize how data pre-processing step is crucial to build a machine learning classifier.

Chapter 4

Experiments and Results

This chapter focuses on the analysis, experiments and evaluations that were carried out during the research. At a glance, multiple discussions are brought up about the nature of the backscatter traffic and how successful the employed network traffic monitoring tools are in detecting such traffic. Additionally, measurements on the employed supervised machine learning techniques are presented.

4.1 Revealing the Characteristics of Backscatter Traffic and Changing Trends in Time

My aim in this section is to find appropriate answers for the following questions:

1. What is the nature of backscatter attacks?
2. What are the roles of well known secure ports or P2P (Peer-to-Peer) applications in this traffic (if any)?
3. What are the major countries producing backscatter traffic?

To this end, I focus on three different backscatter datasets over a 4-year period to discover the nature of backscatter traffic and changed patterns in that time. The earliest days, May 28 from 2004, Feb 23 from 2006 and Feb 22 from 2008, from the entire datasets are selected for this experiment to find out how the behaviour of backscatter attacks differ (if at all) over the two-year intervals.This analysis also gives an insight into the features that are more informative to use for building a machine learning based classification approach.

4.1.1 Data Classification

In general, there are three main categories used to classify darknet datasets: Scanning, backscatter and misconfiguration. Scanning data is generated by attackers to find out

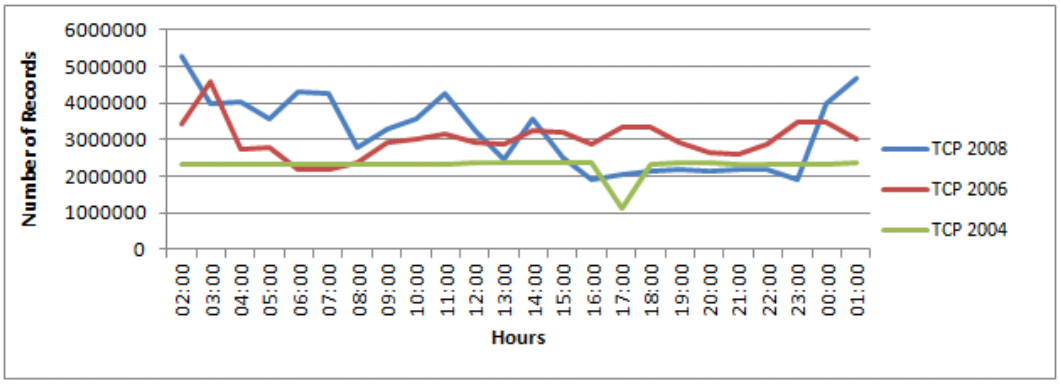

Figure 4.1: Hour-based distribution of the backscatter traffic

the potential vulnerable targets. Backscatter data represents the traffic generated by DDoS victims to many spoofed IP addresses. Finally, misconfiguration data refers to such traffic generated by any software, hardware and user based faults [19].

The packets in the employed traffic are classified into the aforementioned categories by using their TCP flags based on the technique explained in [48] and [58]. According to these researches, the SYN packets represent the scanning traffic, where RST, ACK, RST|ACK and SYN|ACK packets refer to the backscatter traffic and the remaining packets belong to misconfiguration traffic. Table 4.1 presents the distribution of the traffic under these categories for each selected dataset. This table demonstrates that the datasets collected in 2008 and before mostly include backscatter traffic where the darknet dataset collected in 2012 has a more balanced distribution.

Since the older datasets mostly include backscatter traffic according to Table 4.1, the daily hour-based distribution of such traffic is analyzed and represented in Figure 4.1 to explore at what times backscatter attacks are more likely to be generated. Interestingly, the increase of the observed backscatter traffic between 9 a.m to 12 p.m indicates that the attack packets is likely to reach an high intense in working hours since employees may cause backscatter traffic unintentionally via some business tools.

Only the backscatter packets in the traffic are also categorized based on their TCP flags to observe the overall measurement of the TCP flags used and to study the nature of the backscatter traffic in depth, Table 4.2. Clearly, although the number of RST and RST|ACK packets are more in 2004, it decreases for the years 2006 and 2008, in opposed to the increase of SYN|ACK packets. Note that the SYN,

Table 4.1: Types of Traffic Observed in Each Dataset

Category	Dataset				
	May 28, 2004	**Feb 23, 2006**	**Feb 22, 2008**	**Nov 2008**	**April 2012**
Scanning	0.01%	0.2%	0.1%	0.5%	41.6%
Backscatter	90.54%	78.1%	88%	96.2%	32.8%
Misconfiguration	9.45%	21.7%	11.9%	3.3%	25.6%

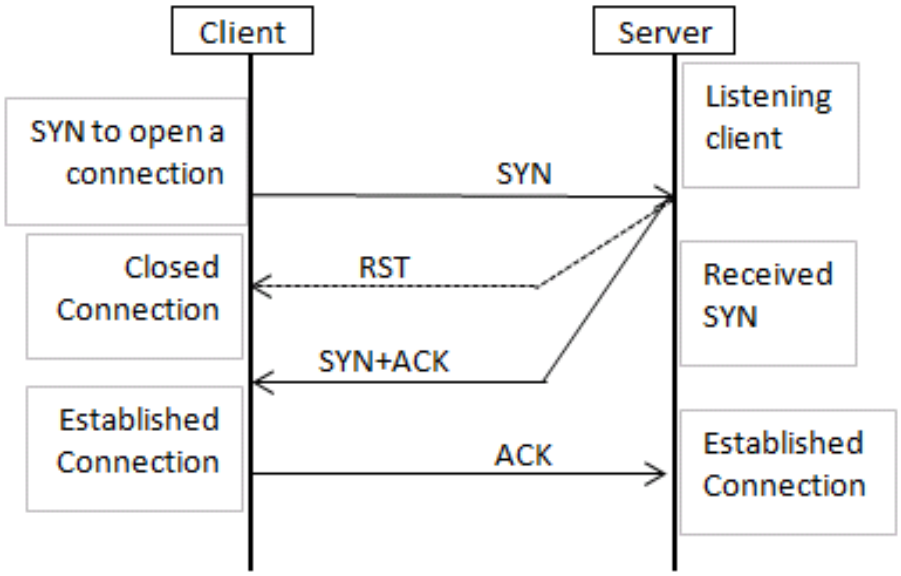

Figure 4.2: Three-way handshaking for TCP connections

SYN|ACK and ACK packets are used for providing three-way handshaking mechanism in TCP connections where the RST packets are potentially used for making a reconnection between the sender and receiver hosts when the sender one does not receive any SYN|ACK packets over a long period of time for any SYN packet it has sent, Figure 4.2. Needless to say, the high percentage in the number of SYN|ACK packets and the low percentage in the number of ACK packets also indicate the total number of incomplete three-way handshaking attempts in TCP connections, which causes an increase in the occurrence of backscatter attacks. As mentioned before in Chapter 3.1, many online gaming web sites [49] were exposed to DDoS attacks in February, 2008. Also, many outstanding blog sites [46] and payment gateways [47] were targeted by DDoS attacks in February 2006. These examples demonstrate that the selected darknet datasets include real DDoS attack traffic. On the other hand, there are less DDoS attacks observed in May 2004 compared to the other years in

this experiment.

Table 4.2: TCP Traffic Distributions of Backscatter Datasets Employed

TCP Flag Type	**Dataset**		
	May 28, 2004	**Feb 23, 2006**	**Feb 22, 2008**
ACK	0.02%	0%	0%
SYN|ACK	21.48%	82.5%	77.1%
RST	35.8%	2.4%	6.1%
RST|ACK	42.7%	15.1%	16.8%

4.1.2 Measurements on Transport and Network Layer Protocols

Transport Layer is a layer that provides reliability and quality in managing end-to-end communications by supporting flow control and error checking mechanisms, whereas network layer provides data routing. The most well-known transport layer protocols are TCP and UDP, which are respectively used for connection-oriented and connectionless transmissions. For network layer, IP is the most popular protocol, and ICMP is an extension of IP that is mostly used for sending error and control messages.

The distribution of TCP, UDP and ICMP traffic are measured and presented in Table 4.3 to clarify the importance of the transport and network layer protocols commonly used in the datasets employed. As to be seen, TCP is the major used protocol in these backscatter datasets since almost 98.5% of the traffic from 2004 is classified as TCP traffic where the other datasets also include almost 90%. The main reason behind that is the use of three-way handshaking mechanism TCP provides. In a way, this provides a useful infrastructure for the well-known techniques to generate backscatter attacks. Note that the researches in [51] reported many network attacks using or targeting TCP ports as well. Also, there is some usage of ICMP but no usage of UDP packets in the datasets employed. It is also seen that when TCP traffic decreases, backscatter traffic also decreases as opposed to the misconfiguration traffic, Table 4.1.

It is important to note here that even though there is no UDP packets seen in these datasets, type-11 ICMP packets, which refer to those packets whose TTL value

is equal to 0, are observed. In this case, the first 64 bits of the packets are kept, which enables one to have the transport layer information of the previous packet. Therefore, through type-11 ICMP packets, it becomes possible to observe the UDP packets in these datasets. Table 4.3 also presents the intensity of these type-11 ICMP packets. It is observed that the dataset from 2006 has the highest ratio of type-11 ICMP packets, 17.2%, whereas the dataset from 2004 has the lowest, 1.4%. The dataset from 2008 has 12% of such packets, half of them has UDP as the transport layer protocol and the remaining has TCP, and comes right after the one from 2006; however, the type-11 ICMP packets this dataset includes have different characteristics than the ones in 2006. Note that the transport layer protocol of the 88% of the type-11 ICMP packets in 2006 is UDP. It is also important to emphasize that according to Table 4.3, the number of such UDP packet packets is maximum in 2006 followed by 2008 and 2004, whereas 2006 dataset includes the least number of backscatter packets following by 2008 and 2004 according to Table 4.1. Therefore, it is seen that the intensity of backscatter traffic decreases when the ratio of these UDP packets increases. Finally, the total ratio of the packets under "other" category is less than 0.1% for each dataset analyzed.

Table 4.3: Protocol Measurements

Protocol			**Dataset**		
			May 28, 2004	**Feb 23, 2006**	**Feb 22, 2008**
		TCP	98.4%	88.22%	87.2%
ICMP	***Type-11***	***TCP***	1.1%	2.2%	6%
		UDP	0.3%	15%	6%
	Other	***Only ICMP***	0.12%	0.63%	0%
		Other	0.08%	0.05%	0.01%

4.1.3 Measurements on Application Layer Protocols

Application layer is the layer that manages the communication between the applications by providing interfaces. There are many different application layer protocols used between the source and the destination hosts.

The port numbers of the packets are analyzed to explore the top used application layer protocols in these one-day datasets since there is no payload information available in the traffic. Figure 4.3 shows the distribution of the application ports. It seems like most of the packets are sent over the HyperText Transfer Protocol (HTTP) protocol since more than 50% of the packets are routed over port 80. Also, although the application layer protocols used in the datasets from 2006 and 2008 show different trends, there is no other major application layer protocol used in the dataset from 2004. Note that besides the HTTP protocol, the following application layer protocols are seen in the datasets 2006 and 2008:

- Nterm: It is a terminal-based application running on port 1026 used for making the use of different applications, directories, documents and URL easier [11].
- Csd-monitor: It is a specific channel running on port 3072 and used by DDoS attackers [42]. Therefore, this port is generally blocked to avoid from being a vulnerable target.
- Dcerpc: This protocol enables one to run a software application on a remote server over the port 135. It should be noted here that W32.Blaster worm, which was used for blocking the Remote Procedure Call (RPC) service in August 2003, uses this protocol, which explains why we see this port a lot in the datasets employed [23].
- SMB: This protocol which uses port 445 to run on Microsoft Active Directory is used to manage shared accesses and file transmission on multiple threads over a network.
- NetBios: This protocol running on the ports 137, 138 and 139 enables to execute applications on different hosts over a LAN [23].
- Telnet: It is a well-known command-based protocol to handle remote connections over port 23.
- IRC: This protocol is used for chatting over the ports 194 or 667.
- FTP: This protocol runs on the ports 20 and 21 and enables file transmission between computers over a network.

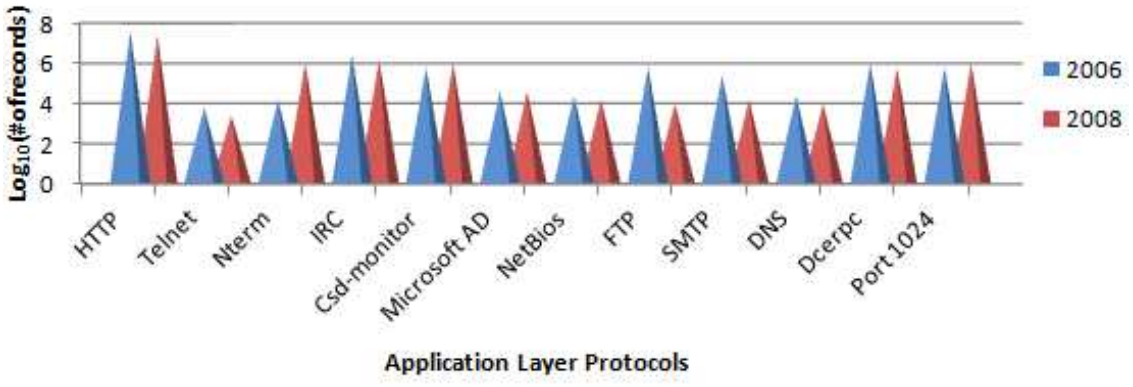

Figure 4.3: Application Layer Protocol Usage for the datasets 2006 and 2008.

- Simple Mail Transfer Protocol (SMTP): This is a very well-known protocol used for e-mail transmission between IP networks on port 25.
- DNS: This protocol runs on port 53 and is used to convert IP addresses into domain names (and vice versa).

Note that all the port numbers above are the default port numbers that these applications use.

4.1.4 Measurements on Secure Traffic

It is not desired for a person listening to a data transmission illegally to understand the meaning of transferred data. Secure protocols are used to provide that reliability during data transmission by encrypting the payload information.

Although it is not possible to be absolutely sure how much of the traffic is encrypted in these datasets since they do not contain any payload information, the usage of the following 38 ports from [13] are measured for in these datasets:

- 22 - Secure Shell (SSH) [12]
- 443 - Secure Sockets Layer (SSL)
- 465 - SMTP over SSL
- 563 - Network News Transfer Protocol (NNTP) over SSL
- 585 - Internet Mail Access Protocol v4 (IMAP4) over SSL

- 614 - SSLshell
- 636 - Lightweight Directory Access Protocol (LDAP) over SSL
- 695 - Multimedia Messaging Service (MMS) over SSL
- 989/990 - FTP over SSL
- 992 - Telnet over SSL
- 993 - IMAPS over SSL
- 994 - IRC over SSL
- 995 - Post Office Protocol (POP) over SSL
- 1701 - Layer Two Tunneling Protocol (L2TP) over Internet Protocol security (IPsec) [22]
- 1723 - Point-to-Point Tunneling Protoco (PPTP)
- 7800 - Apple Software Restore
- 7801 - Secure Server Protocol (SSP) Client
- 7900 to 7913: These are rarely used secure ports.
- 7914 to 7920 - Even though these ports are unassigned by Internet Assigned Numbers Authority (IANA), Snort v2.9.6.0 accepts them as secure ports.

It is challenging to identify all the secure traffic in such datasets since encrypted traffic data can be transferred by using other ports like port 80 (e.g. Skype), too [18].

According to the measurement about the overall usage of these secure ports shown by Figure 4.4, the backscatter dataset from 2008 has the most amount of secure traffic, although the total percentage of the potential secure traffic is less than half a percent for each dataset. Within this small portion, it is seen that SSH and SSL are the top secure ports used in these datasets. SSH is mostly used in 2006 and SSL in 2004. These results seem to indicate that secure ports are rarely used in generating backscatter attacks.

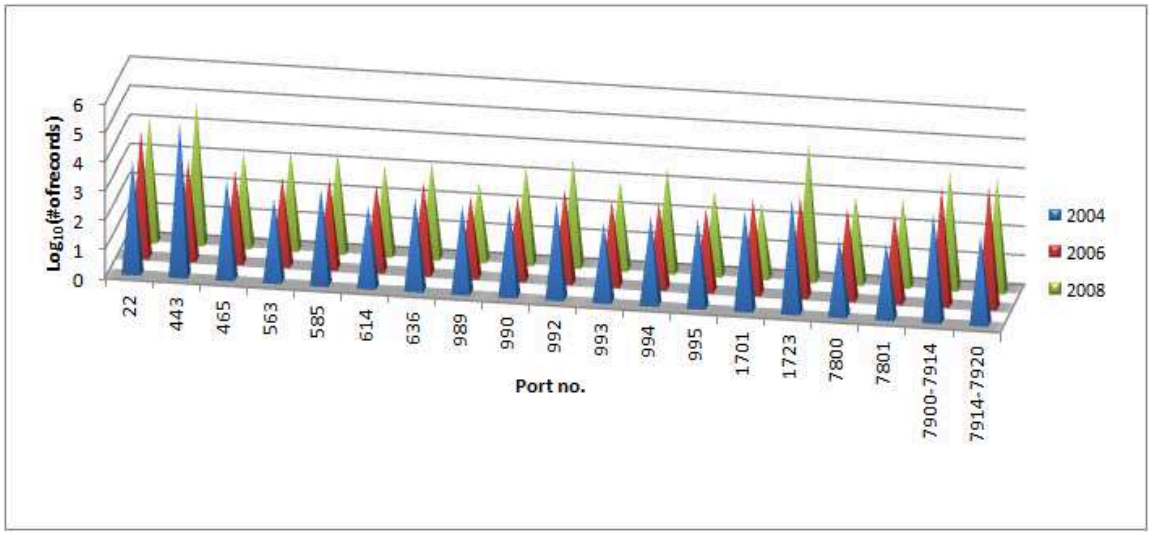

Figure 4.4: Backscatter traffic on secure ports

4.1.5 Measurements on Peer-to-peer (P2P) Traffic

Peer-to-peer (P2P) network is a communication model in which none of the hosts is controlled by a centralized server and each host has the same capabilities. This model allows each host to behave like both a client and a server, which means that any node can start a communication session.

The usage of P2P applications, especially the ones providing file sharing, have considerably grown for the last several years. Even though the port numbers such applications use are modifiable, the usage of the default ports for the 10 following well-known P2P applications is investigated to reveal the P2P activities in backscatter attack traffic: Edonkey[53], Gnutella (bearshare and limewire)[53], KaZaA[53], DirectConnect[53], BitTorrent[53], WinMx[40], Ares[40], Soulseek[40] and Waste[15].

According to the measurements, the total usage of the P2P traffic is less than 0.05% for each dataset. However, SoulSeek, BitTorrent and Edonkey are the top used P2P applications by DDoS attackers in 2004, 2006 and 2008 within this traffic, respectively, shown in Figure 4.5. On the other hand, the usage of KaZaA, Waste and Soulseek drastically decreases in years. It is interesting to note that the researchers in [54] stated that there were 40 different viruses observed between February and May 2006 in more than 15% of tha data KaZaA includes. So, even though KaZaA was a popular file sharing application until 2006, the popularity of KaZaA dramatically decreased since many of its clients were infected because of the vary of viruses KaZaA

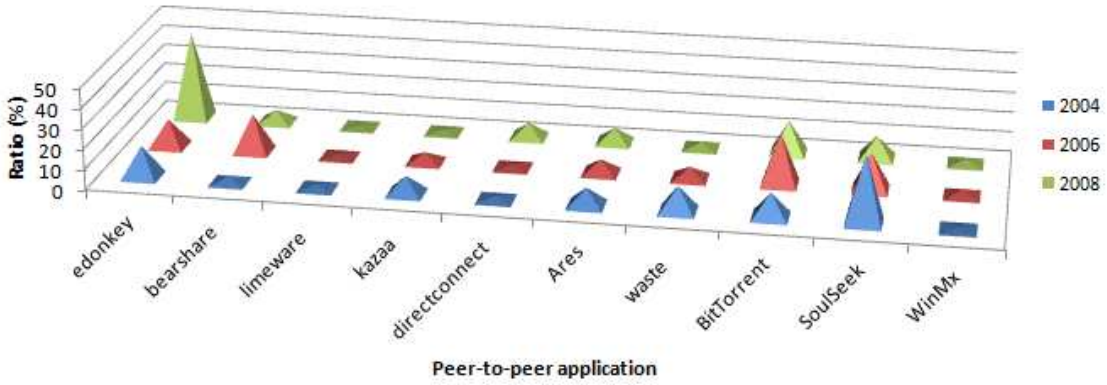

Figure 4.5: Peer-to-peer Traffic Measurements

includes. This explains why the highest usage of KaZaA is observed in the dataset from 2004. and why the usage decreases in the following years. Moreover, Bearshare was also very popular when it was first released in 2006; however, it lost its popularity quickly. This is the reason why the usage of Bearshare is more in the dataset from 2006 than the other ones.

4.1.6 Geolocation Analysis

Geolocation information which can be found by checking the IP addresses has become an important part of monitoring the movement of network data. Analyzing the movements enables one to determine if the incoming traffic is suspicious or not.

The geologic distribution of the backscatter traffic in the datasets employed is measured based on the source IP addresses of the packets generated by backscatter victims to the spoofed IP addresses by using publicly available GeoLite database [10]. Figure 4.6 presents the major source countries seen in these datasets with ratio of how much attack traffic they produce. As observed, China plays the major role for each dataset. Moreover, even though more than 10% of such traffic was generated from Pakistan in 2004, the role of Pakistan has dramatically decreased in the other years. It seems like USA and Taiwan are the only countries where there is an increase in the ratio of the backscatter traffic they generated from 2004 to 2008. However, such attacks were likely to be more distributed since the "other" countries contribute to almost 13% and 14% of the datasets from 2004 and 2006, respectively whereas this number drops to 2% in 2008.

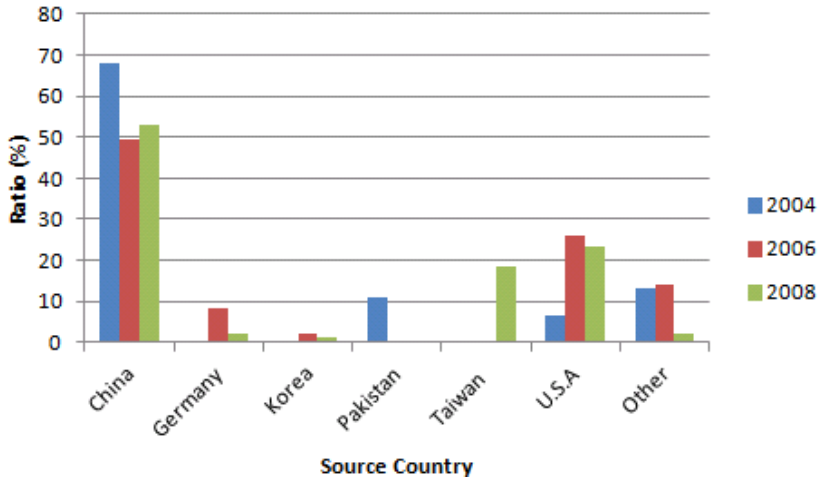

Figure 4.6: Geolocation Measurements

In summary, three different backscatter datasets profiling a general overview of the backscatter patterns from 2004, 2006 and 2008 are analyzed to reveal those patterns and how they change (if at all) over time. According to our results, the patterns of backscatter traffic show changes over that time. However, it seems that TCP is the major transport layer protocol and HTTP (port 80) is the top used application layer protocol used by such traffic for each year. Moreover, while port 22 (SSH) is the most used secure port in 2006, port 443 (SSL) is the most popular secure port for the years 2004 and 2008. Also, Soulseek, BitTorrent and eDonkey are the top used peer-to-peer applications in 2004, 2006 and 2008, respectively. It should be noted here that both secure traffic and peer-to-peer traffic have a very small portion of role in such traffic. Finally, China and USA are the major countries generating backscatter attacks in these datasets.

4.2 Measurements on the Network Traffic Monitoring Tools

In this section, the performances of 5 well-known network traffic monitoring tools are evaluated in terms of their success on detecting one-way attack traffic, and computational time on specific darknet data. Moreover, the signatures, events and features used by these tools are also analyzed. Thus, I aim to explore the main challenges of these tools affecting their performances in terms of both detection rate (false alarm rate) and processing time.

I begin with comparing the performances of Snort and Bro on the one-day darknet datasets from May 28, 2004, Feb 23, 2006 and Feb 22, 2008. Then, I measure the performances of the tools Bro, Iatmon, Corsaro and Cisco ASA 5515-X on the largest backscatter dataset, November-2008. Finally, I analyze the behaviours of Corsaro and Cisco ASA 5515-X on the one-way traffic dataset that includes different types of network attacks instead of only backscatter attacks, April-2012. Note that the tool that has the highest performance has been selected after each step. The reason behind following such a "compare and select the best" strategy is to find out the network analyzer that gives the best results on the one-way darknet datasets. Table 4.4 explains which of these tools are run over which darknet datasets. This experiments would shed light into understanding how much of the attack behaviours seen in real-life datasets can be detected via such tools. All the experiments related to network traffic monitoring tools were performed by using a computer with 32 GB RAM, Intel i5 3.10 GHz CPU and Ubuntu 13.10 operating system.

Table 4.4: Overview of the Employed Tools and Darknet Datasets

Dataset	**Tool**				
	Snort	***Bro***	***Iatmon***	***Corsaro***	***Cisco ASA***
May 28, 2004	Yes	Yes	No	No	No
Feb 23, 2006	Yes	Yes	No	No	No
Feb 22, 2008	Yes	Yes	No	No	No
November, 2008	No	Yes	Yes	Yes	Yes
April, 2012	No	No	No	Yes	Yes

4.2.1 Snort v2.9.1 and Snort v2.9.6.0

As explained before in Chapter 3.2.2, Snort is a rule-based network intrusion detection system that uses special filters on each incoming network packet to decide if the packet is suspicious or not. Table 4.5 presents the number of categories, rule sets for each category and the triggered rules by the employed network packets. The column "# of sub categories" represents the number of the different rule sets under each category. For instance, "dns.rules" and "telnet.rules" are such rule sets under the protocol category. The column "# of rules" shows the total number of the rules for each category. For example, the category "Malware" has 82 different rules defined in

Snort 2.9.1. Finally, the column "# of triggered rules" refers to the number of the rules match with the attack pattern of at least ten packets in the datasets employed. In these experiments, there has to be at least one triggered rule if a packet is defined as suspicious. Therefore, each rule that has caused a packet to be assigned as suspicious for at least once is counted as a triggered rule.

Table 4.5: Analysis on Snort rules

	v2.9.1			**v2.9.6.0**		
	#of sub categories	**#of rules**	**#of triggered rules**	**#of sub categories**	**#of rules**	**#of triggered rules**
Server	3	582	0	10	3779	0
Malware	1	82	0	4	3835	0
Indicator	8	294	4	6	2814	0
Protocol	13	529	20	15	1136	13
PuA	1	18	0	4	876	0
Policy	2	30	0	4	404	0
Web	7	1070	0	0	0	0
Op.Sys.	0	0	0	5	722	0
File	0	0	0	9	4624	0
Browser	0	0	0	6	2875	0
Other	9	506	7	6	1173	7
Total	44	3111	31	69	21238	20

As observed, Snort v2.9.6.0 includes more than 20.000 rules, which is approximately seven times more than the rules in v2.9.1. Although the browser, file and operating system categories are not included by Snort v2.9.1, there are 8221 rules generated for those categories in Snort v2.9.6.0. In fact, the "file" category becomes the largest category with 4624 different rules for v2.9.6.0. Also, the "web" category from v2.9.1 has been removed with all its rules in v2.9.6.0. However, even though the number of total rules increases in the newer version, v2.9.6.0, the number of triggered rules decreases by 11 for the employed darknet data, which drops its detection rate. I speculate that the reason behind the removal of those rules (by the Snort organization) is that those might cause false alarms since the trends of current network attacks are much different than the time Snort v2.9.1 was released.

Here, I also focus on analyzing every single rule included in the both versions to reveal the triggered ones on the employed one-way darknet datasets (see Table 4.4)

Table 4.6: Snort Performance Analysis

Snort Version	May 28, 2004		Feb 23, 2006		Feb 22, 2008	
	All Traffic	O.B.T.	All Traffic	O.B.T.	All Traffic	O.B.T.
Snort v2.9.1	2%	0.01%	15.8%	0.01%	10.6%	0.05%
Snort v2.9.6.0	1.2%	0.04%	17.2%	0.01%	11.6%	0.01%

[1]O.B.T. = Only Backscatter Traffic (based on TCP flags explained in [48] and [58].)

since the number of triggered rules and their content give important indications regarding the type of attack patterns Snort is sensitive to. Table 4.7 presents all the triggered rules by Snort v2.9.6.0 including both the deleted and the newly added ones. It is clear that the rules under the protocol-icmp rule set, especially the ones "Destination & Port Unreachable" and "TTL Exceeded in Transit", have reached the highest detection rates while monitoring a network against backscatter traffic. Also, the rule "Hi Client Unknown Method" seems to be the only added new rule in the version 2.9.6.0 that seems to be useful for identifying these attack behaviours. Finally, one of the triggered rules, namely "Destination Unreachable Communication With Destination Host is Administratively Prohibited", which has managed to identify some malicious traffic in my experiments has been removed in Snort v2.9.6.0.

After completing the analysis on Snort rules, I run those rules on just the backscatter traffic of my datasets (classified by the rules in [48] and [58]) to evaluate the performance of Snort and its two different versions, Table 4.6. It should be noted here that Snort does not inspect the packets being trasmitted over a secure port, which causes it to miss such attack packets. It is interesting to observe from Table 4.6, Snort v2.9.1 gives a better performance in detecting backscatter traffic although the number of rules in the version 2.9.6.0 has drastically increased. However, when the rule set "deleted.rules", which includes all the removed rules of Snort until the release of the version 2.9.6.0, is used, the detection rate becomes closer to the detection rate of version 2.9.1. This indicates that even though the detection rate in Snort v2.9.1 is higher than the version 2.9.6.0, the older version causes false alarms by assigning misconfiguration packets as backscatter since most of the deleted rules are not actually attacks symptoms. Moreover, all the experiments on both Snort versions have taken 30 minutes in total. Finally, Snort does not achieve any high detection rates

(17.2% at most) on any of these datasets.

4.2.2 Bro v2.2

The performance of Bro has been measured by employing one-day datasets as well as the November-2008 dataset in terms of processing time and detecting malicious traffic on various backscatter attack patterns included by those traffic data. It is important to clarify that since there is not a default script Bro v2.2 includes to detect backscatter traffic, the script called "scan.bro", which warns the network administrators in case of the events shown in algorithm 2, has been employed to perform the related experiments. According to that algorithm, Bro basically counts each failed connection attempt and gives alerts when the total number of them reaches a specific threshold. The reason behind those events is that the scanning activity causes a high number of unsuccessful connections since it uses incomplete three-way handshaking attempts [21].

Algorithm 2 How scan.bro Detects Backscatter Traffic

Data: one-way darknet dataset

foreach *packet p in the given dataset* **do**
    read *p*;
    if *(failed connection attempt* || *aborted TCP connection* || *rejected TCP connection* || *interrupted connection)* **then**
        classify *p* under "backscatter" category
    else
        classify *p* under "other" category
    end
end

Table 4.8 presents the detection rates Bro has managed to reach on such backscatter datasets. As to be seen, 15% of the 2006 dataset has matched with the rules defined in "scan.bro" as maximum. Non-existence of any default script for detecting backscatter traffic is the main reason why the detection rates are low. Also, it is seen that misconfiguration traffic has a low influence on the performance of Bro IDS, 3.4% as maximum. Since it is demonstrated on one-day datasets that Bro IDS generates less false positives, it is also run over all the traffic for the November-2008 dataset

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h
id.resp_p       proto   service         duration        orig_bytes
resp_bytes      conn_state      local_orig      missed_bytes    history orig_pkts
orig_ip_bytes   resp_pkts       resp_ip_bytes   tunnel_parents
#types  time    string  addr    port    addr
port    enum    string  interval     count
count   string  bool    count   string
count   count   count   count   set[string]
#data
1226480400.014003       Cv3u6p2eLqZOu52XQi      0.29.7.1        32086   205.177.208.45
80      tcp     -       -       -
-       OTH     -       0       h
0       0       1       40      (empty)
1226480400.010033       CKZViI3tfmm0Hvrzu8      0.43.168.17     1522    205.177.208.45
80      tcp     -       -       -
-       OTH     -       0       h
0       0       1       40      (empty)
```

Figure 4.7: Example of the log representation in Bro

and observed that it classified 13.5% of the traffic as suspicious. Moreover, completing all the experiments on Bro has taken approximately 3 days. The major reason behind the high processing time of Bro is that Bro generates very detailed log files. For example, even though the November-2008 dataset includes 102.7 GB of data, the size of the log files generated by Bro after running on this dataset is 145 GB. Note that the log file generated by Bro, namely "conn.log", includes all the packets in the traffic. Figure 4.7 shows a sample screenshot belonging to the conn.log file generated after running Bro on the November-2008 dataset.

According to the results from the experiments on Snort and Bro, it seems like Bro has a higher performance than Snort in detecting backscatter traffic. It has reached up to 11.2% true positive ratio as maximum whereas Snort has reached only to 0.05%. Also, the ratio of false positives is 22.6% in Bro, while more than 99% of the alerts generated by Snort are false alarms. Therefore, Bro v2.2 has been selected over Snort to run on the November-2008 dataset, Table 4.8. These results also indicate that even though Snort and Bro are effective and widely-used tools to perform network traffic monitoring in general, they are not sufficient to use in detecting backscatter traffic that does not include payload information.

Table 4.8: Bro v2.2 Performance Analysis

Dataset		**Bro v2.2 Detection Rate**
May 28, 2004	***All Traffic***	2.1%
	O.B.T	0.23%
Feb 23, 2006	***All Traffic***	15%
	O.B.T	11.6%
Feb 22, 2008	***All Traffic***	5.5%
	O.B.T	5.2%
November-2008	***All Traffic***	13.5%

[2]O.B.T. = Only Backscatter Traffic (based on TCP flags explained in [48] and [58].)

4.2.3 Iatmon v2.1.2

Iatmon has only been applied to the traffic captured in November-2008. Table 4.9 presents the measurements on the packets in that traffic while Table 4.10 shows the measurements on the source addresses generate those packets. According to the results obtained from these tables, Iatmon detects 3.3% of the packets as backscatter traffic as well as 4.1% of all the source IP addresses as suspicious. The main reason behind these low results is that Iatmon ignores all the packets transmitting over a source that has been idle for at least 120 seconds or has not sent more than two packets at all. Also, Iatmon uses only a few features to detect backscatter traffic: only the ACK and RST flags, and also the TTL value of that packet. Using this system, it is observed that 83% of the employed traffic includes stealth packets generated by 41,318 different hosts that has stayed active for at least 30 minutes. On the other hand, it seems like 14% of the packets, generated by 6,811 different hosts, are categorized as 3-second traffic, which indicates that those packets are generated by hosts that continuously produce packets in 3 seconds intervals. According to those results, although Iatmon does not reach high rates in detecting backscatter traffic and suspicious source addresses, it provides a new perspective on the network traffic to understand the different behaviours as the ones discussed above.

Table 4.7: Snort - Triggered Rules/# of Triggered Times

Rule	Dataset		
	May 28, 2004	**Feb 23, 2006**	**Feb 22, 2008**
protocol-icmp.rules			
TTL exceed in Transit	257,892	2,719,675	4,968,767
Destination Unreachable Host Unreachable	54,269	1,336,068	420,929
Echo Reply	28,649	29,386	44,677
DestinationUnreachable Port Unreachable	5,141	9,343,279	3,146,756
Destination Unreachable Network Unreachable	2,684	39,800	13,835
Destination Unreachable Protocol Unreachable	793	1,225	2,987
Fragment Reassembly Time Exceeded	217	2,681	867
Destination Unreachable Fragmentation Needed and DF bit was set	82	6,794	818
protocol-snmp.rules			
Request TCP	473	692	3,334
AgentX/TCP request	26	476	425
Trap TCP	86	364	1,275
preprocessor.rules			
Hi Client Unknown Method (Added by v2.9.6.0)	804,825	1,306	11,406
Short Fragmentation	576	6,763	0
Anomaly Overlap	452	956	144
Bad Reset	254	88	21
Teardrop	232	71	361
Excessive Overlap	137	698	139
Tiny Fragment	42	288	0
Anomaly Oversize	12	11	0
Hi Server No Contlen	4	76	1
deleted.rules			
Destination Unreachable Communication Administratively Prohibited	217,046	1,027,602	703,227
Destination Unreachable Communication with Destination Network is Administratively Prohibited	51,186	8,682	938
Redirect Host	23,375	169,817	102,122
Source Quench	8,319	1,940	1,660
TCP Port 0 Traffic	6,572	2,874	3,188
Same SRC/DST	2,734	1,321	631
IP Reserved Bit Set	1,200	337	23
Redirect Net	426	10,135	2,344
Bad Frag Bits	207	81	0
Large ICMP Packet	0	3,663	0

Table 4.9: Iatmon packet categorization

	Stealth & 3S	Stealth & Spikes	Stealth Other	Left Skew	Even	Right Skew	Short -lived	High -rate	Dos	Ungrouped
Probe	908	81	163920	86058	0	0	0	27	544	293
Vert. Scan	176679	37144	2298	17728	0	0	0	815	133991	94699
Horz. Scan	25491	11314	2044	34445	0	0	953	2047	11174	23764
Other	8337401	13983	565515	5370542	1082	0	6508	203	15448	18835
Probe	1272652	56934	42928639	10476885	0	10653	80860	4433	347228	590323
Vert. Scan	0	0	0	0	0	0	0	0	0	0
Horz. Scan	0	0	0	0	0	0	0	0	0	0
Other	0	0	0	0	0	0	0	0	0	0
ICMP only	0	0	0	0	0	0	0	0	0	0
Back-scatter	4167493	42659	6820839	31942257	535978	788808	79769	2243	79396	36767
TCP &UDP	775684	424910	1065866544	147252880	5072	938	48618	7316	293025	215190
uTorrent	0	0	0	0	0	0	0	0	0	0
Con-flicker	0	0	0	0	0	0	0	0	0	0
Untyped	0	6	4290981	0	0	0	0	0	11	113

Table 4.10: Iatmon source address categorization

	Stealth & 3S	Stealth & Spikes	Stealth Other	Left Skew	Even	Right Skew	Short -lived	High -rate	Dos	Ungrouped
Probe	5	2	1	3	0	0	0	2	76	5
Vert. Scan	423	7471	2	33	0	0	0	55	12389	3751
Horz. Scan	78	713	1	35	0	0	4	55	727	692
Other	2051	637	7	733	1	0	10	11	1286	535
Probe	3002	6541	184	943	0	3	14	180	32342	15000
Vert. Scan	0	0	0	0	0	0	0	0	0	0
Horz. Scan	0	0	0	0	0	0	0	0	0	0
Other	0	0	0	0	0	0	0	0	0	0
ICMP only	0	0	0	0	0	0	0	0	0	0
Back-scatter	634	1704	176	958	27	26	24	53	10905	827
TCP &UDP	1321	11465	4900	4037	10	2	41	253	26179	5905
uTorrent	0	0	0	0	0	0	0	0	0	0
Con-flicker	0	0	0	0	0	0	0	0	0	0
Untyped	0	2	1	0	0	0	0	0	2	2

4.2.4 Corsaro v2.0.0

The November-2008 and April-2012 darknet datasets have been employed to measure the performance of Corsaro in detecting one-way malicious network traffic. This tool uses the TCP flags as well as the ICMP types to identify the attack packets, Algorithm 3.

Algorithm 3 How Corsaro Detects Backscatter Traffic

Data: Darknet dataset

```
foreach packet p in the given dataset do
    read p;
    if p.protocol = TCP then
        if ((p.SYN = 1 & & p.ACK = 1) || p.RST = 1) then
            categorize p under backscatter class
        else
            categorize p under other class
        end
    else if p.protocol = ICMP then
        if (p.IcmpType ϵ {0, 3, 4, 5, 11, 12, 14, 16, 18}) then
            categorize p under backscatter class
        else
            categorize p under icmpReq class
        end
    else
        categorize p under other class
    end
end
```

Corsaro has managed to detect almost 99% of the backscatter traffic in the November-2008 dataset and 23% of the backscatter traffic in the April-2012 dataset in approximately 2 hours. Those results demonstrate that Corsaro reaches high detection rates against backscatter traffic. Note that Corsaro has provided the best performance in detecting such traffic as an open-source tool for the experiments completed until this point.

4.2.5 Cisco ASA 5515-X

The performance of the Cisco ASA 5515-X device has been evaluated on the November 2008 and April 2012 darknet datasets. According to the measurements, 55% of the traffic from the November 2008 and 22% of the traffic from the April-2012 have been classified as suspicious data and the related packets have been immediately dropped. These results are lower than the Corsaro's performance on the same data sets. This was surprising for me. However, this low performance of such a well known commercial tool might be because of the fact that this is a one-way traffic with no payload.

In summary, it is interesting to observe that the employed signature/event based network traffic monitoring systems seem to be insufficient to detect one-way DDoS-based activity on a network. Note that identifying DDoS-based activity is also very challenging by such systems since the IP addresses and port numbers are generally spoofed and the attack patterns seem to change over time. Table 4.11 presents the summary of the performance results of the tools employed in this thesis. According to these results, Corsaro v2.0.0 reaches the highest detection rate, 99%, when it is run over the November-2008 dataset.

4.3 Building Machine Learning Classifiers

In this section, my main objective is to build supervised learning based models by using C4.5 Decision Tree, Naive Bayes and AdaBoost.M1 techniques to analyze one-way network traffic traces. Additionally, I intend to study whether different feature sets affect the performance of such models as well as to study how small a training dataset could achieve high performances in detecting malicious behaviours in such datasets. I think this is an important question since obtaining a training dataset comes with the cost (challenge) of "labelling" (groundtruth) the dataset. The effect of training set size and its features are analyzed in section 4.3.1, the performances of the employed machine learning classifiers built by small training sets are discussed in section 4.3.2, and the robustness of the rules generated by C4.5 classifier trained by old data against new attack trends is presented in section 4.3.3. It should be noted here that all the experiments related to the machine learning classifiers were performed by using a computer with 32 GB RAM, Intel i5 3.10 GHz CPU and Ubuntu

Table 4.11: Overview of Employed Tools

	Snort v2.9.6.0	**Bro IDS v2.2**	**Iatmon v2.1.2**	**Corsaro v2.0.0**	**Cisco ASA 5515-X**
Released year	2014	2013	2012	2013	2012
Highest detection rate reached	17.2%	13.5%	3.3%	99%	55%
Major advantage	Easy to add, update or remove a rule	Supports script based detection	Various categories for both packets and source IP addresses	Reaches high detection rates in a reasonable time	Supports visual represen-tations
Major dis-advantage	False alarms may occur	Slow because of storing large log files	Ignores packets coming from 2-minutes idle hosts	High performance against only DDoS activity	Low perfor-mance against one-way traffic

14.04 operating system.

4.3.1 Experimens on Training Sets via C4.5 and Naive Bayes

At this point, two machine learning approaches, namely C4.5 and Naive Bayes, are employed with my proposed set of features, shown in Table 3.5, to observe the overall performance of the approaches on the November-200 backscatter dataset as well as to find out the importance of training set size and using IP addresses and port numbers of the attack packets in the training set. Table 4.12 shows the evaluation of these machine learning models built by different feature sets and different sizes of training sets. As observed, C4.5 reaches higher accuracy, precision, recall, specificity and F-score values than Naive Bayes in each case. The changes on the values of those parameters depending on the used features, as to be seen in Table 4.12, indicates that the performance of a machine learning classifier is based on the selected features. Furthermore, employing a small portion of data in the training phase and keeping the remaining for the testing phase provides consistent predictions with reasonable

ratios, which seems to indicate that having a large training set is not required to build a strong classifier. This is an important indication for human experts since it may help them to overcome the labelling challenge with a lower cost.

Table 4.12 also presents the approximate processing time of the employed models. It is clearly seen that even though it is faster to build a Naive Bayes classifier than a C4.5 classifier, the run time of C4.5 is approximately a hundred times less than Naive Bayes's. Also, it seems that IP addresses and port numbers are not required to be used as features to reach high accuracies in detecting malicious network traffic. It should be noted here that "ip.proto", "ack flag", "res flag", "frame.len" and "ip.ttl" are the major features employed by C4.5 classifier in generalizing the tree models.

Table 4.12: Machine Learning with Different Features and Training Sets

		80% train 20% test		**20% train 80% test**	
		C4.5	**Naive Bayes**	**C4.5**	**Naive Bayes**
Feature set-1	*Accuracy*	0.99	0.97	0.99	0.98
	Precision	0.99	0.74	0.96	0.74
	Recall	0.99	0.85	1	0.93
	F-measure	0.99	0.79	0.98	0.82
	Specificity	0.99	0.98	0.99	0.98
	FNR	0.01	0.15	0	0.07
	FPR	0.01	0.02	0.01	0.02
	Building time (mins)	360	150	150	90
	Running time (mins)	20	900	40	5400
Feature set-2	*Accuracy*	0.98	0.97	0.99	0.98
	Precision	0.77	0.79	0.96	0.82
	Recall	0.98	0.83	1	0.88
	F-measure	0.87	0.81	0.98	0.84
	Specificity	0.99	0.99	0.99	0.99
	FNR	0.02	0.17	0	0.12
	FPR	0.01	0.01	0.01	0.01
	Building time (mins)	330	90	90	55
	Running time (mins)	10	1500	30	6300

The experiments in this section demonstrate that the size of a training set and

the selected features affect the performance of a machine learning based classifier. Moreover, the higher performance achieved when Feature set-2 was used indicate that it is not required to use source and destination IP addresses as well as port numbers to detect one-way attack traffic. Since training such a classifier by using 20% of the records with Feature set-2 resulted in higher performance, smaller amount of instances and Feature set-2 (over Feature set-1) are used for the next experiments in the training phase.

4.3.2 Comparing the Employed Machine Learning Techniques

Here, six different machine learning models are constructed by employing 10% of the November-2008 and April-2012 darknet datasets with Feature set-2, since Feature set-2 has showed the most optimal performance in the previous experiments, as on the below:

- C4.5 Decision Tree with 10% of November-2008 dataset
- C4.5 Decision Tree with 10% of April-2012 dataset
- Naive Bayes with 10% of November-2008 dataset
- Naive Bayes with 10% of April-2012 dataset
- AdaBoost with 10% of November-2008 dataset
- AdaBoost with 10% of April-2012 dataset

Once each trained model is generated, the remaining data (90% unseen) of the same dataset that has been used to train the model is used to evaluate the performance of the models in terms of detection rate and processing time. The main purpose of this experiment is to select the most successful supervised learning approach to build a comprehensive model with those datasets collected in 2008 or before to run on the ones collected in 2012 or after to observe how much they are successful in classifying such data that has different attack patterns from the data used to train the models.

Table 4.13 shows the evaluation of the designed machine learning models. As to be observed, C4.5 models achieve almost 99% accuracy with 99% recall and 99% for each experiment. Furthermore, it seems that the online processing times (testing time) of

the C4.5 models are also much lower than the other models. It should be noted here that the C4.5 model built by November-2008 dataset includes 25 rules, while the other C4.5 model built by April-2012 dataset includes 164 rules. These results are promising since they show auto-generated rules by a machine learning approach can enable one to categorize malicious traffic with a high accuracy in a reasonable time. Note that the values for training and testing time parameters shown in the Table 4.13 are approximate values.

Table 4.13: Evaluation of Machine Learning Models

Parameter	Train: 10% of Nov-2008 Test: 90% of Nov-2008			Train: 10% of April-2012 Test: 90% of April-2012		
	C4.5	Naive Bayes	AdaBoost	C4.5	Naive Bayes	AdaBoost
Accuracy	0.999	0.990	0.997	0.998	0.994	0.992
Precision	0.999	0.965	0.998	0.998	0.995	0.994
Recall	0.999	0.998	0.997	0.999	0.995	0.993
Specificity	0.999	0.999	1	0.999	0.986	0.99
F-measure	0.999	0.982	0.998	0.999	0.995	0.994
FPR	0.001	0.001	0	0.001	0.014	0.01
FNR	0.001	0.002	0.003	0.001	0.005	0.006
Training time	30 mins	8 mins	100 mins	10 mins	2 mins	35 mins
Testing time	40 mins	340 mins	60 mins	10 mins	45 mins	20 mins

The experiments in this section demonstrate that C4.5 classifiers present higher performance than Naive Bayes and AdaBoost.M1 classifiers in detecting one-way malicious traffic with a lower processing time. Therefore, C4.5 Decision Tree approach is used to build a classifier which is robust against changing attack patterns by employing the combined datasets shown in Table 3.2.

4.3.3 Detecting Recent Attack Traffic by C4.5 Decision Tree Trained by Old Traces

In this section of my experiments, new data combinations are created to represent older and more recent network traffic to observe how much of such attack traffic including changed patterns can be detected by using a machine learning based model.

Moreover, to find out the importance of feature selection for data preprocessing, four different feature sets for each newly created training set are generated as shown in Table 4.14. Note that the features generated by Chi-Square and Symmetrical Uncertainty algorithms are extracted from all available features. It should also be noted that only the top-5 most informative features generated by these algorithms are used, otherwise the integrated feature selection algorithm of C4.5 classifier (information gain ratio) would select similar features to construct the tree model.

Table 4.14: Selected Features

	Dataset			
	D1	**D2**	**D3**	**D4**
Chi-Squared Features	ip.proto frame.len frame.caplen tcp.stream ip.TTL	frame.caplen frame.len ip.ttl ip.checksum ip.proto	frame.caplen frame.len deltatime ip.ttl tcp.stream	frame.len ip.ttl deltatime ip.proto ack flag
Symmetrical Uncertainty based Features	ip.proto ip.checksum_bad ip.checksum_good ecn flag ack flag	ip.proto icmp.type ns flag icmp.code ecn flag	ip.proto icmp.type ns flag icmp.code ecn flag	frame.len ip.ttl ip.proto deltatime ack flag
My Proposed set of Features	see Feature set-2 in Table 3.5			
All available features	see Table 3.4			

Since the C4.5 models have provided better performances than the others in the experiments explained in the previous sections, the robustness and the rule generalization abilities of the C4.5 classifier are analyzed by building sixteen different C4.5 models, trained on four different datasets, D1, D2, D3 and D4, with four different feature sets, to run them on D5 to explore how powerful this classifier is against changing attack patterns. These experiments enable me to observe the performance and the robustness of the auto-generated rules by the C4.5 models on one-way darknet data (traffic traces).

Table 4.15 presents the performances of the aforementioned C4.5 classifiers on the D5 dataset which is the combination of April-2012 and Normal-2014 datasets. As to

be observed, the C4.5 classifier trained by the combination of backscatter-2008, DDoS-2007 and normal-2008 (D4) provides the highest accuracy with the highest F-measure value for each experiment. While the C4.5 classifier trained by the combination of backscatter-2008 and DDoS-2007 (D2) offers decreased accuracy and precision, it also offers high recall values, which means that while it provides a high detection rate, it also causes false alarms. On the other hand, the C4.5 classifier trained by the combination of backscatter-2008 and Normal-2008 (D3) offers high precision but low accuracy and recall values, which means it provides low detection rate with low false alarms. However, when the features obtained by Chi-square and Symmetrical Uncertainty algorithms are selected, this classifier offers high accuracy with higher recall and precision values but many rules, which might cause overfitting problems. These results are promising since they demonstrate that it is possible to reach high accuracies in detecting one-way malicious network traffic over a time period of seven years where the attack patterns change with time. Figure 4.8 shows the decision tree generated by the C4.5 classifier trained by D4 dataset with my proposed set of features.

Table 4.15: The Results of the Experiments on C4.5

	Train set	Accurary (%)	Recall (%)	Precision (%)	F-msr (%)	#of Leaves	Used Features by C4.5
All Features	*D1*	60.6	88.3	56.8	69.1	5	ip.checksum_bad ip.proto push flag
	D2	49.5	99.0	50.0	66.2	8	ip.checksum_bad frame.caplen push flag
	D3	65.1	30.2	100	46.5	1800	frame.caplen ip.proto push flag frame.len
	D4	81.1	81.9	61.8	70.4	29	frame.caplen ip.ttl frame.len frame.deltatime
My Proposed Set of Features	*D1*	24.2	29.3	26.6	28.0	5	ip.proto frame.len ack flag
	D2	50.0	60.0	50.0	54.5	25	ip.ttl ip.proto frame.len frame.deltatime ack flag
	D3	60.0	23.0	99.0	37.3	390	frame.len ip.ttl syn flag res flag
	D4	88.2	80.0	95.7	87.2	97	frame.len ip.ttl frame.deltatime syn flag ip.proto ack flag rst flag
Chi-Squared Features	*D1*	72.0	88.1	66.8	76.0	5	ip.proto frame.len
	D2	49.0	90.0	47.7	62.4	14	ip.checksum_bad frame.caplen ip.ttl ip.proto
	D3	86.7	78.5	95.3	86.1	164	frame.caplen frame.len ip.ttl
	D4	87.9	79.9	95.9	87.1	89	frame.len ip.ttl frame.deltatime ack flag ip.proto
Symmetrical Uncertainty based Features	*D1*	48.8	76.0	46.4	60.6	4	ip.checksum_bad ecn flag
	D2	50.1	80.2	51.5	62.8	9	ip.proto icmp.type ns flag
	D3	86.7	78.5	95.3	86.1	164	frame.caplen frame.len ip.ttl
	D4	87.9	79.9	95.9	87.1	89	frame.len ip.ttl frame.deltatime ack flag ip.proto

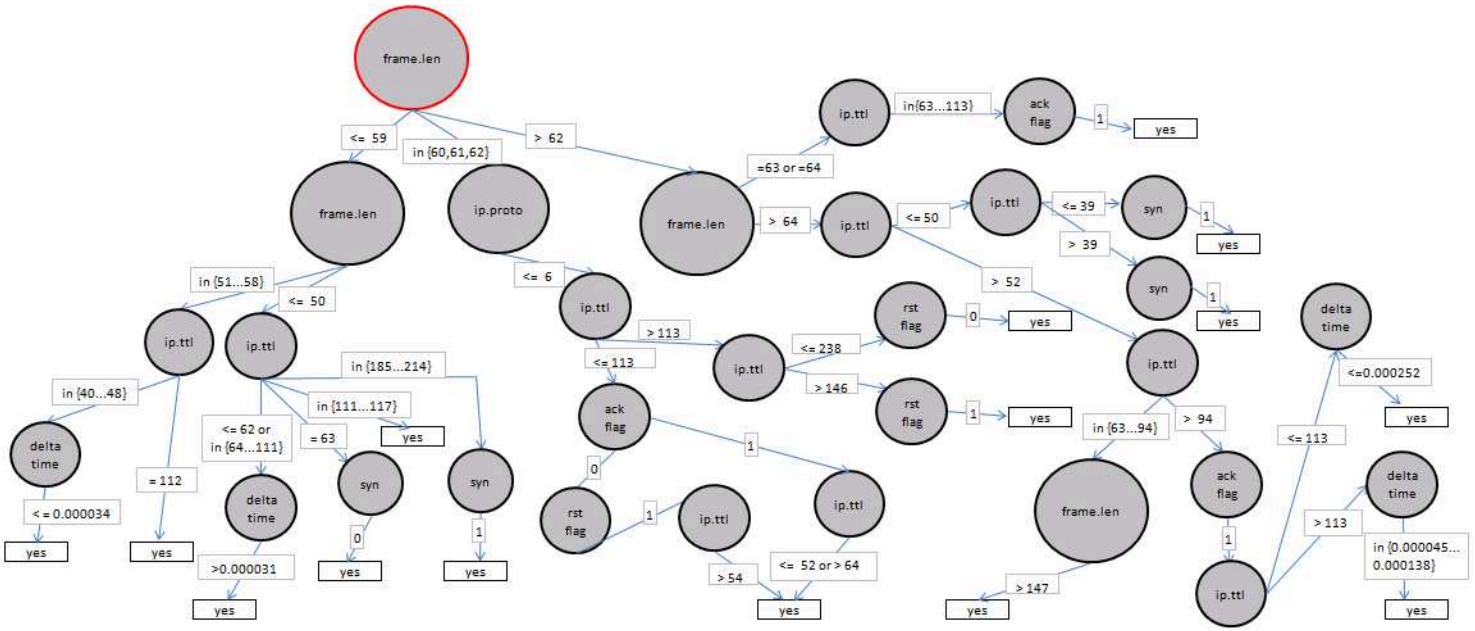

Figure 4.8: Decision Tree built by D4 with my proposed set of features

Chapter 5

Conclusion

In this thesis, the performances of five network traffic monitoring systems, namely Snort, Bro, Iatmon, Corsaro and Cisco ASA 5515-X, as well as three machine learning models, namely C4.5 Decision Tree, Naive Bayes and AdaBoost.M1, are evaluated in terms of analyzing one-way darknet traffic, processing time and robustness against malicious behaviours changing over time. To achieve this, six darknet datasets collected in different years (from 2004 to 2012) via a passive darknet in addition to two datasets including normal traces collected in 2008 and 2014 are employed to perform the experiments on the aforementioned systems and techniques. The nature of the aforementioned traffic is also explored to discover the general darknet (backscatter) trends and how they change over time. All the datasets employed are collected by CAIDA and they are all publicly available, which ensures that this thesis can be easily validated and compared against other researches in the field.

Furthermore, the importance of the properties of a training dataset, such as size and selected features, are studied to understand how they affect the quality of a machine learning classifier. To this end, different feature sets are generated depending on the patterns revealed after analyzing the nature of the backscatter traffic. Furthermore, two well-known feature selection methods, namely Chi-Square and Symmetrical Uncertainty, are used while constructing a C4.5 classifier with different sizes of training sets. All the experiments on machine learning classifiers are performed by using Weka on a machine having 32 GB RAM, Intel i5 3.10 GHz CPU and Ubuntu 14.04 operating system.

The results show that the auto-generated rules by C4.5 Decision Tree classifier presents 99% detection rate while analyzing such datasets having the similar patterns to the one used in training phase, whereas it presents almost 90% detection rate while analyzing such traffic including different patterns. The following list summarizes the results of the experiments performed in this thesis:

- TCP is the top used transport layer protocol while HTTP is the top used application layer protocol in generating backscatter attack. Moreover, secure ports and peer-to-peer applications are not preferred to generate backscatter traffic. Also, it seems that source addresses from China and USA play the major role of generating such attacks in the datasets employed in this thesis.

- The patterns of backscatter traffic show changes over time. Therefore, pre-defined rules/signatures become ineffective to analyze backscatter traffic from one year to the other.

- The network traffic monitoring tools employed mostly benefit from TCP and ICMP based rules to define suspicious packets. The main challenge of these tools is that they may not be able to detect malicious activity having such patterns that do not match with any of the pre-defined rules. In other words, these tools are not robust against changing attack patterns.

- Corsaro has a higher detection rate with a lower processing time than the other network traffic monitoring tools that are employed in this thesis. However, it should be noted that the detection rate of Corsaro decreased to 23% from 99% when a more recent darknet dataset (April-2012) was used instead of an older one (November-2008). It is worth bearing in mind that one-way darknet traffic traces without any payload information are used in this thesis.

- C4.5 Decision Tree classifier achieves higher performance than the other classifiers in terms of both the detection rate and the processing time. Although the general attack patterns change over time, C4.5 classifiers help exploring the resemblances between the changing attack patterns in the analyzed datasets. This enables one to detect such recent attack patterns via a C4.5 classifier trained on older traffic patterns. This demonstrates that an intrusion detection system using a machine learning technique to automatically generate or update its rules can overcome the challenge of generalizing to changing attack patterns.

- The C4.5 classifiers where the features "frame.len", "ip.proto" and "ip.ttl" are used in the training phase show the highest accuracies, whereas TCP flags are commonly used in filtering backscatter traffic both in the literature and by the

employed network traffic monitoring systems. Also, it is demonstrated that using source or destination IP addresses and port numbers are not required to reach high accuracies in detecting backscatter traffic.

- It is possible to build an efficient C4.5 classifier by using a small amount of well-represented training set. This enables machine learning techniques to be used in practice since it is easier to label smaller datasets for a human expert.

Future work will explore the performances of other network traffic analysis tools and machine learning models on larger and more recent darknet datasets in detail. Moreover, dynamic adjustment of composite features and generalization and robustness capabilities of other classification systems will be studied. Finally, clustering approaches while employing a training set will be applied to increase its consistency.

Bibliography

[1] *The CAIDA UCSD - DDoS Attack 2007 Dataset*. http://www.caida.org/data/passive/ddos-20070804_dataset.xml.

[2] *The CAIDA UCSD Anonymized Internet Traces 2008*. http://www.caida.org/data/passive/passive_2008_dataset.xml.

[3] *The CAIDA UCSD Anonymized Internet Traces 2014*. http://www.caida.org/data/passive/passive_2014_dataset.xml.

[4] *The CAIDA UCSD Backscatter-2004-2005 Dataset - 28 May 2004*. http://www.caida.org/data/passive/backscatter_2004_2005_dataset.xml.

[5] *The CAIDA UCSD Backscatter-2006 Dataset - February 2006*. http://www.caida.org/data/passive/backscatter_2006_dataset.xml.

[6] *The CAIDA UCSD Backscatter-2008 Dataset Nov, 2008*. http://www.caida.org/data/passive/backscatter_2008_dataset.xml.

[7] *The CAIDA UCSD Network Telescope Educational Dataset*. http://www.caida.org/data/passive/telescope-educational_dataset.xml.

[8] *Cisco Adaptive Security Appliance (ASA) Software*. http://www.cisco.com/c/en/us/products/security/adaptive-security-appliance-asa-software/index.html.

[9] *Cisco ASA 5500-X Series Adaptive Security Appliances*. http://www.cisco.com/web/TH/solutions/smb/velocity/Security/ASA_5500_X_Series.html.

[10] *Geolite*. http://dev.maxmind.com/geoip/legacy/geolite/.

[11] *Port 1026*. http://kb.prismmicrosys.com/evtpass/evtpages/PortNo_1026_nterm_55033.asp.

[12] *SecureShell*. http://www.rfc-archive.org/getrfc.php?rfc=4251.

[13] *Snort*. http://www.snort.org/.

[14] *Tshark*. http://www.wireshark.org/docs/man-pages/tshark.html.

[15] *Waste*. http://waste.sourceforge.net/.

[16] *Wireshark*. http://www.wireshark.org/.

[17] Hall Mark A. *Correlation-based feature selection for machine learning.* The University of Waikato, Diss, 1999.

[18] Riyad Alshammari and A. Nur Zincir-Heywood. *Can encrypted traffic be identified without port numbers, IP addresses and payload inspection?* Computer networks 55.6: 1326-1350, 2011.

[19] Michael Bailey, Evan Cooke, Farnam Jahanian, Andrew Myrick, and Sushant Sinha. *Practical darknet measurement.* Information Sciences and Systems, 40th Annual Conference on pp. 1496-1501, IEEE, 2006.

[20] Cui-Mei Bao. *Intrusion detection based on one-class svm and snmp mib data.* Information Assurance and Security, Fifth International Conference on. Vol. 2, 2009.

[21] Irwin Barry and J-P. van Riel. *Using inetvis to evaluate snort and bro scan detection on a network telescope.* VizSEC 2007. Springer Berlin Heidelberg:255-273, 2008.

[22] Thomas Berger. *Analysis of current VPN technologies.* In Availability, Reliability and Security, pp. 8. IEEE, 2006.

[23] Hal Berghel. *Malware month.* Communications of the ACM, 46.12: 15-19, 2003.

[24] Nevil Brownlee. *One-way traffic monitoring with iatmon.* Passive and Active Measurement. Springer Berlin Heidelberg, 2012.

[25] CAIDA. *The UCSD Network Telescope.* http://www.caida.org/projects/network_telescope/.

[26] Ashley Carman. *Lizard Squad begins selling DDoS tool for commercial use.* http://www.scmagazine.com/lizard-stresser-tool-sold-online/article/390558/, 2015.

[27] Paquet Catherine. *Implementing Cisco IOS Network Security (IINS).* Cisco Press, 2009.

[28] Jin Cheng, Haining Wang, and Kang G. Shin. *Hop-count filtering: an effective defense against spoofed DDoS traffic.* Proceedings of the 10th ACM conference on computer and communications security, 2003.

[29] Fachkha Claude, Elias Bou-Harb, and Mourad Debbabi. *Towards a Forecasting Model for Distributed Denial of Service Activities.* 12th IEEE International Network Computing and Applications, 2013.

[30] Fachkha Claude, Elias Bou-Harb, and Mourad Debbabi. *Fingerprinting Internet DNS Amplification DDoS Activities.* New Technologies, Mobility and Security (NTMS), 2014 6th International Conference on. IEEE, 2014.

[31] Marcos Colon. *Tidal waves of spoofed traffic: DDoS attack*. http://www.scmagazine.com/tidal-waves-of-spoofed-traffic-ddos-attacks/article/393059/, 2015.

[32] Frank Eibe, Mark Hall, and Bernhard Pfahringer. *Locally weighted naive bayes*. Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc, 2002.

[33] Dainotti Alberto et al. *Estimating internet address space usage through passive measurements*. ACM SIGCOMM Computer Communication Review 44.1: 42-49., 2013.

[34] Livadas Carl et al. *Using machine learning technliques to identify botnet traffic*. 31st IEEE Local Computer Networks, 2006.

[35] Saad Sherif et al. *Detecting P2P botnets through network behavior analysis and machine learning*. Ninth Privacy, Security and Trust (PST) Annual International Conference on. IEEE, 2011.

[36] Xia Fen et al. *Ranking with decision tree*. Knowledge and information systems 17.3:381-395, 2008.

[37] Masashi Eto, Daisuke Inoue, Jungsuk Song, Junji Nakazato, Kazuhiro Ohtaka, and Koji Nakao. *nicter: a large-scale network incident analysis system: case studies for understanding threat landscape*. In Proceedings of the First ACM Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (pp. 37-45), 2011.

[38] C. Fachkha, E. Bou-Harb, A. Boukhtouta, S. Dinh, F. Iqbal, and M. Debbabi. *Investigating the dark cyberspace: Profiling, threat-based analysis and correlation*. IEEE Risk and Security of Internet and Systems (CRiSIS), 7th International Conference on (pp. 1-8)., 2012.

[39] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. *The WEKA Data Mining Software: An Update*. hIGKDD Explorations, Volume 11, Issue 1, 2009.

[40] Thomas Karagiannis, Andre Broido, , and Michalis Faloutsos. *Transport layer identification of P2P traffic*. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pp. 121-134. ACM, 2004.

[41] Alistair King and Alberto Dainotti. *Corsaro*. www.caida.org/tools/measurement/corsaro, 2012.

[42] John Kristoff. *Ops: TCP port 1024 and 3072 traffic*. http://www.cymru.com/jtk/blog/2011/03/04/.

[43] NSFOCUS Ltd. *Mid-Year DDoS Threat Report 2013*. http://www.nsfocus.com/SecurityReport/, 2013.

[44] Farid Dewan Md, Nouria Harbi, and Mohammad Zahidur Rahman. *Combining naive bayes and decision tree for adaptive intrusion detection.* CoRR, vol. abs/1005.4496, 2010.

[45] Khamruddin Md and Ch Rupa. *A rule based DDoS detection and mitigation technique.* Nirma University International Conference on. IEEE, 2012.

[46] Rich Miller. *DoS attacks on blogs.* http://news.netcraft.com/archives/2006/02/28/, 2006.

[47] Rich Miller. *DoS attacks on payment gateways.* http://news.netcraft.com/archives/2006/02/10/, 2006.

[48] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. *Inferring internet denial-of-service activity.* Transactions on Computer Systems (TOCS) 24.2: 115-139, ACM, 2006.

[49] Jose Nazario. *DDoS Events of Note: WordPress, Gambling Sites.* http://www.arbornetworks.com/asert/2008 /02/ddos-events-of-note-wordpress-gambling-sites/, 2008.

[50] C. Lee Noh, K. Choi, and G. Jung. *Detecting distributed denial of service (DDoS) attacks through inductive learning.* Lecture Notes in Computer Science, vol. 2690, pp. 286295, 2003.

[51] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. *Characteristics of internet background radiation.* In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, 2004.

[52] Vern Paxson. *Bro: a system for detecting network intruders in real-time.* Computer networks 31, no. 23: 2435-2463, 1999.

[53] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. *Accurate, scalable in-network identification of p2p traffic using application signatures.* In Proceedings of the 13th international conference on World Wide Web, pp. 512-521. ACM, 2004.

[54] Seungwon Shin, Jaeyeon Jung, and Hari Balakrishnan. *Malware prevalence in the KaZaA file-sharing network.* Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, 2006.

[55] Fawcett Tom. *An introduction to ROC analysis.* Pattern recognition letters 27.8: 861-874, 2006.

[56] Qian Wang, Zesheng Chen, and Chao Chen. *Darknet-based inference of Internet worm temporal characteristics.* Information Forensics and Security, IEEE Transactions on 6.4: 1382-1393, 2011.

[57] Lee Wenke, Salvatore J. Stolfo, and Kui W. Mok. *Adaptive intrusion detection: A data mining approach*. Artificial Intelligence Review 14.6: 533-567, 2000.

[58] E. Wustrow, M. Karir, M. Bailey, F. Jahanian, and G. Huston. *Internet background radiation revisited*. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp. 62-74, 2010.

[59] Vinod Yegneswaran, Paul Barford, and Dave Plonka. *On the design and use of Internet sinks for network abuse monitoring*. Recent Advances in Intrusion Detection. Springer., 2004.

[60] Yang Yiming and Jan O. Pedersen. *A comparative study on feature selection in text categorization*. ICML. Vol. 97, 1997.

[61] Freund Yoav and Robert E. Schapire. *Experiments with a new boosting algorithm*. ICML. Vol. 96, 1996.