AN ENTROPY BASED DDOS DETECTION METHOD AND IMPLEMENTATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

SÜLEYMAN FÜRKAN YÜCEBAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
CYBER SECURITY

DECEMBER 2019

Approval of the thesis:

AN ENTROPY BASED DDOS DETECTION METHOD AND IMPLEMENTATION

Submitted by SÜLEYMAN FÜRKAN YÜCEBAŞ in partial fulfillment of the requirements for the degree of **Master of Science in Cyber Security Department, Middle East Technical University**, by

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics** _____

Assoc. Prof. Dr. Aysu Betin Can
Head of Department, **Cyber Security** _____

Assoc. Prof. Dr. Aysu Betin Can
Supervisor, **Information Systems Dept., METU** _____

Examining Committee Members:

Asst. Prof. Dr. Aybar Can Acar
Health Informatics Dept., METU _____

Assoc. Prof. Dr. Aysu Betin Can
Information Systems Dept., METU _____

Prof. Dr. Kemal Bıçakcı
Computer Engineering Dept., TOBB ETU _____

Date: **09/12/2019**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: SÜLEYMAN FÜRKAN YÜCEBAŞ

Signature : _____

ABSTRACT

AN ENTROPY BASED DDOS DETECTION METHOD AND IMPLEMENTATION

YÜCEBAŞ, SÜLEYMAN FÜRKAN

MSc., Department of Cyber Security

Supervisor: Assoc. Prof. Dr. Aysu Betin Can

December 2019, 56 pages

Distributed Denial of Service (DDoS) is a cyber attack type involving multiple computer sources which aims to temporarily or permanently deactivate the service provided by a device. This attack type has been listed multiple times as the most used attack types and has a great portion in all types of cyber attacks. Also, these attacks are increasing day by day and poses a threat for cyber security ecosystem. In today's world, these attacks target world-wide organizations and cause them to suffer. DDoS attacks are easy to employ but hard to prevent. There are various methods to decrease the impact of attacks but none of them are exact solutions. With further research about DDoS detection approaches, it is observed that, methods using statistical approaches have better performance than other approaches.

In this thesis, we describe an entropy based detection method and implement our method on software defined networks (SDN). The performance of the method is evaluated for various attack types. We propose the use of multiple entropy values and a novel alarm determination based on these entropy values. We conducted a series of experiments with real datasets for four different attack types to evaluate our method. We compare the effectiveness of our entropy parameter selection (5 single attributes and 10 pair of attributes) to entropy calculation with all 3 elements and 4 elements subsets. The results show that our method detects most common attack types at very early stages.

Keywords: DoS, DDoS, SDN, entropy, detection methods

ÖZ

ENTROPİ TABANLI DDOS TESPİT YÖNTEMİ VE UYGULAMASI

Yücebaş, Süleyman Fürkan
Yüksek Lisans, Siber Güvenlik Bölümü
Tez Yöneticisi: Doç. Dr. Aysu Betin Can

Aralık 2019, 56 sayfa

Dağıtılmış hizmet reddi saldırıları (DDoS), bilgisayar ağlarında bulunan bir cihazın sağladığı hizmetin birden fazla farklı kaynaklar kullanılarak geçici veya kalıcı olarak servis dışı bırakılıp ulaşımının engellendiği bir saldırı tipidir. DDoS saldırıları, mevcut siber saldırıların başında gelmektedir ve geçtiğimiz her gün bu saldırı tipinin kullanımı artarak siber dünya için büyük bir tehdit oluşturmaktadır. Günümüzde DDoS saldırıları büyük organizasyonları hedef alıp çok geniş zararlara neden olmaktadır. Bu saldırıları uygulamak kolay fakat tespit edip engellemek zordur. Bu alanda birden fazla yaklaşım üzerinde çalışmalar yapılmaktadır fakat henüz kesin bir çözüm sunulmamıştır. Yapılan araştırmalar, istatistiksel bazlı tespit yöntemlerinin DDoS saldırı tespitinde başarılı sonuçlar verdiğini göstermektedir. Bu tezde, entropi bazlı saldırı tespiti yapan bir yöntem sunulup farklı atak tipleri için yazılım tanımlı ağ üzerinde performansı değerlendirilmiştir. Sunulan yöntem, saldırı tespiti için çoklu entropi değerlerinin kullanılmasını ve bu entropi değerlerine dayanan yeni bir alarm sistemi önermektedir. Sunduğumuz yöntemi değerlendirmek üzere, dört farklı atağa karşı gerçek trafik setleri kullanılarak bir dizi deney gerçekleştirildi. Yaptığımız çalışmada entropi hesaplanmasında önerilen 5 farklı paket parametresi ve bunların ikili kombinasyonlarının, 3'lü ve 4'lü kombinasyonlarına karşı verimliliği karşılaştırılmıştır. Sonuçlar, metodumuzun yaygın olan atak tipleri için saldırıyı erken aşamalarda tespit ettiğini göstermektedir.

Anahtar Sözcükler: DoS, DDoS, SDN, entropi, tespit yöntemi

To my beloved brother, Efekan…

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Assoc. Prof. Dr. Aysu Betin Can for her support, encouragement and endless guidance at every stage of my thesis.

I would like to thank my manager and our faculty member Dr. Attila Özgit for allowing me the time and his support to complete this work.

I thank İbrahim Ercan for providing attack datasets which were used in experiment session.

I would also like to thank my friends Efe Erdil, Naz Özcan, Emre Mülazimoğlu, Göksu Deniz Aksak and Esra Metin for their contribution to the thesis and being a great support in my life with their existence.

Last but not least, this thesis would not have been possible without the support and love of family, especially my brother Efekan, my parents, Çetin and Hatice Yücebaş, and finally my aunt Ayşe İyigün who have supported me through all my life, and for whom my words of gratitude would not suffice.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DDoS	Distributed Denial of Service
DNS	Domain Name System
IP	Internet Protocol
SDN	Software Defined Networking
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
TP	True Positive
TN	True Negative
FP	False positive
FN	False Negative

CHAPTER 1

INTRODUCTION

Cyber attack is an attempt to gain illegal access to a computer or computer system for the purpose of causing damage or harm. One type of these attacks is Distributed Denial of Service (DDoS) attacks. DDoS is a cyber attack which targets a specific online service and aims to disable the service by overwhelming it with traffic from multiple sources. Attackers target a wide range of assets, from banks to government websites. They build “botnets” which are networks of computers infected by attackers via spreading malicious software through various methods such as e-mails or fake software. Once a system is infected, these machines are controlled from outside the system, without the system owners' knowledge, and used like an army to launch an attack against any target.

According to Kaspersky Lab report, the number of a distributed denial of service (DDoS) attacks have been increased more than 2.5 times over the last 3 years before 2018. In 2018, however, the number of DDoS attacks reduced 13% compared to the previous year. Despite the decrease in the attack numbers, total damage caused because of DDoS attack has been increasing. The average length of an attack climbed from 95 minutes in the first quarter to 218 minutes in the fourth quarter [1]. If the attacks cannot be detected and mitigated within a short period of time, financial and reputational losses of the firms increase accordingly. These changes show the growing danger of DDoS attacks; hence, necessary precautions should be taken for this threat.

With the advancements in technology, traditional networks have also changed. In 2004, research on new management paradigms was started. Research has yielded successful results and first software defined networking (SDN) is implemented in 2008 [3]. With this development, modern network paradigm has begun with software defined networking. However, DDoS attacks are also a serious threat to SDNs as well as traditional networks.

Software-defined networking (SDN) is a new network technology that decouples the control plane and forwarding plane in order to make networks programmable, agile and flexible. These features bring many advantages for users. As it brings favorable advantages for network management, SDN usage has been increased and will possibly replace with

traditional networking. Leading network companies [4] have been involved in this technological development and have started to work in this field. However, SDN technology has some disadvantages in terms of security. Because of SDN's nature, this type of systems is highly vulnerable to DDoS attacks. The connection between the controller and other devices can be deprecated, consuming resources with flooding malicious packets by attackers. As a result, the controller may be compromised, and the entire network structure may be corrupted or become out of service. There are several studies on DDoS detection and mitigation methods on SDN such as [13], [14], [15], [18]. However, there is still room for improvement. In [23], the authors show that entropy based DDoS detection methods performs better than other approaches.

In this thesis, we describe an entropy based detection method and propose to use it on software defined networks. Our entropy based detection method propose the use of multiple entropy values and a novel alarm determination based on these entropy values. We implemented a python software that runs on SDN network to collect all incoming packets from Openswitch using POX controller. Different packet field parameters are examined in entropy calculation to evaluate the performance of our method.

We conducted a series of experiments to evaluate our implementation on SDN using an instant virtual network called Mininet with 235 hosts. We replayed previously captured 4 different pcap files consisting of attack packets on real network traffic dataset [50] that is used as benign traffic. We evaluate our implementation with respect to different victim hosts that have different traffic rates, under two different protocol attacks (TCP and UDP) and under single source address and distributed source address of the same attack. In addition, we compare the effectiveness of our entropy parameter selection (5 single attributes and 10 pair of attributes) to entropy calculation with all 3 elements and 4 elements subsets. Finally, we show the effectiveness of proposed alarm mechanism to detect any anomaly detection in the entropy values. According to evaluation results, the proposed method can detect various DDoS attacks at a very early stage.

The following sections are organized as follows: In Chapter 2, background information on DDoS, attack types and detection methods are given. In Chapter 3, our proposed detection method and implementation is explained in detail. Chapter 4 presents a series of experiments conducted to evaluate our implementation on SDN using an instant virtual network called Mininet[1]. In Chapter 4, experiments results are given. Finally, Chapter 5 concludes our work and discusses future work.

[1] http://mininet.org/

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1. DDoS Overview

The main objective of DoS attacks is preventing legitimate use of a service, which is achieved by overloading server, network or service resources. As an attempt to make the attack more effective, DoS attacks can be executed in a coordinated way, using a large number of clients. This attack type is called DDoS, as it uses distributed sources to target a service. These distributed sources are often geographically separated; thus, it is hard to detect the source of the attack and block malicious traffic. Since DDoS attacks require coordination of the clients, they are mostly executed by using botnets. A botnet is a sheer number of zombie computers (bots), all controlled by a command and control server.

Considering the enormous number of devices that are connected to the internet, and recent development of internet infrastructure, DDoS attacks can reach extreme level of network power. In February 2018, GitHub was the target of a DDoS attack which reached 1.35 Tbps. This was an amplification attack that abused Memcached instances, which originated from more than a thousand different autonomous systems across tens of thousands of unique endpoints [24]. A DDoS attack using the Mirai [25] botnet is another major instance, which peaked to 1.1 Tbps volume, and affected a significant part of the internet services in October 2016. Mirai IoT botnet commanded almost 100,000 bots, which are various devices such as cameras, home routers, DVRs, and printers, all exploited due to their poor security configurations [22].

2.2. DDoS Attack Types

There are different types of DDoS attacks, so it is essential to classify these attacks in order to understand these attacks better. Classifications of DDoS attacks are proposed in several studies such as Mirkovic et al. [26] and Lee [27]. A combined classification of DDoS attacks which is consisted of two levels. The first level classifies the attacks according to their degree of automation, exploited vulnerability, attack rate dynamics, and their impact. The second level recognizes the specific characteristics of each first level categories. We classify DDoS attacks in terms of exploited vulnerability. In the next sections this classification is explained in detail.

2.2.1. Flood Attack

A flood attack is often performed by using large number of bots of a botnet to send huge amount of traffic to a victim system. The aim is to saturate the targeted system's network bandwidth with this high volume of IP traffic, thus slowdown or crash the victim host so that it cannot respond to legitimate requests [28]. Flood attacks can be executed using UDP and ICMP packets.

UDP Flood

UDP Flood attacks are executed by targeting large number of UDP packets to a victim system. Consequently, it exhausts the network resources and consume all the available bandwidth, thus preventing the victim server to answer legitimate service requests. Although the UDP packets can be sent to random or specified ports of the targeted system, typically the DDOS UDP Flood attacks are designed to target random ports. When a targeted system receives a UDP packet, the service accepting connections on the specified port is identified by the system, however, if there is no service running on the system for that particular port, the server generates an ICMP packet of "destination unreachable" [29] and sends it to the source address, which is typically forged. As the resources of the targeted system is limited, by sending sufficient number of packets, the victim system will not be able to respond to any legitimate request, or even fail to operate at all. Typically, attacker spoofs the source IP addresses of the attacking packets, so that the real attackers are kept hidden and the zombies, which are often weak clients, would not receive response that would hinder their network resources. [23]

TCP Flood / SYN Flood

TCP Flood attacks exploits TCP connection sequence by sending TCP traffic using different flags. TCP connections originated from a client to a server are established with three-way handshake technique by sending packets with SYN, SYN-ACK, ACK flags between the client and the server. When a TCP SYN Flood attack is performed, the victim server receives numerous SYN packets, and replies these requests with SYN-ACK packets. After this, the victim server waits for an ACK packet from the clients, and in order to do so it reserves system resources to manage a table to keep connection status for each request. Since the source address is typically spoofed, the victim system will not receive the ACK packets. When the number of open connections is large enough, the system will run out of resources to accept new connections, so that the denial of service occurs. [28]

ICMP Flood

In ICMP Flood attacks, the attacker sends an enormous number of ICMP_ECHO_REPLY (ping) packets to the victim host. The objective is to fill the incoming bandwidth of the victim, or outgoing bandwidth of the system by making it try to respond to all the ping requests, and even exhaust the system so that it will not be able to respond to legitimate requests.

2.2.2. Amplification Attack

In amplification attacks, an attacker, who typically controls a botnet, uses means to multiply the attack power of the agents (bots) to target a victim host. This means is an application factor that magnifies the bandwidth amount of the attack, so that the effect of the attack becomes even more severe. These attacks use spoofed source addresses so they can be also considered as reflection attacks. To be able to set the message with a spoofed source IP address, protocols such as UDP and ICMP are used as they do not require a 3-way handshake. Amplification attacks can be conducted by exploiting vulnerable network amplifiers (Smurf and Fraggle attacks) or application servers (DNS and NTP amplification attacks).

In amplification attacks that use network amplifiers, the attacker or attack agents exploit the broadcast IP address feature in vulnerable routers. In essence, the attack agents send messages to broadcast IP (i.e., x.x.x.255) with messages, of which the source IP addresses is spoofed by forging the IP address of the target machine, so that the router service these packets across the network. As a result of this, the malicious traffic triggered by the attacker and produced by the router hinders the network bandwidth [23]. Also, when the source address is spoofed, the broadcasted message makes all the hosts in the network to send a message to a specific target host. With this attack, the amplification factor is proportional to the number of alive hosts in the network. Note that, this attack occurred in old routers, and current routers do not have these vulnerabilities. Amplification attacks that use application servers are application layer attacks, which triggers the application with a request and make the server to reply with a much larger packet. This amplification factor is dependent on the protocol and the request, and when the request is spoofed, the server itself sends this magnified traffic to the target host instead of the attacker agents. Together with the bandwidth power of botnets, amplification attacks can reach very high throughputs.

Smurf Attack

This attack is executed by the Smurf malware, which generates spoofed ICMP packets set to IP address of the target as the source IP address, and send these ping packets to IP broadcast address of an intermediate network. In this way, all the connected hosts in the network replies with ICMP_ECHO_REPLY packets to victim host. Since the attack factor is multiplied by the number of alive hosts in the network, and with the large number of the attacking agents, the target host's bandwidth is saturated, and the host can be overwhelmed so that it cannot respond to legitimate requests. [22] [23] [30]

Fraggle Attack

Similar to Smurf attacks, this attack uses network amplifiers and spoofed IP addresses, however, instead of ICMP, Fraggle attacks use UDP echo packets. Fraggle attacks causes larger attack traffics and more severe effects on the victim host compared to a Smurf attack [23].

DNS Amplification Attack

In this attack the attacker exploits the open DNS resolvers, which do not enforce access control on public requests and supports open recursive relay [22]. Considering the thousands of open recursive DNS resolvers and rouge DNS servers in the world, this attack type is still a significant threat to the internet. Typically, the attacker spoofs the source IP addresses of the DNS requests coming from the attacking agents as the IP address of the targeted host, so that the replies to the DNS queries are reflected to the victim system. The amplification factor differs on the query type, for example when the DNS server is queried for any resource record the query size is 64 byte and the response size is more than 3 KB, which means the bandwidth size of the attacker is multiplied with more than 50 amplification factors [31]. As a result, the botnet bandwidth power combined with the amplification factor of the DNS attack can result severe attacks that can deplete the bandwidth of the victim system and cause a denial of service. For example, in March 2013, a DNS amplification type DDoS attack was executed that targeted Spamhaus project [22]. According to Nexusguard's 2019 quarter 2 threat report [32], DNS amplification attacks have risen 1000% since 2018, and ironically, the main reason behind this is the adaptation of DNSSEC which generates larger responses and grater amplification factors.

NTP Amplification Attack

In this attack, the attacker exploits open NTP servers to amplify their attacks. Similar to DNS amplification attacks, NTP amplification attacks use amplified UDP traffic which are reflected to victim host. Publicly accessible vulnerable NTP servers can respond to queries such as "monlist", "version", and "showpeers" with a large size of packets. For example, monlist command returns the last 600 clients connected to the server in a very large file compared to the query size which yields the amplification factor of 4x to 15x for the attack [33]. With the bandwidth power of the attacker's botnet and amplification factor which is reflected to the victim host thanks to spoofed UDP queries, the power of the attack can escalate quickly and result denial of service of the target.

2.2.3. Protocol Exploit Attack

This attack exploits a particular characteristic of the design or implementation of a protocol running on the target system, aiming to make the victim exhausts its own resources [23]. TCP SYN attack and PUSH + ACK attack are two examples of protocol exploit attacks.

PUSH + ACK ATTACK

In this attack, attacker targets the victim system by sending TCP packets, of which the PUSH and ACK bits are set to one. This TCP packet header causes the victim host to unload all data in the TCP buffer and respond with an acknowledgement [28]. When this attack is performed with very large number of agents, such as bots, the target host would not be able to process all this traffic with its limited resources and stop responding to legitimate requests or crash eventually.

2.2.4. Malformed Packet Attack

In this attack, attacker agents send malformed packets to the victim in an attempt to crash the target system [27]. Malformed packet attacks can be performed in two ways: IP address attack and IP packet options attack. In the IP address attack, the attacker sends a packet of which the source IP address is spoofed, so that both source IP address and destination IP address of the packet made the same. By doing so, the attacker expects the target system get confused by these packets and crash. In the IP packet options attack, the attacker sends the IP packet so that its optional fields are randomly filled, and its quality of service bits are set to one. As a result of this, in order to analyze the traffic, victim system would require extra processing resources. For IP packet options attack to be effective, an attacker often utilizes a botnet so that enormous number of bots perform this attack, which causes the exhaustion of the victim system's resources, thus crash the target [23].

2.3. DDoS Detection Methods

In previous sections we have discussed how DDoS is one of the most significant threats to the internet services. In order to mitigate this threat, it is necessary to detect the attack in the first place. Thus, DDoS detection methods is a well-studied area. To be an effective mitigation tool, detection methods should have characteristics such as short detection time, low false-positive rate (FPR), and low false-negative rate (FNR). Moreover, DDoS detection methods can be classified into four approaches: statistical methods, knowledge-based methods, soft computing-based methods, and data mining and machine learning methods.

2.3.1. Statistical Methods

The characteristics of IP packets and traffic generated by a DDoS attack often makes it possible to distinguish attack traffic from normal traffic. Statistical properties of certain fields in IP packet headers, such as the entropy of the source IP addresses, or statistical properties of normal and attack traffic patterns can be used to detect DDoS traffic. In this approach, a statistical model for a normal traffic is generated, and new traffic is statistically evaluated according to this model to decide if it suits the model or marked as anomaly if does not.

In [36], the authors developed a distributed change point detection mechanism to detect DDoS flooding attacks at an early stage by using a technique called Change Aggregation Tree (CAT). The network traffic pre-change and post-change were defined via a nonparametric approach to Cumulative Sum (CUSUM) sequential analysis technique. Typically, the cumulative deviation of the router-level traffic flows during a DDoS attack is higher than normal traffic flow. Accordingly, CAT system is designed to detect sudden changes in the router-level traffic flows. The CATs that represent the attack flow patterns are created by the domain server where the traffic change patterns are detected. Proposed scheme achieved a 98% detection rate with less than 1% false positive rate in a DETER testbed.

In [37], authors monitored the increase of new source IP addresses in incoming traffic to detect bandwidth depletion attacks. This way they provided a more effective approach to detect highly distributed DoS attacks compared to approaches where traffic volume is monitored. Their technique utilizes a sequential nonparametric change detection method, CUSUM, to provide demonstrably high accurate detection without the need of detailed normal and attack traffic models.

In [38], authors utilized specific characteristics of DDoS attacks like abrupt traffic change, flow dissymmetry, distributed source IP addresses, and concentrated target IP addresses for the IP Flow Feature Value (FFV) algorithm they proposed. In their work, Cheng et al. used a linear prediction technique to construct an efficient ARMA prediction model for normal traffic flow. Consequently, with the linear prediction model and anomaly detection techniques a DDoS detection method is designed with demonstrably accurate results with low false positive rate.

In [39], authors present LOT, which is a practical lightweight secure tunneling protocol that is deployed at network level communication gateways, for the purpose of mitigating DDoS attacks by preventing IP spoofing and flood attacks from specific networks. Thanks to the plug and play feature of the protocol, two gateways that run LOT can automatically detect each other and set up a tunnel between each other for secure communication. Accordingly, the tunnel enables the gateways to discard spoofed IP packets whose source IP address is in tunneled in the other gateway and detect congesting traffic.

In [40], the authors proposed a detection technique for DDoS attacks with an approach based on game theory. In their proposed model of the DDoS attack, they explored several characteristics with regard to malicious traffic distribution and the number of attackers. They proved that a particular optimal strategy of defense against DDoS attacks is viable, where attacker payoff for rational or irrational attack agents are set as the upper boundaries.

In [17], the authors proposed joint entropy based DDoS defense mechanism scheme in SDN. In their proposed defense scheme, entropy value is calculated for all possible k-element subsets of P where P= {IPsrc, IPdst, Psrc, Pdst, Prot, PKTsize, TTL, TCPflag} for $1 \leq k \leq P$. They use score-based packet marking system for filtering the suspicious packets. The method uses packet-based sliding window with 1000 packets. The method marks the packets as suspicious traffic whenever any current entropy value falls below the threshold value. They use runtime threshold and it is calculated by dividing the sum of current entropy and previous entropy by 2.

2.3.2. Knowledge Based Methods

Knowledge based methods utilize predefined rules or predetermined attack patterns from available datasets or information about relevant attacks to evaluate traffic for DDoS detection. In this approach, signatures of existent DDoS attack traffic are used to construct a detection knowledge. If the traffic characteristic matches the rules or signatures, it means an anomaly is detected and an alarm is produced.
In [41], the authors proposed a DDoS mitigation mechanism called NetBouncer, which accepts requests according to a whitelist of recognized and legitimate clients. In this

approach if a client is not in the predefined list of accepted users, its request is not allowed. When this happens, the client is asked for an authorization as a legitimacy test, and if the client is authorized it is whitelisted, and after this the traffic originated from this client is considered legitimate.
In [42], the authors proposed a technique to differentiate normal and abnormal traffic in which DDoS attack signatures are discovered by analyzing the TCP/IP packet headers against predefined rules and conditions. By doing so, unusual high traffic caused by an attack is distinguished from the high traffic caused by a flash crowd from legitimate users. ICMP, TCP and UDP flooding attacks were observed as the main focus of DDoS attack types.
In [43], the authors proposed a framework which exploits spatial and temporal correlation of DDoS attack traffic to recognize attack packets and detect DDoS attacks. For this purpose, a perimeter-based DDoS defense system is defined which analyzes the traffic at the edge routers of an ISP network. This framework presents two techniques: (i) temporal-correlation based feature extraction, and (ii) spatial-correlation based detection. By using these two techniques, DDoS detection is made possible without the need of editing existing IP forwarding mechanisms in routers.

2.3.3. Soft Computing Based Methods

Unlike traditional (hard) computing techniques which deals with precision and exactness, soft computing scheme deals with optimization and approximate models in order to solve problems. Soft computing techniques can be used to design intelligent systems tolerant to imprecision, uncertainty, partial truth, and approximation [44]. Soft computing methods are capable of classifying intelligently and automatically with optimization and processing techniques, thus learning paradigms such as neural networks, radial basis functions, genetic algorithms and fuzzy logic are used in DDoS detection [45].

In [46], authors presented SPUNNID, a DDoS attack detection system which uses statistical preprocessor to extract features from the network traffic, and unsupervised artificial neural network (ANN) to analyze and categorize traffic patterns as either attack or normal traffic.

In [47], the authors presented an anomaly-based DDoS detection technique in which Radial Basis Function (RBF) neural networks are used for analyzing features of attack patterns. The proposed method runs on victim side edge routers and applied to classify network data in to normal and attack categories in real time. If the network data is categorized as attack traffic, the source IP addresses of the attack packets are extracted and sent to the Filtering Module and Attack Alarm Module where necessary actions are taken. On the other hand, if the network data is categorized as normal traffic it is sent to the destination IP addresses.

In [48], the authors proposed a method for DDoS attack detection based on fuzzy estimators with mean packet inter-arrival times. The method first detects the DDoS attack and then identify IP addresses of the attack agents. The presented method is competent to identify malicious IPs before the victim system actually suffer from resource exhaustion with 80% success rate and 20% false negative rate.

2.3.4. Data Mining and Machine learning Methods

Signature-based DDoS detection mechanisms cannot detect new attack types. Although anomaly-based mechanisms can detect new attacks, they are generally limited to application design and environment capabilities. Data mining and machine learning techniques use traffic and packet features such as average packet size, inter arrival time, packet size, packet rate, bit rate, etc. to decide if the traffic patterns are attack or normal traffic [49].

In [50], authors presented DDoS Container which is a comprehensive network-based DDoS attack detection framework. The framework works in inline mode in order to inspect and manipulate ongoing traffic in real time. Moreover, by keeping track of all traffic, DDoS container executes stateful inspection on data streams and correlates events among sessions. Traffic pattern analysis and data correlation mechanisms of the framework increases its detection rate for evasive patterns such as encrypted packets. When a DDoS attack is detected, the framework takes appropriate actions such as alerting, blocking, and proactive session termination.

In [51], the authors proposed a model to automatically detect DDoS traffic, based on the discovery that normal traffic pattern does not change much with time, however abrupt changes occur when an attack exists. This network anomaly detection approach is based on discrete wavelet transformation (DWT) and probability theory in order to reduce the false error rate for attack identification.

In [52], authors introduced a DDoS attack detection model based on data mining algorithms, i.e., FCM cluster algorithm and Apriori association algorithm. These algorithms are used to extract a network traffic model and a network packet protocol status model with a threshold is set for detection model. The model captures network traffic value based on k-means clustering algorithm in the data mining module to build threshold values of the network traffic. When a network traffic segment is over a threshold value then the network packet protocol status is checked to detect abnormal packets. Depending on the abnormal network packet protocol status, an attack alarm is triggered.

In [53], the authors proposed a new DDoS detection model that uses a multiclass Support Vector Machine (SVM) to reduce false positive rate. Characteristics of the DDoS attack network traffic are analyzed via implementation of Traffic Rate Analysis (TRA) method. By using multiple SVM model instead of single SVM model, authors achieved more accurate attack detection with lower false positives.

CHAPTER 3

METHODOLOGY

In this chapter, we present an entropy based DDoS attack detection method and implementation on software defined networks.

Entropy is a statistical measure that shows the randomness in a data set. In our case, the data set is the field values of the packets received in a network. Since an attacker sends similar packets consistently, randomness in packet properties eventually decreases. Inversely, attacker can also perform flooding attacks using spoofed packet parameters such as source IP. In this case, randomness in packet properties increases more than expected point. Therefore, entropy calculation is a helpful measurement method to track these unexpected changes in randomness. Each different type of attacks causes changes in the randomness of different fields of IP packets such as destination port. We consider the most affected IP packet fields for entropy calculation. Randomness, i.e. entropy, can be calculated for more than one fields of IP packets at the same time. By means of this characteristic, we can detect different types of attacks by using entropy calculation. Even if the number of malicious packets is less than the number of legitimate packets, this method works because the randomness will change sharply as inspected attribute data is intensified at certain points.

There are many advantages of entropy based statistical approach to detect DDOS attacks. Since this approach only examines incoming packets, it does not cause increased network traffic. Memory and CPU overhead are also negligible

For this study, we use Shannon Entropy [5]. Shannon entropy adapted for DDoS Detection in the literature is defined as;

$$H = -\sum_{n=0}^{M-1} P(X_n)\, log(P(X_n))$$

Here *P* means probability of occurrence of the *X*, where *X* is the attribute value, and *M* is the number of different attribute values. Here attribute means the network packet fields inspected.

3.1 Method

Our entropy based detection method propose the use of multiple entropy values and a novel alarm determination based on these entropy values.

3.1.1. Entropy Attributes

In order to observe randomness in a data set of given network traffic, network packet fields can be selected to observe changes. Unlike many other studies [13], [14], [18], we do not only focus on one attribute like destination IP only. We also focus on other combinations. For different attack types, we consider different packet fields for entropy calculation. In general, we consider destination IP, source IP, destination port, source port and packet size fields of network packet for entropy calculation.

Table 1 shows the packet fields that we consider valuable to observe changes of randomness to detect an attack.

Table 1 List of the fields in the packet header that we focus in entropy calculation

Packet Type	Field	Name in packet header
IP	destination IP	dst
IP	source IP	src
TCP/UDP	destination port	dst_port
TCP/UDP	source port	src_port
TCP/UDP	packet size	total_length

In this method, the entropy values are calculated not only for single field but also for two element combinations of the fields in the IP packet. Let *p* be the set of fields

$$p = \{dst, src, dst\_port, src\_port, total\_length\}$$

We consider all 2 element subsets of the parameter set p and all single elements of p. In total, we have 15 different attribute sets to calculate the entropy for inspection. Table 2 shows the list of 15 different attribute sets that are used in entropy calculations.

Table 2 List of attribute sets

Attribute set number(i)	Packet fields used in entropy calculation
1	Dst
2	Src
3	dst_port
4	src_port
5	total_length
6	{dst , src}
7	{dst , dst_port}
8	{dst , src_port}
9	{dst , total_length}
10	{src , dst_port}
11	{src , src_port}
12	{src , total_length}
13	{dst_port , src_port}

14	{dst_port , total_length}
15	{src_port , total_length}

The number of packets with corresponding attributes for each attribute set are represented with;

$$AS_i^n$$

where *i* is the attribute set number and *n* is the size of the domain of the attribute set *i.*

For example, suppose there are 3 hosts in a network and only 2 ports are in use at each host. While listening to network traffic, we will have a number of packets for 15 attribute sets. Consider, for example, the attribute set 7. The domain of this attribute set is 3x2 since there are 3 hosts (3 different destination IPs) and 2 ports at each host. Therefore, we will have 6 different number of packets for attribute set 7 which will be denoted as AS_7^1, AS_7^2, AS_7^3... AS_7^6

Table 3 Sample matrix for attribute set 1

Attribute: DestinationIP(dst)	The number of packets that matched with the attribute value AS_i^n
10.0.0.14	330
10.0.0.9	221
10.0.0.53	174
10.0.0.44	119
...	...
10.0.0.99	28

In our implementation, we create one matrix for each of the attributes set listed in Table 2. For attribute lists 1 to 5, the corresponding tables have two columns. First column represents

the attribute value (e.g. 10.0.0.14 as destination IP) and the second column shows the number of packets received with this attribute value AS_i^n where 1< i < 5 For example, table 3 shows a sample matrix for attribute set 1 which consists of destination IP only.

For attribute lists 6 to 15, the corresponding tables have three columns. First and second columns represent the attribute value (e.g. 22 as source port and 10.0.0.1 as destination IP) and the third column shows the number of packets received with these attributes value AS_i^n where $1 \leq i \leq 5$. Table 4 shows a sample matrix for attribute set 8 which consists of destination IP and source port only.

Table 4 Sample matrix for attribute set 8

Attribute Set		The number of packets that matched with the attribute value AS_i^n
SourcePort (src_port)	DestinationIP(dst)	
22	10.0.0.14	110
30	10.0.0.9	101
80	10.0.0.53	88
8088	10.0.0.44	35
…	...	...
53	10.0.0.99	25

Here we describe how we calculate the entropy. Let W denote the window size, which is 1000 in our implementation. We use packet-based window sliding by the interval of 250 packets. Let $P(AS_i^n)$ be the probability of each element in the window and calculated as;

$$P(AS_i^n) = \quad AS_i^n / W$$

Entropy(E) is calculated as;

$$E_i = -\sum_{n=0}^{M-1} P(AS_i^n)\, log(P(AS_i^n))$$

Here *i* is the attribute set number and *n* is the size of the domain of the attribute set *i*. *M* is the number of different attribute values. Here attribute means the network packet fields inspected. We calculate 15 different entropy value for each attribute set.

Table 5 List of notations

$E_i(N)_{current}$	Current entropy value in pre-experiment session for each parameter set (i)
$E_i(N)_{prev}$	Previous entropy value in pre-experiment session for each set (i)
$E_i(N)_{stdev}$	Standard deviation value of entropies in pre-experiment session for each set (i)
$E_i(N)_{mean}$	Mean value of entropies in pre-experiment session for each set (i)
$E_i(N)_{min}$	The minimum entropy value of normal traffic captured in pre-experiment session for each set (i)
$E_i(N)_{max}$	The maximum entropy value of normal traffic captured in pre-experiment session for each set(i)
$D_{ij}[CW]$	Difference list of entropy value for each consecutive windows (j) in pre-experiment session for each set(i)
$D_i[CW]_{max}$	Max value in difference list of entropy value for each consecutive window in pre-experiment session for each set(i)
$D_i[CW]_{stdev}$	Standard deviation value of difference list $D_i[CW]$

$D_i[CW]_{mean}$	Mean value of difference list $D_i[CW]$
$E_i(prev)$	Previous entropy value in each set
$E_i(current)$	Current entropy value in each set
$E_i[W]$	Entropy list for each set (i)
W	Windows size
AS_i^n	The number of packets that match with the attribute value in each set

3.1.2. Attack Determination

In the literature, a common attack determining method is to compare current entropy with a constant threshold value directly such as [13][17].

In our method, we focus on more than one condition to determine the attack. We have three different condition statement and each of them evaluates the entropy in different aspects.

Note that we calculate 15 entropy values. Let $E_i(current)$ denote the entropy value calculated for the current window using the i[th] attribute value in Table 2. To determine if there is an attack, we calculate current entropy value $E_i(current)$ and previous entropy value $E_i(prev)$ at each window W. Then our method gives a decision using the following three conditions:

C1) In this condition, we observe consecutive windows difference $E_i(current)$ - $E_i(prev)$ against the mean and standard deviation value of difference list $D_{ij}[CW]$ of consecutive windows in the normal traffic. If the current consecutive windows difference greater than our expected deviation from mean value, we give an alert with "Alert1" sign. This condition is used for early detection.

$$\text{if } E_i(current) - E_i(prev) > (2*D_i[CW]_{stdev}+D_i[CW]_{mean})$$

C2) In this condition, we observe both consecutive windows differences $E_i(current)$ - $E_i(prev)$ and current entropy value $E_i(current)$against the upper and lower limit of our trained traffic. If the current consecutive windows difference greater than the max

consecutive window difference $D_i[CW]_{max}$ or current entropy value falls outside the maximum $E_i(N)_{max}$ and minimum $E_i(N)_{min}$ entropy value, we give an alert with "Alert2" sign .

if $E_i(current)$ - $E_i(prev) > D_i[CW]_{max}$ OR $E_i(current) > E_i(N)_{max}$ OR $E_i(current) < E_i(N)_{min}$

C3) In this condition, we observe current entropy value $E_i(current)$ against the mean and standard deviation value of entropy list $E_i[W]$. If the current entropy value falls outside the expected deviation from mean value, we give an alert with "Alert3" sign.

if $E_i(current) < E_i(N)_{mean} - E_i(N)_{stdev}$ OR $E_i(current) > E_i(N)_{mean} + E_i(N)_{stdev}$

At each window we calculate all 15 entropy values, i.e. $E_i(current)$ for 0<=i<=15. If at least one of the entropy values satisfy C1 OR C2 OR C3, then we give an alert that there may be a DDoS attack on the system.

3.2. Proposed Implementation

In this section, we present our attack detection implementation. In order to implement such a system, the following components must be determined during the design phase:

- Incoming packet inspect: Where to collect the incoming packets to inspect. They can be collected on controller or switches.
- Window size: When to calculate entropy value of the collected packets. It can be packet volume based, or time based.
- Expected value of entropy for normal traffic, i.e. threshold entropy: How to measure and decide the expected value of entropy. This value will be used as a threshold in attack determination.

In the following sections, we explain our method in terms of these components respectively. For each component, we first discuss the approaches used in the literature and then present our design approach.

3.2.1. Incoming packet inspection

In SDN, whenever a new incoming packet arrives to switch, forwarding table is checked for a match. If an incoming packet has a match in the forwarding table, the packet is sent directly to the corresponding switch. Otherwise, it is sent to controller to determine the destination. The most powerful attack can be placed here is to deplete controller resources and make the entire network out of service. A typical DDoS attack works as follows. All

incoming packets are spoofed and have different source addresses so that these packets may never have a match in the forwarding table, and they will be directly forwarded to the controller. If there are more packets than a controller can handle, the controller will be unable to respond to these requests.

There are two different methods for incoming packet inspection: one method is for inspecting all incoming packets, and the other is for inspecting only packets that are not matched in flow tables of switches. In order to inspect all incoming packets, collection process should be placed on switches. On the other hand, if data collection is on controller then only unknown unique packets will be inspected. Mousavi et al. [13] gathers packet information from controller whereas Wang et al [14] and Giotis et al [15] and Kalkan et al [17] gathers from switches in their studies.

Our aim is to detect such a DDoS attack before they disrupt the availability of any device on the SDN network. In this regard, our method inspects every network packet passing through open flow switches and looks at the distribution of specific attributes of the packets to calculate entropy value. When randomness changed unexpectedly in a given data set, the entropy value falls outside the expected entropy range and alert is given. For instance, when an attacker sends hundreds of spoofed packets per second to a server within the network, the number of packets normally begins to increase targeted to a single machine while the number of packets is scattered across all the machines on the network. In this case, as the randomness decreases, measured data of attribute is intensified at a single point and the entropy value decreases.

3.2.2. Window size

Entropy value is calculated for more than one packet; this calculation is repeated at certain periods. Each period called a window and the size of the window can be packet volume based, or time based. In [13], [17] and [18] packet size-based method is used and in [14] and [15] time domain-based method is preferred. The authors in [19] made a comparison between packet volume based and time-based approaches and the results suggest that packet volume-based intervals are more effective than time-based intervals

There are two types of packet-based window: fixed-size window and sliding window. In order to calculate entropy for given attribute in specific interval, packet size-based sliding window approach is more effective than fixed-size window. In sliding window technique, window starts from the first element and keeps shifting right by one element in a frame of fixed size. In fixed window technique, interval is set as fixed frame and after frame is completed new frame starts for completely new packets. This characteristic can cause sharp changes in entropy value and increases the false positive rate. Kalkan et al [17] examine this situation for TCP attack and their results show that usage of sliding window is more effective than fixed-size window to reduce the FPR rate.

Another study [18] made a test for different size of windows to find the best size for packet-based intervals. Their study suggests that for optimal entropy measurement, size of 50 is

the lowest size that effectively detected attacks. Beside this study, the authors in [17] use window size as 1000 packets in their method. Optimal lowest size for window depends on the used network and its characteristics. Accurate size should be found by training the network traffic.

When we trained the traffic dataset we use, we decided to use windows size as 1000 packets sliding by the interval of 250 packets. Our sliding mechanism does not shift one by one, we shift the windows by 250 packets as soon as the new 250 packets arrive. In other words, for each new 250 packets there is a new window and entropy calculation is made for 1000 packets in total together with the previous 750 packets for each new 250 packets

3.2.3. Finding Entropy threshold

In order to detect the changes in an entropy value, a reference value is needed. The current value is compared with this reference value (i.e. threshold) to decide whether there is an attack.

Some studies [14], [17], [18] calculate this value during runtime dynamically, while others such as [13] observe it in pre-experiment before runtime. To decide this value, authors in [14] calculate expected normal entropy with the normal network traffic initially. They assume that when the previous entropy value is not marked as attack traffic then it is regarded as normal entropy value. Then, they calculate standard deviation of this value multiplied by threshold multiplicative factor. Kalkan et al [17] calculate average of current and previous entropy value during runtime and regard as threshold value. Another study [18] calculate standard deviation between average normal traffic and average attack traffic. Other than these, authors in [13] observe this value before runtime by running a 25% rate attack for 25 times to find a suitable threshold. They calculate both the highest value during attack traffic and lowest value during normal traffic then take differences of these values and use it as the threshold value.

In order to find expected value of normal and attack traffic, we made a set of pre-experiment in a similar way to [13].

Dataset

We use real network traffic dataset from Canadian Institute for Cybersecurity research group [54] to train our method for determining expected values of some parameters. CICIDS2017 [55] dataset consists of capturing 1 week of network traffic and it contains benign traffic on Monday. We use this traffic dataset that captured on Monday for both training and experiment session separately. Training and experiments were conducted with different parts of the traffic dataset. We conducted a training session with the first half of this traffic and conducted experiment session with the other half. Our method was run in training mode, the parameters, shown in table 6, and their corresponding values were collected from this part of the dataset. We use these 7 parameters as threshold values for attack determination.

Table 6 List of parameters that captured in pre-experiment session

Notation	Explanation
$E_i(N)_{stdev}$	Standard deviation value of entropies in pre-experiment session for each attribute set (i)
$E_i(N)_{mean}$	Mean value of entropies in pre-experiment session for each attribute set (i)
$E_i(N)_{min}$	The minimum entropy value of normal traffic captured in pre-experiment session for each attribute set (i)
$E_i(N)_{max}$	The maximum entropy value of normal traffic captured in pre-experiment session for each attribute set(i)
$D_i[CW]_{max}$	Max value in difference list of entropy value for each consecutive window in pre-experiment session for each attribute set(i)
$D_i[CW]_{stdev}$	Standard deviation value of difference list $D_i[CW]$
$D_i[CW]_{mean}$	Mean value of difference list $D_i[CW]$

We have 3 different attack determination conditions (see Section 3.1.2) and these parameters are used for threshold value in our conditions. Recall that these parameters are calculated for each 15 different attribute sets. Since each of the attribute set has a different value interval, different thresholds are obtained. The expected value for entropy is an experimental data previously determined for this system. These threshold values should be adjusted according to the institution benign traffic with a training session.

3.2.4. Covered attack types

Entropy based detection methods on software defined networks are designed to detect various attack types. [13], [14] and [16] are designed to detect common flooding attack types such as UDP flood, ICMP floods and TCP SYN flood whereas [15] focuses on DDoS flooding attacks, Worm propagation and Portscan attacks.

According to Kaspersky report [2], the most common type of attack by a wide margin is UDP flooding in 2018. In that regard, our implementation is designed based on detect to UDP and TCP protocol-based attacks such as TCP flood and UDP flood. However, our method can adapt to any flooding attacks. For future work, we aim to make this study large-scale attack detection module covered all types of attacks.

3.3. Comparison with Entropy based implementations in the literature

We investigate the applicability and performance of our proposed mechanism pertaining to other known anomaly detection algorithms reported in the literature. With table 7 we compare our method with others according to specific features.

Table 7 Comparison of literature work

Paper Name	Windows size	Entropy Attributes	Finding expected value for Entropy	Attack determine	Covered Attack Types	Incoming packet inspection
Early detection of DDoS attacks against SDN controllers [13]	Fixed packet size window = 50 packets	dstIP	Pre-Experimental Threshold: Difference of highest value during attack traffic and lowest value during normal traffic	if Entropy < threshold	TCP flood - UDP flood	on controller (only unmatched packets)
An entropy-based distributed DDoS detection mechanism in software-defined networking [14]	time domain windows = between 3s and 7s	dstIP	Runtime threshold: δ= The standard deviation of normal entropy values * Threshold multiplicative factor (this one constraint and not specified)	if E (S_j) − H (S_j) > δ	UDP flood	on OpenFlow switch (all incoming packets)
Combining OpenFlo	Time domain	srcIP,	N. S	N. S	Distributed Denial of Service	on sFlow switch

w and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments [15]	windows = 30s	dstIP, srcPort dstPort			(DDoS), Worm Propagation, Portscan	(all incoming packets)
JESS: Joint Entropy Based DDoS Defense Scheme in SDN [17]	Packet size-based sliding window = 1000	All 2 elements subset of P = {srcIP, dstIP, srcPort dstPort,ProType,pktSize, TTL, TCPflag} (28 pairs)	Runtime threshold: Threshold = (Current Entropy + previous entropy)/ 2	If current entropy <= Threshold	NTP Attack, DNS Amplification SYN flood, generic and mixed attacks.	on OpenFlow switch (all incoming packets)
Early DoS/DDoS Detection Method using Short-term Statistics [18]	Packet size based sliding window = 50 & 500	srcIP	Runtime threshold: Standard deviation between average normal traffic and average attack traffic	N. S	DDoS and the slow DoS (DNS-Amplification)	N. S
Proposed Method	Packet size-based sliding window = 1000 sliding interval = 250	Single and two element combinations of the fields (dstIP, srcIP, dstPort, srcPort, total_length) 15 different attribute sets in total	Pre-Experimental Threshold : Each attribute set has more than one threshold value (7 different	If there is an $E_i(current)$ -that satisfies C1 or C2 or C3 C1) if $E_i(current)$ - $E_i(prev)$ > (2*$D_{ij}[CW]_{stdev}$+ $D_{ij}[CW]_{mean}$) C2) if $E_i(current)$ - $E_i(prev)$ >	UDP flood, TCP flood	on OpenFlow switch (all incoming packets)

			parameters are used)	$D_{ij}[CW]_{max}$ OR $E_i(current) > E_i(N)_{max}$ OR $E_i(current) < E_i(N)_{min}$ C3) if $E_i(current) < E_i(N)_{mean}$- $E_i(N)_{stdev}$OR $E_i(current) > E_i(N)_{mean}+ E_i(N)_{stdev}$		

The studies given in the table were evaluated using different traffic datasets. The success rates given by these studies depend on the used benign and attack traffic and given rates may change by the usage of different traffic sets. Therefore, these rates are not shown in the table as there will be no fair comparison. A performance evaluation using only the same benign and attack traffic would be fair.

CHAPTER 4

EXPERIMENTS

We conducted a series of experiments to investigate the effectiveness of our proposed multiple entropy calculations and our attack determination mechanism. Experiment setup is explained in Section 4.1. Experiment results and comparisons are discussed in Section 4.4

4.1 Setup

In order to evaluate our entropy-based detection method, we conducted a series of experiments. We use Mininet[2] virtual network simulator for these experiments. Mininet is a realistic virtual network with a running kernel, switch and application code on a single machine. This machine can be VM or cloud. Moreover, Mininet supports entire SDN capabilities for this reason it is a great way to experiment with SDN systems. Mininet does not require physical switches, controller, hosts or another network device. To create a complete virtual network including hosts, switches, links and controllers, it is enough to run a single line of code. This simulator comes with default OpenFlow controller which turns switches into learning switches. We use python based open source remote controller to analyze network packets for our detection system. In accordance with this purpose, POX[3] controller is incorporated with Mininet.

Our simulation is run on a Kali GNU/Linux 2018.4 machine with Intel Core i5-4200H 2.80GHz CPU and 8 GB Ram. All simulation software is written using Python language.

Traffic Dataset

We used real network traffic dataset [50] to verify our method in experiment sessions. As explained in Section *3.2.3.* we used only the second half of the "Monday" traffic dataset for

[2] http://mininet.org/

[3] https://noxrepo.github.io/pox-doc/html/

experiment sessions. This part of dataset consists of benign traffic and used as benign activity in our experiments. In the traffic dataset, there are 235 different hosts as destination. All destination addresses directly mapped to hosts addresses running on *Mininet* network with *tcpdump* tool.

Attack Datasets

We used previously captured real attack datasets for our experiments. These datasets consist of only filtered attack packets. We performed the experiments with 4 different dataset we obtained.

Performed attack types:

- UDP Flood DoS Attack
- UDP Flood DDoS Attack
- TCP Flood DoS Attack
- TCP Flood DDoS Attack

We examined our benign traffic to select victim hosts for attack. In benign traffic dataset, the received packet rates of the hosts are different from each other, some hosts are used more intensively. In this context, we selected 3 different victim hosts to measure the performance of our detection method on victim hosts that have different traffic rates. Therefore, we repeated each attack datasets for 3 different hosts. First victim is host14 with the highest traffic, second is host9 with medium traffic and third victim is host104 with least traffic rate on the network. Recall that, as in the benign traffic dataset, all destination addresses directly mapped to hosts addresses running on *Mininet* network with *tcpdump* tool.

In total, we have 12 experiments set up as shown in Table 8. There were three victims host14, host9, and host 104 with high, medium, and low traffic rate in the training data set, respectively. For each victim, we experiment with one of four attack traffic discussed above.

Table 8 List of experiments

Experiment Id	Attack type	Victim host
1	DDoS/UDP Flood	14
2	DDoS/UDP Flood	9
3	DDoS/UDP Flood	104
4	DoS/UDP Flood	14

5	DoS/UDP Flood	9
6	DoS/UDP Flood	104
7	DDoS/TCP Flood	14
8	DDoS/TCP Flood	9
9	DDoS/TCP Flood	104
10	DoS/TCP Flood	14
11	DoS/TCP Flood	9
12	DoS/TCP Flood	104

Attack datasets was replayed 130 seconds after benign traffic started and they continued to work through the whole experiment. The detailed explanation about the running steps of experiments is given in section 4.3.

4.2. Software Defined Networking (SDN)

Software-Defined Networking (SDN) is a new network technology that lets the network to be dynamic, manageable, cost-effective, and adaptable using software applications. This characteristic helps operators manage the entire network consistently and holistically, regardless of the underlying network technology. This approach is based on the physical separation of the network control and forwarding functions. The features of SDN technology are as follows :

- Agile: Separating control mechanisms apart from forwarding allows admins to adjust traffic flow dynamically.
- Centrally managed: Network intelligence is centralized in software-based SDN controllers in order to keep a global view of the network, which appears to applications and policy engines as a single, logical switch.
- Programmatically Configured: SDN allows admins to configure, manage, secure, and optimize network resources very quickly thanks to dynamic, automated SDN programs. This program can run themselves because the programs do not depend on proprietary software.
- Open Standards-Based and Vendor-Neutral: SDN is implemented through open standards, and this simplifies network design and operations. Also, instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

An SDN architecture has three layers: application, control and infrastructure. The application layer contains the network applications or functions. SDN is differed from traditional networks by including application that uses the controller to manage data plane behavior where traditional networks would use a specialized appliance.

The control layer represents the SDN controller software which is centralized and acts as a brain of the software-defined network. This controller resides on a server and manages policies and the flow of traffic throughout the network. The infrastructure layer is made up of the physical switches in the network.

4.3. Simulation Environment

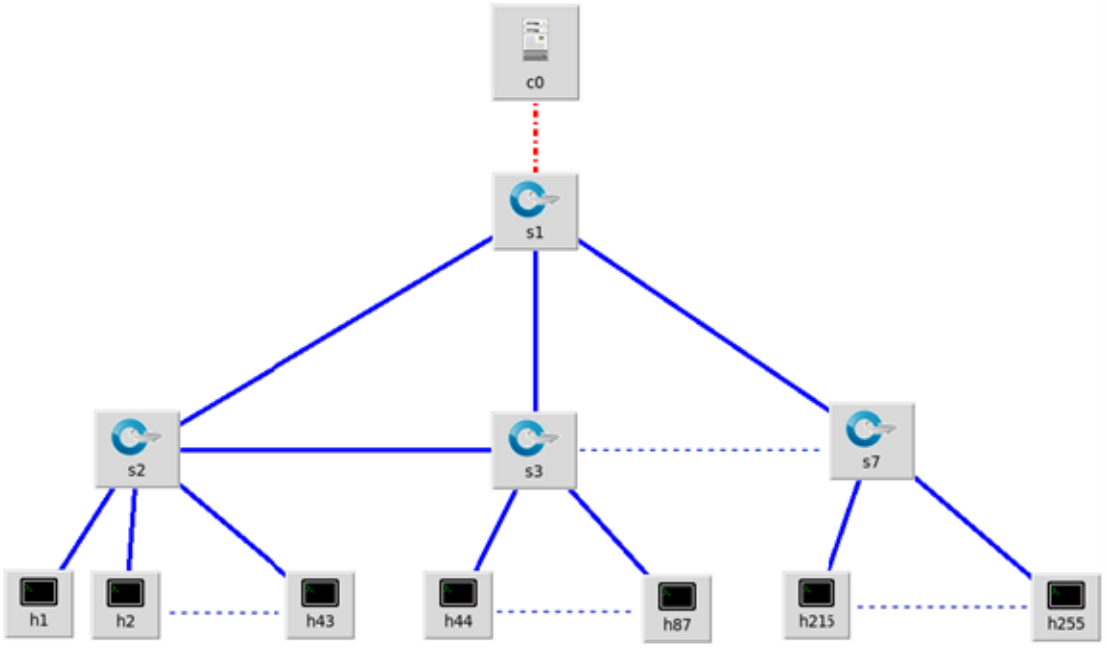

Figure 1 Network Topology

The network topology we created for simulation is shown in Figure 1. Switch 1 (S1) is the switch which is managed by the controller and connected to eight switches S2 to S7. The switches (S2, S3…S6) is connected to 43 hosts and the last switch S7 is connected to 41 hosts. Host h2 was used to replay benign traffic pcap file. Host h3 is used to replay attack traffic pcap file to the victim hosts. Although this is not a large-scale topology, an attacker is able to generate spoofed packet streams with a single machine using packet manipulation tools. These streams seem like they are coming from different IP addresses thus this environment emulates a much larger topology in practice.

In order to perform controlled experiments, a python script was written that automates the experiments. This script first creates the network topology that is shown in Figure 1, then runs the POX controller. Our detection software runs on POX controller, as soon as POX is started our detection software starts to run and continue until controller closes. The script replays benign traffic datasets using "*tcpreplay*" tool as next step. Training and experiments were conducted with different parts of the Monday traffic dataset [54]. The part of the traffic

dataset used for experiments is called benign traffic. Benign traffic dataset was used to simulate real network traffic between hosts on our network topology. Attack dataset is replayed 130 seconds after benign traffic started and they both continue to run for 176 seconds. These steps were repeated for each experiment and running time for each one was 306 seconds. In this way, we can make a healthy comparison for 12 different experiment setups. We used four different attack datasets to verify our method on different types of attacks. For each experiment that run for different attack types, the selected attack traffic was replayed by selecting the corresponding victim hosts.

4.3.1. Experiment Results

In our experiments we evaluate our implementation with respect to the following aspects

1) Evaluate our implementation with respect to different victims of benign traffic: Attacker usually gets information about the devices on the network before initiating the attack. Each device on the network has a different role and accordingly has different traffic rate. Our aim is to measure the performance of our detection method on victim hosts that have high, medium, and low traffic rates.
2) Evaluate our implementation under two different protocol attacks (TCP and UDP): We aim to investigate which attributes in the entropy calculation are more effective under different protocol attacks
3) Evaluate our implementation under single source address and distributed source address of the same attack: Distributed attacks have spoofed source addresses and our goal is to observe the performance differences between these two types. In spoofed DDoS attacks, randomness in packet properties increases more than expected. Therefore, we do not only observe the randomness in packet properties for decreases, we observe unexpected changes in randomness for both decrease and increase.

4) Compare the effectiveness of our entropy parameter selection (5 single attributes and 10 pair of attributes) to entropy calculation with all 3 element and 4 elements subsets

We performed a series of controlled experiments that evaluates the combination of these aspects.

4.3.1.1. Evaluation on DDoS attacks on different protocol types

In this experiment we evaluated our methodology with respect to TCP flood and UDP flood attacks with distributed source addresses on high, medium, and low traffic victims. The evaluation is performed using number of windows the attack detected and the number of attack packets missed. Recall that our methodology gives alert when at least of the 15 entropy attribute set satisfies at least one of the detection conditions (C1 or C2 or C3). To give a detailed evaluation, we present the number of window attack detected and missed for each of the entropy values for each of the detection condition. Table 9 presents the results on DDoS TCP flood and DDoS UDP flood on the victim with the high traffic rate

in the benign traffic (i.e. the victim is host14). During the attack traffic there were 262 windows in total. The last two rows show the results for our methodology and detection rate. The detection rate of the methodology is calculated when at least one entropy attribute set detected.

Table 9 DDoS TCP flood versus DDoS UDP flood on the victim with the high traffic rate (victim is host 14)

	DDOS TCP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	246	750	0	0	246	750
[DSTPORT]	0	249	240	750	4	261	261	250
[SRCIP]	4	261	262	250	4	261	262	250
[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	259	243	500	0	155	0	1000
[DSTIP_SRCIP]	4	261	261	250	4	261	261	250
[DSTIP_DSTPORT]	0	0	0	N/A	4	261	261	250
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	175	0	1000	0	173	0	1000
[DSTPORT_SRCIP]	4	261	261	250	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	4	14	261	250	4	261	261	250
[DSTPORT_SRCPORT]	4	14	261	250	4	15	261	250
[DSTPORT_PCKTLEN]	0	252	210	500	4	260	261	250
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given entropy sets gives alarm:	**36**	**2788**	**3289**		**48**	**3211**	**3379**	
Total number of windows the attack detected:	**4**	**261**	**262**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**100%**		**1,52%**	**99,61%**	**100%**	

Table 10 presents the results on DDoS TCP flood and DDoS UDP flood on the victim with the medium traffic rate in the benign traffic (i.e. the victim is host9).

Table 10 DDoS TCP flood versus DDoS UDP flood on the victim with the medium traffic rate (victim is host 9)

	DDOS TCP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	0	N/A	0	0	0	N/A
[DSTPORT]	0	249	241	750	4	261	261	250
[SRCIP]	4	261	262	250	4	261	262	250
[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	259	241	500	0	156	0	1000
[DSTIP_SRCIP]	4	261	261	250	4	261	261	250
[DSTIP_DSTPORT]	0	0	0	N/A	4	261	261	250
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	174	0	1000	0	173	0	1000
[DSTPORT_SRCIP]	4	261	261	250	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	4	261	261	250	4	261	261	250
[DSTPORT_SRCPORT]	4	12	261	250	4	15	261	250
[DSTPORT_PCKTLEN]	0	252	207	500	4	260	261	250
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given entropy sets gives alarm:	**36**	**3032**	**3039**		**48**	**3212**	**3133**	
Total number of windows the attack detected:	**4**	**261**	**262**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**100%**		**1,52%**	**99,61%**	**100%**	

Table 11 presents the results on DDoS TCP flood and DDoS UDP flood on the victim with the low traffic rate in the benign traffic (i.e. the victim is host104).

Table 11 DDoS TCP flood versus DDoS UDP flood on the victim with the low traffic rate (victim is host 104)

	DDOS TCP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	0	N/A	0	0	0	N/A
[DSTPORT]	0	250	241	750	4	261	261	250
[SRCIP]	4	261	262	250	4	261	262	250
[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	259	241	500	0	155	0	1000
[DSTIP_SRCIP]	4	261	261	250	4	261	261	250

[DSTIP_DSTPORT]	0	0	0	N/A	4	261	261	250
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	176	0	1000	0	172	0	750
[DSTPORT_SRCIP]	4	261	261	N/A	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	4	261	261	250	4	261	261	250
[DSTPORT_SRCPORT]	4	13	261	250	4	15	261	250
[DSTPORT_PCKTLEN]	0	252	210	500	4	260	261	250
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given entropy sets gives alarm:	**36**	**3036**	**3042**		**48**	**3210**	**3133**	
Total number of windows the attack detected:	**4**	**261**	**262**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**100%**		**1,52%**	**99,61%**	**100%**	

According to results, selecting different victim hosts for the attacks does not affect respectable amount on detection rate of performed experiments. Overall detection rate is the same for all victim hosts in both DDoS UDP Flood and DDoS TCP Flood. The only difference is on the total number of alarms that is given by entropy attribute sets individually. When we analyze DDoS UDP Flood attack, C3 gives more alarms than C2. The difference is approximately 4% and this rate continues to decrease between high traffic rate victim 14 and low traffic rate victim 104. On the other hand, in DDoS TCP Flood the difference is approximately 4% and this rate continues to decrease between victim 14 and victim 104.

4.3.1.2 Evaluate our implementation under single source address (DoS) and distributed source address (DDoS) of the same attack

In this experiment we evaluated our methodology with respect to single source address (DoS) and distributed source address (DDoS) for both TCP flood and UDP flood attacks on high, medium, and low traffic victims. Table 12 presents the results on DoS TCP flood and DDoS TCP flood on the victim with the high traffic rate in the benign traffic (i.e. the victim is host14).

Table 12 DoS TCP flood versus DDoS TCP flood on the victim with the high traffic rate (victim is host 14)

	DOS TCP				DDOS TCP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	245	750	0	0	246	750
[DSTPORT]	0	249	240	750	0	249	240	750
[SRCIP]	0	0	1	12000	4	261	262	250

[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	259	243	500	0	259	243	500
[DSTIP_SRCIP]	0	0	0	N/A	4	261	261	250
[DSTIP_DSTPORT]	0	0	0	N/A	0	0	0	N/A
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	174	0	1000	0	175	0	1000
[DSTPORT_SRCIP]	0	0	0	N/A	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	0	175	0	1000	4	14	261	250
[DSTPORT_SRCPORT]	4	12	261	250	4	14	261	250
[DSTPORT_PCKTLEN]	0	252	210	500	0	252	210	500
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given entropy sets gives alarm:	**20**	**2163**	**2244**		**36**	**2788**	**3289**	
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**100%**	

Table 13 presents the results on DoS TCP flood and DDoS TCP flood on the victim with the medium traffic rate in the benign traffic (i.e. the victim is host9).

Table 13 DoS TCP flood versus DDoS TCP flood on the victim with the medium traffic rate (victim is host 9)

	DOS TCP				DDOS TCP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	0	N/A	0	0	0	N/A
[DSTPORT]	0	249	241	750	0	249	241	750
[SRCIP]	0	0	1	12000	4	261	262	250
[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	259	242	500	0	259	241	500
[DSTIP_SRCIP]	0	0	0	N/A	4	261	261	250
[DSTIP_DSTPORT]	0	0	0	N/A	0	0	0	N/A
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	174	0	1000	0	174	0	1000
[DSTPORT_SRCIP]	0	0	0	N/A	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	0	173	0	1000	4	261	261	250
[DSTPORT_SRCPORT]	4	11	261	250	4	12	261	250
[DSTPORT_PCKTLEN]	0	252	210	500	0	252	207	500
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given	**20**	**2160**	**1999**		**36**	**3032**	**3039**	

entropy sets gives alarm:								
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**100%**	

Table 14 presents the results on DoS TCP flood and DDoS TCP flood on the victim with the low traffic rate in the benign traffic (i.e. the victim is host104).

Table 14 DoS TCP flood versus DDoS TCP flood on the victim with the low traffic rate (victim is host 104)

	DOS TCP				DDOS TCP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	0	N/A	0	0	0	N/A
[DSTPORT]	0	249	241	750	0	250	241	750
[SRCIP]	0	0	1	12000	4	261	262	250
[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	259	242	500	0	259	241	500
[DSTIP_SRCIP]	0	0	0	N/A	4	261	261	250
[DSTIP_DSTPORT]	0	0	0	N/A	0	0	0	N/A
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	174	0	1000	0	176	0	1000
[DSTPORT_SRCIP]	0	0	0	N/A	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	0	173	0	1000	4	261	261	250
[DSTPORT_SRCPORT]	4	11	261	250	4	13	261	250
[DSTPORT_PCKTLEN]	0	252	210	500	0	252	210	500
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given entropy sets gives alarm:	**20**	**2160**	**1999**		**36**	**3036**	**3042**	
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**100%**	

Table 15 presents the results on DoS UDP flood and DDoS UDP flood on the victim with the high traffic rate in the benign traffic (i.e. the victim is host14).

Table 15 DoS UDP flood versus DDoS UDP flood on the victim with the high traffic rate (victim is host 14)

	DOS UDP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	246	750	0	0	246	750
[DSTPORT]	0	0	0	N/A	4	261	261	250
[SRCIP]	0	0	0	N/A	4	261	262	250
[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	156	0	1000	0	155	0	1000
[DSTIP_SRCIP]	0	0	0	N/A	4	261	261	250
[DSTIP_DSTPORT]	0	0	0	N/A	4	261	261	250
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	176	0	1000	0	173	0	1000
[DSTPORT_SRCIP]	0	0	0	N/A	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	0	177	0	1000	4	261	261	250
[DSTPORT_SRCPORT]	4	13	261	250	4	15	261	250
[DSTPORT_PCKTLEN]	0	175	0	1000	4	260	261	250
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given entropy sets gives alarm:	**20**	**1739**	**1551**		**48**	**3211**	**3379**	
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**100%**	

Table 16 presents the results on DoS UDP flood and DDoS UDP flood on the victim with the medium traffic rate in the benign traffic (i.e. the victim is host9).

Table 16 DoS UDP flood versus DDoS UDP flood on the victim with the medium traffic rate (victim is host 9)

	DOS UDP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	0	N/A	0	0	0	N. A
[DSTPORT]	0	0	0	N/A	4	261	261	250
[SRCIP]	0	0	1	12000	4	261	262	250
[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	155	0	1000	0	156	0	1000
[DSTIP_SRCIP]	0	0	0	N/A	4	261	261	250
[DSTIP_DSTPORT]	0	0	0	N/A	4	261	261	250
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	174	0	1000	0	173	0	1000

[DSTPORT_SRCIP]	0	0	0	N/A	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	0	171	0	1000	4	261	261	250
[DSTPORT_SRCPORT]	4	13	261	250	4	15	261	250
[DSTPORT_PCKTLEN]	0	171	0	1000	4	260	261	250
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given entropy sets gives alarm:	**20**	**1726**	**1306**		**48**	**3212**	**3133**	
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**100%**	

Table 17 presents the results on DoS UDP flood and DDoS UDP flood on the victim with the low traffic rate in the benign traffic (i.e. the victim is host104).

Table 17 DoS UDP flood versus DDoS UDP flood on the victim with the low traffic rate (victim is host 104)

	DOS UDP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP]	0	0	0	N/A	0	0	0	N/A
[DSTPORT]	0	0	0	N/A	4	261	261	250
[SRCIP]	1	0	0	1000	4	261	262	250
[SRCPORT]	4	261	261	250	4	261	261	250
[PCKTLEN]	0	156	0	1000	0	155	0	1000
[DSTIP_SRCIP]	0	0	0	N/A	4	261	261	250
[DSTIP_DSTPORT]	0	0	0	N/A	4	261	261	250
[DSTIP_SRCPORT]	4	261	261	250	4	261	261	250
[DSTIP_PCKTLEN]	0	176	0	1000	0	172	0	750
[DSTPORT_SRCIP]	0	0	0	N/A	4	261	261	250
[SRCIP_SRCPORT]	4	261	261	250	4	261	261	250
[SRCIP_PCKTLEN]	0	176	0	1000	4	261	261	250
[DSTPORT_SRCPORT]	4	13	261	250	4	15	261	250
[DSTPORT_PCKTLEN]	0	175	0	1000	4	260	261	250
[SRCPORT_PCKTLEN]	4	259	261	250	4	259	261	250
Total number of windows the given entropy sets gives alarm:	**21**	**1738**	**1305**		**48**	**3210**	**3133**	
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**100%**	

When we compare the results in terms of the performance difference between two different protocols, it is seen that overall detection performance is the same for two different protocol UDP Flood and TCP Flood. However, there is a difference on the total number of alarms that is given by entropy attribute sets individually. When we analyze DoS UDP Flood attack, C3 gives more alarms than C2. The difference is approximately 5% and this rate continues to increase between the high traffic rate victim 14 and the low traffic rate victim 104. On the other hand, in DoS TCP Flood the difference is approximately 4% and this rate continues to decrease between victim 14 and victim 104.

Another aspect of this experiment is to evaluate our implementation under single source address and distributed source address of the same attack. In DoS UDP Flood, detection rate of C1 is 1.52%, C2 and C3 are 99.61% for all victim hosts whereas, in DDoS UDP Flood, detection rate of C1 is 1.52%, C2 is 99.61% and C3 is 100% for all victim hosts. According to result we can say that our method is more successful in detecting distributed attacks than single source attacks.

4.3.1.3. Effectiveness of our entropy parameter selection

In this experiment we evaluated our methodology with respect to entropy parameter selection. We suggest that entropy calculation should be performed for 5 single attributes and two element combination of these attributes. In this section we compare the effectiveness of our entropy parameter selection (5 single attributes and 10 pair of attributes) to entropy calculation with all 3 element and 4 elements subsets. The evaluation is performed with respect to single source address (DoS) and distributed source address (DDoS) for both TCP flood and UDP flood attacks on high, medium, and low traffic victims.

Table 18 presents the results on DDoS TCP flood and DDoS UDP flood on the victim with the high traffic rate in the benign traffic (i.e. the victim is host14).

Table 18 DDoS TCP flood versus DDoS UDP flood on the victim with the high traffic rate (victim is host 14)

	DDOS TCP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP_SRCIP_SRCPORT]	*4*	*253*	*261*	*250*	4	252	261	250
[DSTIP_DSTPORT_SRCIP]	*4*	*260*	*261*	*250*	4	260	261	250
[DSTIP_SRCIP_PCKTLEN]	*4*	*258*	*261*	*250*	4	258	261	250
[DSTIP_DSTPORT_SRCPORT]	*4*	*3*	*261*	*250*	4	3	261	250
[DSTIP_DSTPORT_PCKTLEN]	*0*	*171*	*0*	*1000*	4	58	261	250
[DSTIP_SRCPORT_PCKTLEN]	*4*	*258*	*261*	*250*	4	257	261	250
[DSTPORT_SRCIP_SRCPORT]	*4*	*4*	*261*	*250*	4	3	261	250
[DSTPORT_SRCIP_PCKTLEN]	*4*	*260*	*261*	*250*	4	260	261	250

[SRCIP_SRCPORT_PCKTLEN]	*4*	*15*	*261*	*250*	4	15	261	250
[DSTPORT_SRCPORT_PCKTLEN]	*4*	*2*	*261*	*250*	4	2	261	250
Total number of windows the given entropy sets gives alarm:	***36***	***1484***	***2349***		***40***	***1368***	***2610***	
Total number of windows the attack detected:	***4***	***260***	***261***		**4**	**260**	**261**	
Rate:	**1,52%**	***99,23%***	***99,61%***		**1.52%**	**99,23%**	**99,61%**	
[DSTIP_DSTPORT_SRCIP_SRCPORT]	*4*	*2*	*261*	*250*	*4*	*1*	*261*	*250*
[DSTIP_DSTPORT_SRCIP_PCKTLEN]	*4*	*24*	*261*	*250*	*4*	*23*	*261*	*250*
[DSTPORT_SRCIP_SRCPORT_PCKTLEN]	*4*	*13*	*261*	*250*	*4*	*1*	*261*	*250*
[DSTIP_SRCIP_SRCPORT_PCKTLEN]	*4*	*13*	*261*	*250*	*4*	*13*	*261*	*250*
[DSTIP_DSTPORT_SRCPORT_PCKTLEN]	*4*	*1*	*260*	*250*	*4*	*1*	*260*	*250*
Total number of windows the given entropy sets gives alarm:	***20***	***53***	***1304***		***20***	***39***	***1304***	
Total number of windows the attack detected:	***4***	***24***	***261***		***4***	***23***	***261***	
Rate:	**1,52%**	***9,16%***	***99,61%***		***1.52%***	***8,77%***	***99,61%***	
Proposed Attribute sets Result								
Total number of windows the attack detected:	**4**	**261**	**262**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**100%**		**1,52%**	**99,61%**	**100%**	

Table 19 presents the results on DDoS TCP flood and DDoS UDP flood on the victim with the medium traffic rate in the benign traffic (i.e. the victim is host9).

Table 19 DDoS TCP flood versus DDoS UDP flood on the victim with the medium traffic rate (victim is host 9)

	DDOS TCP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP_SRCIP_SRCPORT]	*4*	*252*	*261*	*250*	*4*	*252*	*261*	*250*
[DSTIP_DSTPORT_SRCIP]	*4*	*259*	*261*	*250*	*4*	*260*	*261*	*250*
[DSTIP_SRCIP_PCKTLEN]	*4*	*257*	*261*	*250*	*4*	*258*	*261*	*250*
[DSTIP_DSTPORT_SRCPORT]	*4*	*3*	*261*	*250*	*4*	*4*	*261*	*250*
[DSTIP_DSTPORT_PCKTLEN]	*0*	*172*	*0*	*1000*	*4*	*57*	*261*	*250*
[DSTIP_SRCPORT_PCKTLEN]	*4*	*257*	*261*	*250*	*4*	*258*	*261*	*250*
[DSTPORT_SRCIP_SRCPORT]	*4*	*3*	*261*	*250*	*4*	*4*	*261*	*250*
[DSTPORT_SRCIP_PCKTLEN]	*4*	*260*	*261*	*250*	*4*	*260*	*261*	*250*
[SRCIP_SRCPORT_PCKTLEN]	*4*	*15*	*261*	*250*	*4*	*15*	*261*	*250*

[DSTPORT_SRCPORT_PCKTLEN]	*4*	*3*	*261*	*250*	*4*	*2*	*261*	*250*
Total number of windows the given entropy sets gives alarm:	***36***	***1481***	***2349***		***40***	***1370***	***2610***	
Total number of windows the attack detected:	***4***	***260***	***261***		***4***	***260***	***261***	
Rate:	**1,52%**	***99,23%***	***99,61%***		***1.52%***	***99,23%***	***99,61%***	
[DSTIP_DSTPORT_SRCIP_SRCPORT]	*4*	*1*	*261*	*250*	*4*	*2*	*261*	*250*
[DSTIP_DSTPORT_SRCIP_PCKTLEN]	*4*	*24*	*261*	*250*	*4*	*24*	*261*	*250*
[DSTPORT_SRCIP_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
[DSTIP_SRCIP_SRCPORT_PCKTLEN]	*4*	*13*	*261*	*250*	*4*	*13*	*261*	*250*
[DSTIP_DSTPORT_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
Total number of windows the given entropy sets gives alarm:	***20***	***40***	***1305***		***20***	***41***	***1305***	
Total number of windows the attack detected:	***4***	***24***	***261***		***4***	***24***	***261***	
Rate:	**1,52%**	***9,16%***	***99,61%***		***1.52%***	***9,16%***	***99,61%***	
Proposed Attribute sets Result								
Total number of windows the attack detected:	**4**	**261**	**262**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**100%**		**1,52%**	**99,61%**	**100%**	

Table 20 presents the results on DDoS TCP flood and DDoS UDP flood on the victim with the low traffic rate in the benign traffic (i.e. the victim is host104).

Table 20 DDoS TCP flood versus DDoS UDP flood on the victim with the low traffic rate (victim is host 104)

	DDOS TCP				DDOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP_SRCIP_SRCPORT]	*4*	*253*	*261*	*250*	*4*	*253*	*261*	*250*
[DSTIP_DSTPORT_SRCIP]	*4*	*259*	*261*	*250*	*4*	*259*	*261*	*250*
[DSTIP_SRCIP_PCKTLEN]	*4*	*258*	*261*	*250*	*4*	*258*	*261*	*250*
[DSTIP_DSTPORT_SRCPORT]	*4*	*3*	*261*	*250*	*4*	*3*	*261*	*250*
[DSTIP_DSTPORT_PCKTLEN]	*0*	*174*	*0*	*1000*	*4*	*58*	*261*	*250*
[DSTIP_SRCPORT_PCKTLEN]	*4*	*257*	*261*	*250*	*4*	*256*	*261*	*250*
[DSTPORT_SRCIP_SRCPORT]	*4*	*3*	*261*	*250*	*4*	*3*	*261*	*250*
[DSTPORT_SRCIP_PCKTLEN]	*4*	*260*	*261*	*250*	*4*	*260*	*261*	*250*
[SRCIP_SRCPORT_PCKTLEN]	*4*	*15*	*261*	*250*	*4*	*16*	*261*	*250*

[DSTPORT_SRCPORT_PCKTLEN]	*4*	*3*	*261*	*250*	*4*	*2*	*261*	*250*
Total number of windows the given entropy sets gives alarm:	***36***	***1485***	***2349***		***40***	***1368***	***2610***	
Total number of windows the attack detected:	***4***	***260***	***261***		***4***	***259***	***261***	
Rate:	**1,52%**	***99,23%***	***99,61%***		***1.52%***	***98,85%***	***99,61%***	
[DSTIP_DSTPORT_SRCIP_SRCPORT]	*4*	*2*	*261*	*250*	*4*	*2*	*261*	*250*
[DSTIP_DSTPORT_SRCIP_PCKTLEN]	*4*	*26*	*261*	*250*	*4*	*27*	*261*	*250*
[DSTPORT_SRCIP_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
[DSTIP_SRCIP_SRCPORT_PCKTLEN]	*4*	*13*	*261*	*250*	*4*	*15*	*261*	*250*
[DSTIP_DSTPORT_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
Total number of windows the given entropy sets gives alarm:	***20***	***43***	***1305***		***20***	***46***	***1305***	
Total number of windows the attack detected:	***4***	***26***	***261***		***4***	***24***	***261***	
Rate:	**1,52%**	***9,92%***	***99,61%***		***1.52%***	***9,16%***	***99,61%***	
Proposed Attribute sets Result								
Total number of windows the attack detected:	**4**	**261**	**262**		**4**	**261**	**262**	
Rate:	**1,52%**	**99,61%**	**100%**		**1,52%**	**99,61%**	**100%**	

Table 21 presents the results on DoS TCP flood and DoS UDP flood on the victim with the high traffic rate in the benign traffic (i.e. the victim is host14).

Table 21 DoS TCP flood versus DoS UDP flood on the victim with the high traffic rate (victim is host 14)

	DOS TCP				DOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP_SRCIP_SRCPORT]	*4*	*244*	*261*	*250*	*4*	*245*	*261*	*250*
[DSTIP_DSTPORT_SRCIP]	*0*	*0*	*0*	*N/A*	*0*	*0*	*0*	*N/A*
[DSTIP_SRCIP_PCKTLEN]	*0*	*172*	*0*	*1000*	*0*	*174*	*0*	*1000*
[DSTIP_DSTPORT_SRCPORT]	*4*	*3*	*261*	*250*	*4*	*2*	*261*	*250*
[DSTIP_DSTPORT_PCKTLEN]	*0*	*170*	*0*	*1000*	*0*	*0*	*0*	*N/A*
[DSTIP_SRCPORT_PCKTLEN]	*4*	*256*	*261*	*250*	*0*	*173*	*0*	*250*
[DSTPORT_SRCIP_SRCPORT]	*4*	*3*	*261*	*250*	*4*	*2*	*261*	*250*
[DSTPORT_SRCIP_PCKTLEN]	*0*	*172*	*0*	*1000*	*0*	*175*	*0*	*1000*
[SRCIP_SRCPORT_PCKTLEN]	*4*	*15*	*261*	*250*	*4*	*14*	*261*	*250*
[DSTPORT_SRCPORT_PCKTLEN]	*4*	*2*	*261*	*250*	*4*	*2*	*261*	*250*

Total number of windows the given entropy sets gives alarm:	***24***	***1037***	***1566***		***20***	***787***	***1305***	
Total number of windows the attack detected:	***4***	***261***	***261***		***4***	***261***	***261***	
Rate:	**1,52%**	***99,61%***	***99,61%***		**1,52%**	***99,61%***	***99,61%***	
[DSTIP_DSTPORT_SRCIP_SRCPORT]	*4*	*2*	*261*	*250*	*4*	*1*	*261*	*250*
[DSTIP_DSTPORT_SRCIP_PCKTLEN]	*0*	*169*	*0*	*1000*	*0*	*173*	*0*	*1000*
[DSTPORT_SRCIP_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
[DSTIP_SRCIP_SRCPORT_PCKTLEN]	*4*	*11*	*261*	*250*	*4*	*12*	*261*	*250*
[DSTIP_DSTPORT_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
Total number of windows the given entropy sets gives alarm:	***16***	***184***	***1044***		***16***	***188***	***1044***	
Total number of windows the attack detected:	***4***	***169***	***261***		***4***	***173***	***261***	
Rate:	**1,52%**	***64,50%***	***99,61%***		**1,52%**	***66,03%***	***99,61%***	
Proposed Attribute sets Result								
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**261**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**99,61%**	

Table 22 presents the results on DoS TCP flood and DoS UDP flood on the victim with the medium traffic rate in the benign traffic (i.e. the victim is host9).

Table 22 DoS TCP flood versus DoS UDP flood on the victim with the medium traffic rate (victim is host 9)

	DOS TCP				DOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP_SRCIP_SRCPORT]	*4*	*246*	*261*	*250*	*4*	*244*	*261*	*250*
[DSTIP_DSTPORT_SRCIP]	*0*	*0*	*0*	*N/A*	*0*	*0*	*0*	*N/A*
[DSTIP_SRCIP_PCKTLEN]	*0*	*73*	*0*	*1000*	*0*	*171*	*0*	*1000*
[DSTIP_DSTPORT_SRCPORT]	*4*	*3*	*261*	*250*	*4*	*3*	*261*	*250*
[DSTIP_DSTPORT_PCKTLEN]	*0*	*173*	*0*	*1000*	*0*	*171*	*0*	*1000*
[DSTIP_SRCPORT_PCKTLEN]	*4*	*257*	*261*	*250*	*4*	*257*	*261*	*250*
[DSTPORT_SRCIP_SRCPORT]	*4*	*3*	*261*	*250*	*4*	*3*	*261*	*250*
[DSTPORT_SRCIP_PCKTLEN]	*0*	*173*	*0*	*1000*	*0*	*171*	*0*	*1000*
[SRCIP_SRCPORT_PCKTLEN]	*4*	*15*	*261*	*250*	*4*	*13*	*261*	*250*
[DSTPORT_SRCPORT_PCKTLEN]	*4*	*2*	*261*	*250*	*4*	*2*	*261*	*250*

Total number of windows the given entropy sets gives alarm:	***24***	***945***	***1566***		***24***	***1035***	***1566***	
Total number of windows the attack detected:	***4***	***260***	***261***		***4***	***259***	***261***	
Rate:	**1,52%**	***99,23%***	***99,61%***		**1,52%**	***98,85%***	***99,61%***	
[DSTIP_DSTPORT_SRCIP_SRCPORT]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
[DSTIP_DSTPORT_SRCIP_PCKTLEN]	*0*	*173*	*0*	*1000*	*0*	*171*	*0*	*1000*
[DSTPORT_SRCIP_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
[DSTIP_SRCIP_SRCPORT_PCKTLEN]	*4*	*11*	*261*	*250*	*4*	*12*	*261*	*250*
[DSTIP_DSTPORT_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	*4*	*1*	*261*	*250*
Total number of windows the given entropy sets gives alarm:	***16***	***187***	***1044***		***16***	***186***	***1044***	
Total number of windows the attack detected:	***4***	***173***	***261***		***4***	***171***	***261***	
Rate:	**1,52%**	***66,03%***	***99,61%***		**1,52%**	***65,26%***	***99,61%***	
Proposed Attribute sets Result								
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**261**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**99,61%**	

Table 23 presents the results on DoS TCP flood and DoS UDP flood on the victim with the low traffic rate in the benign traffic (i.e. the victim is host104).

Table 23 DoS TCP flood versus DoS UDP flood on the victim with the low traffic rate (victim is host 104)

	DOS TCP				DOS UDP			
Attribute sets:	C1	C2	C3		C1	C2	C3	
	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect	Number of windows captured	Number of windows captured	Number of windows captured	Number of passed packets to detect
[DSTIP_SRCIP_SRCPORT]	*4*	*246*	*261*	*250*	4	244	261	250
[DSTIP_DSTPORT_SRCIP]	*0*	*0*	*0*	*N/A*	0	0	0	*N/A*
[DSTIP_SRCIP_PCKTLEN]	*0*	*173*	*0*	*1000*	0	174	0	1000
[DSTIP_DSTPORT_SRCPORT]	*4*	*3*	*261*	*250*	4	3	261	250
[DSTIP_DSTPORT_PCKTLEN]	*0*	*173*	*0*	*1000*	0	172	0	1000
[DSTIP_SRCPORT_PCKTLEN]	*4*	*257*	*261*	*250*	4	257	261	250
[DSTPORT_SRCIP_SRCPORT]	*4*	*3*	*261*	*250*	4	3	261	250
[DSTPORT_SRCIP_PCKTLEN]	*0*	*173*	*0*	*1000*	0	174	0	1000
[SRCIP_SRCPORT_PCKTLEN]	*4*	*15*	*261*	*250*	4	14	261	250
[DSTPORT_SRCPORT_PCKTLEN]	*4*	*2*	*261*	*250*	4	2	261	250
Total number of windows the given entropy sets gives alarm:	***24***	***1045***	***1566***		***24***	***1043***	***1566***	

Total number of windows the attack detected:	***4***	***260***	***261***		**4**	**261**	**261**	
Rate:	**1,52%**	***99,23%***	***99,61%***		**1,52%**	**99,61%**	**99,61%**	
[DSTIP_DSTPORT_SRCIP_SRCPORT]	*4*	*1*	*261*	*250*	4	1	261	250
[DSTIP_DSTPORT_SRCIP_PCKTLEN]	*0*	*173*	*0*	*1000*	0	172	0	1000
[DSTPORT_SRCIP_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	4	1	260	250
[DSTIP_SRCIP_SRCPORT_PCKTLEN]	*4*	*11*	*261*	*250*	4	12	260	1000
[DSTIP_DSTPORT_SRCPORT_PCKTLEN]	*4*	*1*	*261*	*250*	4	1	260	1000
Total number of windows the given entropy sets gives alarm:	***16***	***187***	***1044***		***16***	***187***	***1041***	
Total number of windows the attack detected:	***4***	***173***	***261***		**4**	**172**	**261**	
Rate:	**1,52%**	***66,03%***	***99,61%***		**1,52%**	**65,64%**	**99,61%**	
Proposed Attribute sets Result								
Total number of windows the attack detected:	**4**	**261**	**261**		**4**	**261**	**261**	
Rate:	**1,52%**	**99,61%**	**99,61%**		**1,52%**	**99,61%**	**99,61%**	

According to results, the detection rate decreases as the number of element combinations increases in all experiments.

UDP DOS:

Our Method: Detection rate of C1 is 1.52%, C2 and C3 are 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C2 gives more alarms than C3. The difference is approximately 5% and this rate continues to increase between high traffic rate victim 14 and low traffic rate victim 104.

For 3 element combinations of 5 single attributes: Detection rate of C1 is 1.52% for all victims. Detection rate of C2 is 98.85% for victim 14 and 99.61% for the victim with medium traffic rate (victim 9) and low traffic rate victim 104. Detection rate of C3 is 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 20% and this rate continues to increase between victim 14 and victim 104.

For 4 element combinations of 5 single attributes: Detection rate of C1 is 1.52% for all victims. Detection rate of C2 is 66.03% for victim 14 and 65.26% for victim 9 and 65.64% for victim 104. Detection rate of C3 is 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 65% and this rate continues to decrease between victim 14 and victim 104.

UDP DDOS:

Our Method: Detection rate of C1 is 1.52%, C2 is 99.61% and C3 is 100% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 4% and this rate continues to decrease between high traffic rate victim 14 and low traffic rate victim 104.

For 3 element combinations of 5 single attributes: Detection rate of C1 is 1.52% for all victims. Detection rate of C2 is 99.23% for victim 14, victim 9 and 98.85% victim 104. Detection rate of C3 is 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 47% and this rate continues to increase between high traffic rate victim 14 and low traffic rate victim 104.

For 4 element combinations of 5 single attributes: Detection rate of C1 is 1.52% for all victims. Detection rate of C2 is 8.77 % for victim 14 and 9.16% for victim 9, victim 104. Detection rate of C3 is 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 97% and this rate continues to decrease between victim 14 and victim 104.

TCP DDOS:

Our Method: Detection rate of C1 is 1.52%, C2 is 99.61% and C3 is 100% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 13% and this rate continues to decrease sharply between high traffic rate victim 14 and low traffic rate victim 104.

For 3 element combinations of 5 single attributes: Detection rate of C1 is 1.52% for all victims. Detection rate of C2 is 99.23% for all victim hosts. Detection rate of C3 is 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 33% and this rate continues to increase between victim 14 and victim 104.

For 4 element combinations of 5 single attributes: Detection rate of C1 is 1.52% for all victims. Detection rate of C2 is 9.16 % for victim 14 and victim 9, 9.92% for victim 104. Detection rate of C3 is 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 95% and this rate continues to increase between victim 14 and victim 104.

TCP DOS:

Our method: Detection rate of C1 is 1.52%, C2 and C3 are 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2 with the difference of %2 for high

traffic rate victim 14. However, C2 gives more alarms than C3 for medium traffic rate victim 9 and low traffic rate victim 104 with the rate of 4%.

For 3 element combinations of 5 single attributes: Detection rate of C1 is 1.52% for all victims. Detection rate of C2 is 99.61% for victim 14, 99.23% for victim 9 and victim 104. Detection rate of C3 is 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 20% and this rate continues to increase between victim 14 and victim 104.

For 4 element combinations of 5 single attributes: Detection rate of C1 is 1.52% for all victims. Detection rate of C2 is 64.50% for victim 14, 66.03% for victim 9 and victim 104. Detection rate of C3 is 99.61% for all victim hosts. When we compare the total number of alarms that is given by entropy attribute sets individually, it is seen that C3 gives more alarms than C2. The difference is approximately 65% and this rate continues to decrease between victim 14 and victim 104.

4.3.1.4. Success rates

In this section, we present success rates of given method in terms of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) rates. We also give these rates for both three and four element combinations of 5 single attribute sets and compare them with our method. Table 24 shows the definition of predicted outcomes.

Table 24 Definitions of predicted outcomes

Predicted outcomes	Meaning
True positive (TP)	Attack exists and method gives alarm
True negative (TN)	Attack does not exist, and method does not give alarm
False positive (FP)	Attack does not exist, but method gives alarm
False negative (FN)	Attack exists but method does not give alarm

The experiments given in the previous sections show our detection rates when an attack traffic appears on benign traffic. As a result of these experiments we obtained false negative and detection rates (positive rates) of our method. However, observed detection rates can be consisting of both false positive (FP) and true positive (TP) alarms. We made an additional experiment to determine that whether the generated alarms were really generated by malicious attack traffic or not. Through this experiment we can distinguish true positive (TP) alarms from false positive (FP) alarms and we also show true negative rates of the proposed method.

We also measure these values for both three and four element combinations of 5 single attribute sets. We used the ‘Monday’ traffic of the benign traffic dataset [50]. We run our

benign traffic dataset that used in experiments for the same period (306 seconds) as the experiments. This experiment was performed for 46000 packets and these packets are exactly the same as those used in the experiments. We did not send attack traffic in this experiment, therefore all the alarms produced by our method are false positives.

Table 25 shows the success rates of given method in terms of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) rates for selected attack type "*DDoS UDP flood on the victim 14"*.

Table 25 Confusion matrices

Included parameters in entropy calculation	FP	FN	TN	TP
5 single attributes and two element combinations	2.71%	0%	10.17%	87.11%
Three element combinations of 5 single attributes	2.37%	0.33%	10.50%	86.78%
Four element combinations of 5 single attributes	2.03%	0.33%	10.84%	86.78%

CPU and Memory Overhead

Since our detection technique relies on statistical calculation, it does not impose an overhead on the system unlike other detection techniques such as machine learning approaches. Our detection method has two different processes; incoming packet collection and entropy calculation. These two processes have $O(N)$ complexity where N is the number of different packets in every attribute set. Memory and CPU overhead were measured during the experiments. Memory overhead is approximately 48 MB and CPU usage increased by %4 during attack stage shown in Appendix A.

4.4. Limitations

We performed a series of experiments to evaluate the performance of our method. These experiments were tested in the simulation environment and this causes some limitations. There are very few datasets published for academic usage and since these datasets are recorded for a specific time of a specific network, results obtaining from it cannot be generalized for other network traffic. Different results may be obtained in the usage of different experimental environment, benign traffic and attack traffic. However, these differences do not mean that the method does not work and fails.

4.5. Discussion

The following inference can be drawn from the results of the experiments:

- There is no case where the 3 and 4 element combinations of attributes found the attack, that our proposed method could not. Therefore, we concluded that the proposed single field and two element combination is enough in detection.

- In all the experiments, our method detected the attack at the 250th which is the first window packet which corresponds to approximately 400 milliseconds with the condition C1 after the attack starts.

- There is no single entropy that detects the attacks in all cases. Therefore, we suggest to use all 15 entropy attribute sets together.

- Our method detected all simulated DDOS attacks with 97.28% accuracy by condition 3 for every victim hosts. If we look at the detection rate of all entropies separately, we can say that our method has more success rate in TCP protocol for DOS attacks, and UDP protocol for DDOS attacks (those data are all related with attacks traffic). Between these protocols, there is no parameter that an entropy based detection method can recognize the difference. The changes in the fields that we calculated for entropy can be protocol independent.

- The condition C1 gave alert in the first 4 windows during all experiments. This is because this condition generates an alarm by looking at the difference between consecutive windows. Once the attack has started, the difference between the entropy continues to increase and at the end of window 4, the window to be calculated will be received from completely new packages (because the sliding interval is 250 and the window size is 250x4 = 1000), which results in uncaught attacks by the C1 condition. The entropy after 4 windows results in minimum or maximum values and stabilizes at that values, which does not cause alarms. Therefore, the C1 condition is ideal for the early detection systems. However, it should be used in combination with C2 and C3 to detect the continuation of the attacks.

- Because of this characteristic that condition C1 has, cascade warning system can be deployed in implementation stage. Condition C2 and condition C3 can be activated after condition C1 is triggered. This change will prevent false positive alarms that can be generated by C2 and C3 when there is no attack.

CHAPTER 5

CONCLUSION

In this thesis, we describe entropy based detection method and implement our method on software defined networks (SDN). We implemented a python software that collects all incoming packets from Openswitch using POX controller. Different packet field parameters are examined in entropy calculation to evaluate the performance of our method. According to evaluation results, given method can detect the attacks at a very early stage. Our method can detect various attacks in average of 250 packets with 96.95% accuracy and 2.71% false positive rate. This result shows the success rate of the proposed method.

We conducted a series of experiments with real datasets for four different attack types to evaluate our method. We compare the effectiveness of our entropy parameter selection (5 single attributes and 10 pair of attributes) to entropy calculation with all 3 elements and 4 elements subsets According to our results, it is insufficient to calculate entropy for only one field, we suggest that entropy calculation should be performed 5 single attributes and 10 pair of attributes that are shown in table 2.

The following ideas can be considered as future work. Other detection methods proposed in the literature can be evaluated using same traffic and attack datasets to compare their performance accurately. IPv6 packets should be included in entropy calculation. Moreover, different attack types should be tested for performance evaluation. Expanded experiment can be conducted to evaluate our method more precisely.

Our implementation is published at *https://github.com/users/frknycbs/projects/1*

REFERENCES

[1] "DDoS attacks dropped 13 percent last year but the average duration increased," [Online]. Available: https://www.techspot.com/news/78661-ddos-attacks-dropped-13-percent-last-year-but.html. [Accessed June 2019].

[2] "DDoS attacks in Q4 2018," [Online]. Available: https://securelist.com/ddos-attacks-in-q4-2018/89565/. [Accessed June 2019].

[3] "SDN," [Online]. Available: https://cseweb.ucsd.edu/classes/fa16/cse291-g/applications/ln/SDN.pdf. [Accessed June 2019].

[4] "Open SDN Controller," [Online]. Available: https://www.cisco.com/c/en/us/products/cloud-systems-management/open-sdn-controller/index.html. [Accessed July 2019].

[5] "Entropy," [Online]. Available: http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf. [Accessed July 2019].

[6] A. Wagner and B.Plattner, "Entropy based worm and anomaly detection in fast IP networks," in *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*, 2005.

[7] "SDN Architecture," [Online]. Available: https://www.researchgate.net/figure/A-three-layer-software-defined-networking-SDN-architecture_fig1_284696928. [Accessed September 2019].

[8] "Flash Crowd," [Online]. Available: https://www.yourdictionary.com/flash-crowd. [Accessed September 2019].

[9] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "Aroma: An SDN based autonomic DDoS mitigation framework," *Computers & Security, vol. 70*, p. 482–499, 2017.

[10] K. Kumar, R. C. Joshi, K. Singh, "A Distributed Approach using Entropy to Detect DDoS Attacks in ISP Domain," in *2007 International Conference on Signal Processing, Communications and Networking*, Chennai, 2007.

[11] Y. Chen, X. Ma and X. Wu, "DDoS Detection Algorithm Based on Preprocessing Network Traffic Predicted Method and Chaos Theory," *IEEE Communications Letters*, vol. vol. 17, no. 5, pp. 1052-1054, 2013.

[12] S. Yu, W. Zhou, R. Doss and W. Jia, "Traceback of DDoS Attacks Using Entropy Variations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 3, pp. 412-425, March 2011.

[13] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, Garden Grove, CA, February 2015.

[14] R. Wang, Z. Jia and L. Ju, "An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking," *2015 IEEE Trustcom/BigDataSE/ISPA*, pp. 310-317, Helsinki, 2015.

[15] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 62, pp. 122-136, 2014.

[16] M. Wang, B. Li and Z. Li, "sFlow: towards resource-efficient and agile service federation in service overlay networks," *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, pp. 628-635, Tokyo, Japan, 2004.

[17] K. Kalkan, L. Altay, G. Gür and F. Alagöz, "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358-2372, Oct. 2018.

[18] S. Oshima, T. Nakashima, and T. Sueyoshi, "Early DoS/DDoS detection method using short-term statistics," in *CISIS 2010 - The 4th International Conference on Complex, Intelligent and Software Intensive Systems*, 2010, pp. 168–173.

[19] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: Statistics-based overload control against distributed denial-of-service attacks," in *Proceedings - IEEE INFOCOM*, 2004, vol. 4, pp. 2594–2604.

[20] I. Özçelik and R. R. Brooks, "Deceiving entropy based DoS detection," *Comput. Secur.*, vol. 48, pp. 234–245, Feb. 2015.

[21] D. Erhan and E. Anarim, "Statistical Properties of DDoS Attacks," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019, pp. 1238–1242.

[22] M. D. Prasad, P. B. V, and C. Amarnath, "Machine Learning DDoS Detection Using Stochastic Gradient Boosting," *Int. J. Comput. Sci. Eng.*, vol. 7, no. 4, pp. 157–166.

[23] C. Douligeris and A. Mitrokotsa, “DDoS attacks and defense mechanisms: classification and state-of-the-art,” *Comput. Networks*, vol. 44, no. 5, pp. 643–666, Apr. 2004.

[24] “DDoS Incident Report.” [Online]. Available: https://github.blog/2018-03-01-ddos-incident-report/. [Accessed: 30-Nov-2019].

[25] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, “Understanding the Mirai Botnet,” *26th USENIX Secur. Symp. (USENIX Secur. 17)*, pp. 1093--1110, 2017.

[26] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, p. 39, Apr. 2004.

[27] Specht, S.M., & Lee, R.B. (2003). Taxonomies of Distributed Denial of Service Networks, Attacks, Tools, and Countermeasures.

[28] S. & L. R. Specht, “Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures.,” pp. 543–550, 2004.

[29] “2001 Tech Tip: Trends in Denial of Service Attack Technology.” [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=52490. [Accessed: 18-Dec-2019].

[30] Daemon9, Route, Infinity, IP-spoofing demystified: trustrelationship exploitation, Phrack Magazine, GuildProductions, kid, June 1996.

[31] “Deep Inside a DNS Amplification DDoS Attack.” [Online]. Available: https://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack/. [Accessed: 18-Dec-2019].

[32] “DDoS Threat Report 2019 Q2”, https://www.nexusguard.com/threat-report-q2-2019, Accessed in October 2019.

[33] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, “Taming the 800 Pound Gorilla,” in *Proceedings of the 2014 Conference on Internet Measurement Conference - IMC ’14*, 2014, pp. 435–448.

[34] Cisco Systems, Inc., “Defining Strategies to Protect Against TCP SYN Denial of Service Attacks,” *2006*.

[35] Y. Lee and Y. Lee, “Detecting DDoS attacks with Hadoop,” in *Proceedings of The ACM CoNEXT Student Workshop on - CoNEXT ’11 Student*, 2011, pp. 1–2.

[36] Chen, Yu & Hwang, Kai. (2010). Distributed Change-Point Detection of DDoS Attacks over Multiple Network Domains *.

[37] T. Peng, C.Leckie, and RM Rao, K. (2004) ‘Detecting distributed denial of service attacks using source IP address’ monitoring. Proceedings of the 3rd International IFIP-TC6 Networking Conference, Athens, Greece, 9-14 May, pp. 771–782. Springerverlag.

[38] Cheng J., Yin J., Wu C., Zhang B., Liu Y. (2009) DDoS Attack Detection Method Based on Linear Prediction Model. In: Huang DS., Jo KH., Lee HH., Kang HJ., Bevilacqua V. (eds) Emerging Intelligent Computing Technology and Applications. ICIC 2009. Lecture No.

[39] Y. Gilad and A. Herzberg, "LOT," *ACM Trans. Inf. Syst. Secur.*, vol. 15, no. 2, pp. 1–30, Jul. 2012.

[40] T. Spyridopoulos, G. Karanikas, T. Tryfonas, and G. Oikonomou, "A game theoretic defence framework against DoS/DDoS cyber attacks," *Comput. Secur.*, vol. 38, pp. 39–50, Oct. 2013.

[41] R. Thomas, Hong Zhu, T. Huck, and T. Johnson, "NetBouncer: client-legitimacy-based high-performance DDoS filtering," in *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 2, p. 111.

[42] L. Limwiwatkul and A. Rungsawang, "Distributed denial of service detection using TCP/IP header and traffic measurement analysis," in *IEEE International Symposium on Communications and Information Technology, 2004. ISCIT 2004.*, vol. 1, pp. 605–610.

[43] K. Lu, D. Wu, J. Fan, S. Todorovic, and A. Nucci, "Robust and efficient detection of DDoS attacks for large-scale internet," *Comput. Networks*, vol. 51, no. 18, pp. 5036–5056, Dec. 2007.

[44] D. Ibrahim, "An Overview of Soft Computing," *Procedia Comput. Sci.*, vol. 102, pp. 34–38, 2016.

[45] M. H. Bhuyan, H. J. Kashyap, D. K. Bhattacharyya, and J. K. Kalita, "Detecting Distributed Denial of Service Attacks: Methods, Tools and Future Directions," *Comput. J.*, vol. 57, no. 4, pp. 537–556, Apr. 2014.

[46] R. Jalili, F. Imani-Mehr, M. Amini, and H. R. Shahriari, "Detection of Distributed Denial of Service Attacks Using Statistical Pre-processor and Unsupervised Neural Networks," 2005, pp. 192–203.

[47] A. Karimazad, R., & Faraahi, "An Anomaly-Based Method for DDoS Attacks Detection using RBF Neural Networks.," 2011.

[48] S. N. Shiaeles, V. Katos, A. S. Karakos, and B. K. Papadopoulos, "Real time DDoS detection using fuzzy estimators," *Comput. Secur.*, vol. 31, no. 6, pp. 782–790, Sep. 2012.

[49] M. Alkasassbeh, G. Al-Naymat, A. B.A, and M. Almseidin, "Detecting Distributed Denial of Service Attacks Using Data Mining Techniques," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 1, 2016.

[50] Z. Chen, Z. Chen, and A. Delis, "An Inline Detection and Prevention Framework for Distributed Denial of Service Attacks," *Comput. J.*, vol. 50, no. 1, pp. 7–40, Oct. 2006.

[51] M. Li and M. Li, "A New Approach for Detecting DDoS Attacks Based on Wavelet Analysis," in *2009 2nd International Congress on Image and Signal Processing,*

2009, pp. 1–5.

[52] Zhong, Rui & Yue, Guangxue. (2010). DDoS Detection System Based on Data Mining.

[53] J. Seo, C. Lee, T. Shon, K.-H. Cho, and J. Moon, “A New DDoS Detection Model Using Multiple SVMs and TRA,” 2005, pp. 976–985.

[54] “UNB IDS 2017 Dataset,” 2017. [Online]. Available: http://205.174.165.80/CICDataset/CIC-IDS-2017/. [Accessed: 30-Nov-2019].

[55] “UNB IDS 2017,” 2017. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html/. [Accessed: 30-Nov-2019].

APPENDICES

APPENDIX A

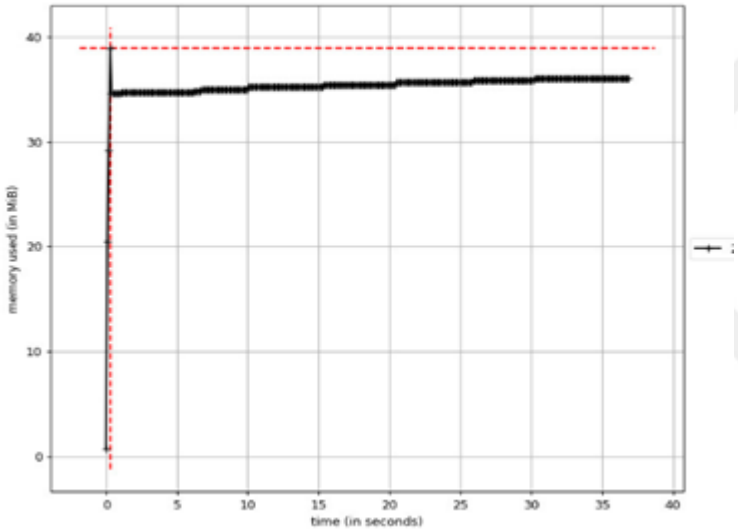

Figure 2 Memory overhead

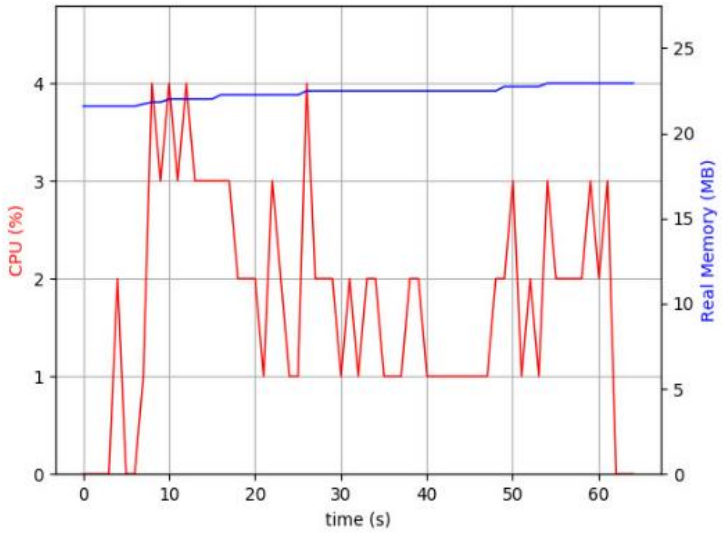

Figure 3 CPU overhead

TEZ İZİN FORMU / THESIS PERMISSION FORM

ENSTİTÜ / INSTITUTE

Fen Bilimleri Enstitüsü / Graduate School of Natural and Applied Sciences ☐

Sosyal Bilimler Enstitüsü / Graduate School of Social Sciences ☐

Uygulamalı Matematik Enstitüsü / Graduate School of Applied Mathematics ☐

Enformatik Enstitüsü / Graduate School of Informatics ☐

Deniz Bilimleri Enstitüsü / Graduate School of Marine Sciences ☐

YAZARIN / AUTHOR

Soyadı / Surname : .....
Adı / Name : .....
Bölümü / Department : .....

TEZİN ADI / TITLE OF THE THESIS (**İngilizce** / English) : .....
.....
.....
.....
.....

TEZİN TÜRÜ / DEGREE: Yüksek Lisans / Master ☐ **Doktora** / PhD ☐

1. **Tezin tamamı dünya çapında erişime açılacaktır. /** Release the entire work immediately for access worldwide. ☐
2. **Tez iki yıl süreyle erişime kapalı olacaktır.** / Secure the entire work for patent and/or proprietary purposes for a period of **two year. *** ☐
3. **Tez altı ay süreyle erişime kapalı olacaktır.** / Secure the entire work for period of **six months**. ***** ☐

***** *Enstitü Yönetim Kurulu Kararının basılı kopyası tezle birlikte kütüphaneye teslim edilecektir.*
A copy of the Decision of the Institute Administrative Committee will be delivered to the library together with the printed thesis.

Yazarın imzası / Signature ..... **Tarih** / Date .....