12-2019

Comparative Analysis Based on Survey of DDOS Attacks' Detection Techniques at Transport, Network, and Application Layers

Mustafa Khambatta
mukham27@gmail.com

Comparative Analysis Based on Survey of DDOS Attacks' Detection Techniques at Transport, Network, and Application Layers

by

Mustafa Khambatta

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Information Assurance

December, 2019

Starred Paper Committee:
Mark Schmidt, Chairperson
Dennis Guster
Lynn Collen

Abstract

Distributed Denial of Service (DDOS) is one of the most prevalent attacks and can be executed in diverse ways using various tools and codes. This makes it very difficult for the security researchers and engineers to come up with a rigorous and efficient security methodology. Even with thorough research, analysis, real time implementation, and application of the best mechanisms in test environments, there are various ways to exploit the smallest vulnerability within the system that gets overlooked while designing the defense mechanism. This paper presents a comprehensive survey of various methodologies implemented by researchers and engineers to detect DDOS attacks at network, transport, and application layers using comparative analysis. DDOS attacks are most prevalent on network, transport, and application layers justifying the need to focus on these three layers in the OSI model.

Keywords: DDOS, DOS, Machine Learning, Security, Detection

Acknowledgments

This research was initially supported by first advisor Dr. Tirthankar Ghosh. It was a pleasure working with him, who helped me understand, encouraged me to explore this topic further. He is no longer working at St. Cloud State University as he found a new opportunity in a different university. I would like to further thank Dr. Mark Schmidt who was my second advisor that has helped me be part of the Las Vegas security conference and SCSU Research Colloquium in 2017, that immensely helped me in getting more interested in this research and security in general.

I would also like to thank Dr. Dennis Guster and Dr. Lynn Collen of their continuous support in my endeavor of writing this starred research paper. It was with their help in critically analyzing every process of this paper that helped me improve overall in both my writing and delivering of information.

Lastly, I would like to thank my parents, siblings, and family, who have been a great support in my academic and professional journey.

Table of Contents

Chapter Page

List of Tables

List of Figures

Chapter I: Introduction

Introduction

Distributed Denial of Service (DDOS) attacks are one of the most prevalent threats among other cyber-attacks. Among many DDOS attacks on the seven OSI layers, the ones executed on network, transport, and application layer are of the highest volume (network and transport—46% and application—54%) (Kostadinov, 2013).

This paper's focus is to study the vulnerabilities within a network and application layer that leads to DDOS attacks and to compare various detection techniques used to secure these two OSI layers. It will delve into specific network and application layer attacks and their detection techniques in a way to cover ample information for comparative analysis which will be conducted using specific performance metrics.

The paper will be divided into three sections. The first section will be transport layer attacks and detection techniques, the second section will be network layer attacks and detection techniques, and the third section will be application layer attacks and detection techniques.

In the first section, there will be literature reviews of specific attack's detection and mitigation techniques. These techniques will be analyzed either quantitatively or qualitatively based on the performance metrics used within a particular research article/paper.

The performance metrics like false positive/false negative rates (high or low), memory consumption, high rate traffic accuracy detection, and low rate traffic accuracy detection are used to conduct the comparative analysis of various detection techniques. The paper will also have qualitative analysis based on the advantages or disadvantages of the mitigation techniques.

The results of the paper would focus on the best five mitigation techniques that proved to be efficient in delivering the results based on the comparative analysis.

The most common DDOS Application layer attacks consist of HTTP GET Floods, HTTP POST floods, PUSH floods, and HEAD floods. The application layer DDOS attacks are way more difficult to mitigate and are not easily prevented using firewall layer security or just blocking IP's. There is ongoing research being conducted in this area and not much information is out there in terms of discovering an extremely effective solution to this layer of DDoS attacks.

Problem Statement

DDOS attacks on network, transport, and application layers are one of the most predominant cyber-attacks and are of the highest volume compared to other OSI layers. Moreover, application layer attacks are difficult to prevent because they are low rate attacks (sneaky) making it appear as normal traffic. This paper presents a comprehensive survey of various methodologies implemented by researchers and engineers to detect DDOS attacks at network, transport, and application layers with their comparative analysis.

Nature and Significance of the Problem

Network, transport, and especially application layer DDOS attacks are becoming common since the last few years. Application layer DDOS attacks are more cost effective, easy to evade defenses, difficult to detect, and causes extensive damage. This research paper's aim is to compare the detection techniques/methodologies provided by researchers from various mathematical (machine learning engineers) and security backgrounds. It would showcase the wide usage of machine learning in regard to detecting these attack scenarios. It is also necessary to keep in mind that machine learning is still in its development stages and monitoring its results

and inputs on a frequent basis is of utmost importance. This paper would not only help with DDoS attacks, but the machine learning methodologies' fundamentals explained here would be helpful in other security related threat/vulnerability scenarios.

Objective of the Research

The objective is to present a comprehensive survey of various detection techniques for DDOS attacks at network, transport, and application layers and present a comparative analysis to help prepare different mitigation approaches. This is a theoretical approach using quantitative data from various tested methodologies and its varying results in a test environment. This paper's focus is to provide future researchers with the fundamental knowledge of machine learning algorithms for DDoS detection and the capabilities it holds regarding molding these algorithms to work for mitigating these attacks. This will not only impact the way we approach security, but the policies involved while dealing with security threats and vulnerabilities. It is about entering a technological era where we need to be more vigilant of the tools (like machine learning) available to us that could be useful (if used carefully). It is important to understand these methodologies because not only it has implementations in detection and mitigation but also in creating new attacks. If we do not understand these techniques then bad actors using these methodologies to create attacks could have a disastrous impact on a company/organization/nation during a security threat/emergency due to our lack of knowledge/understanding of the attack tools (like machine learning) that could have been potentially used. As the famous saying goes in cyber-security "a defender has to know everything in order to protect valuable information from attackers; whereas an attacker just needs to know/exploit a single vulnerability within a system to render all defenses useless" making it

highly important to know every aspect of security threats. Machine learning could prove to be one which if not studied and developed well by the good guys for defense, then an attack (using ML alg.) by bad actors can be quite difficult to mitigate.

Research Questions and/or Hypothesis

- Research Question 1. What is the false positive and false negative rate? (Quantitative)—the before and after study. This is the most challenging part of the paper as it would be difficult to compare the results or performance of a detection technique since each would have a different rate of attack flow. For this reason, I would simplify it and trim it down to the detection rate accuracy, false positive, and false negative rate, forming better understanding.
- Research Question 2. How effective is it in terms of detecting these attacks? (Quantitative Analysis-Survey Comparison)

The above questions can later help with the longitudinal study of being able to do continuing research to find resolutions and solutions for the vulnerabilities which were not addressed or left out due to the limitations of these mechanisms. Therefore, this research paper could be helpful for other researchers who want to do thesis or continue the study on DDOS attacks prevention mechanism (more effective than the ones discussed in this paper). This is more of a retrospective and non-experimental study for the fact that the results obtained are from the past tests done within a test or real environments; which are being used for the comparative analysis.

Shortcomings

This research would not be able to test out the machine learning algorithms' tools as it is practically not feasible to create a scenario to generate a high volumetric DDoS attack or varied behavioral application layer attacks due to the time constraint (it would take months just to test out one algorithm and making it learn different attack patterns to test out its true efficiency or effectiveness and let alone create a volumetric attack (it would require a high network bandwidth and big data centers to create a real scenario)). To just test out on a small network or a laptop/PC would be meaningless; since the real attack scenario would be way more complicated and/or would consist of high-volume network attack traffic. Therefore, it would not develop the correct test results necessary to be able to determine its true efficiency and/or effectiveness in a real-world situation.

Definition of Terms

- ARP: Address resolution protocol matches the physical address to right IP address when a host needs to send packets across a physical network. It could be between Ethernet and IP addresses or ATM (asynchronous transfer mode) and Ethernet.
- BGP: Border Gateway Protocol is the protocol responsible for transmitting data to the client using the best route (hops) available. It performs this task based on the available paths and network policies set by the network admin.
- Botnet: It is a group of connected computers used to perform repetitive tasks. It is also used by attackers through infecting these computers also called as zombies to perform DDOS attacks.

- Cache: The most common caches are memory cache, processor cache, browser cache, and disk cache. It is a temporary storage of information by a computer, browser application, or processor that can later be accessed by a that program at a much faster rate.
- CDN: Content Distribution Network is a group of servers that cache content and are distributed across many geographical locations; help in delivering the content faster. The web server accesses the server (CDN network) closest to your location.
- Datagram: It is like a packet, but it does not need a confirmation (connectionless communication) that it was received. It is popular amongst streaming services due to its efficiency of continuous delivery over reliability.
- DDOS: In this attack, multiple machines operate together as a botnet to flood the target system. It is the same concept as DOS except it done on a large scale.
- Deep Learning: It is a subset of ML that uses perceptron like our human brain to compute various features based on the assigned weight or the weight assigned by the algorithm if it is unsupervised. Artificial Neural Networks (ANN) can be a good example of deep learning algorithms.
- DNS: Domain Name System helps translates domain names (e.g., google.com) into IP addresses; thereby allowing you to access a website using their domain name (google.com).
- DOS: Denial of Service is a cyberattack where the attacker tries to consume network resources, devices, or host systems and flood it to the point that it is no longer available for normal users to access.

- Fuzzy C Means: This algorithm is a clustering methodology where each data point is assigned to the cluster based on the distance between the data point and the cluster's centroid. The results (both clusters centroid and the data points assignment to a cluster) are recomputed till we get an efficient and effective result.
- HTTP: Hypertext transfer protocol is used for communication between a browser and a web server or "between intermediate machines and web server." It operates at the application level and uses TCP as a reliable transmission method since it does not provide its own reliability.
- HTTP-GET request: This is the browser that sends the GET request and to this the server responds with the information that was requested.
- HTTP-POST request: This sends the data to the server and it usually done using HTML form.
- I/O bandwidth: Bandwidth is the rate at which data gets transmitted. I/O bandwidth means the rate at which the data is transmitted to (input) or from (output) a device.
- ICMP: Internet control message protocol is used by routers to send error or control messages to other routers or hosts. It provides communication between various Internet Protocol software.
- IPV4: This is the fourth version and one of the core Internet protocol. It is described in RFC 791.
- IPV6: This is the sixth version of Internet protocol. It is described in RFC 2460.

- ISP: This is the Internet Service providers like Comcast, Qwest, and others. They provide services for users to access or participate in the online world (Internet). ISP's can provide these services through multiple methods like DSL, dial-up, cable, and wireless.
- K-means: This is an unsupervised clustering algorithm where the number of clusters (k) is fixed beforehand. The data points are assigned different clusters based on the data points features. Each cluster has its centroid
- KNN: K-nearest neighbor is a classification method where the probability of a data point (X) belonging to one group or another relies on the distance of other data points (belonging to a group) closest to X.
- Machine Learning: This is a form of computer learning done using various algorithm that are fed data for different purposes to create an autonomous system that can perform a task at high efficiency and speed. The algorithms that are fed data improve over time as they learn new patterns and behaviors.
- Multinomial Naïve Bayes: In multinomial Naïve Bayes, each data point from one classification is related to another data point from a different classification. It also takes into consideration of multiple instances.
- Naïve Bayes: This algorithm works using Bayes theorem that relies on classification of objects based on the features. It is probabilistic since each data point can be reassigned if new developments are made. Each datapoint is independent of the other.
- NAT Table: The Network Address Translation table is the most important part of the NAT functionality. It is processed within a router or a NAT enabled device for incoming and outgoing packets. The table keeps track of all the re-assigned IP addresses and ports.

This helps in sending the message to the right local host IP received by the public through translating the public IP to local host IP and port number.

- OSI Model: This is called Open Systems Interconnection Model created by International Organization for Standardization (ISO) which contains seven layers in this order: Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer, and Application Layer.
- PCA-Naïve Bayes: PCA is a statistical method that is used for extracting relevant information from multidimensional datasets. This is then combined with Naïve Bayes algorithm used in certain situations.
- Python: This is an object-oriented language that has clear syntax with its ability to work across various platforms like UNIX, MAC, and Windows.
- SCAPY: This is a python program mainly used to inspect, send, sniff, or forge network packets.
- Sci-kit Learn: This is a machine learning library that consists of algorithms like SVM, Random Forests, K-means, DBSCAN for classification, regression, and clustering; created using python programming language.
- Slowloris: This is a DDOS attack conducted from a computer requiring minimal bandwidth by sending partial HTTP requests to the target's web server.
- SMTP: Simple Mail Transport protocol is used to send mail over the internet through applications like Outlook, Thunderbird, and eM client. It is simpler than MTP where the communication between two machines/servers contains readable ASCII text.

- Smurf Attack: Smurf-malware is a way to conduct DDOS attack at Network layer using ICMP and IP protocols' vulnerabilities.
- Sockets: This is related to network. A computer application/program establishes connection with the internet through socket; which allows data to be read and written over the network.
- SQL Injection: This is a vulnerability related to the database. It is done by injecting malicious code into an application that is not designed keeping security in mind. Any vulnerabilities in the database related to sql queries can be exploited; thus, giving the attacker the access to potential sensitive information.
- SVM: Support Vector Machine is a machine learning algorithm used for classification and regression analysis.
- TCP: Transmission Control Protocol is a standard that uses three-way handshake to establish and maintain reliable network communication via IP network; which is used by application programs to exchange data.
- UDP: This is called User Datagram Protocol that sends messages between machines using IP through connectionless delivery service. It is best used for VOIP and DNS lookup due to its effectiveness in fast message delivery as it does not use any "handshake" method for secure transmission.
- Wordpress XMLRPC: This method of data transmission is done using HTTP as transportation method and XML to encode the information. It makes communication of Wordpress with different systems efficient.

- XSS: Cross-Site Scripting (XSS) is an attack that inject malicious code into a web application. So, once a victim opens a website containing that injected code, it (victim's computer/device) gets infected with the malware or virus.

Summary

This chapter focused on the brief introduction of DDOS attacks, the problems associated with it, detection methodologies that may be used, questions that need to be answered related to this study, and the shortcomings of this research paper.

Chapter II: Background and Review of Literature

Introduction

The chapter "Literature Related to the Problem" focuses on analyzing attack scenarios/methodologies on transport (Layer 4), network (Layer 3), and application (Layer 7) layers of the OSI Model and the vulnerabilities present within these three layers that are exploited to carry out a successful DDoS attack. The most widely used protocols within these layers to carry out DDoS are UDP, TCP, ICMP, and HTTP methods. After careful description of the problem the next section "Literature related to the methodology" delves into various detection techniques to help identify anomalies within the network infrastructure. After careful and rigorous research into various methodologies used to detect DDoS, machine learning is the most promising one due to its capability to adapt to new varied attack patterns and behaviors.

Background Related to the Problem

The main focus of this section would be addressing the attack scenarios (problem) to understand the traffic flow and how one can carefully differentiate between normal flood traffic (normal surge times during the day-where this is observed on a regular basis) and the anomalous flood traffic that can be detected through an identifiable pattern. This section would also focus on the slow-rate application layer attacks that are more difficult to identify. After a thorough explanation of the attack scenarios, this section would continue with the analysis of research articles' detection techniques as an "exploratory research" to provide future researchers with useful information for developing an efficient machine learning algorithm for DDoS mitigation.

Literature Related to the Problem

According to Sieklik (2016), web threats were almost up to 600% by year 2015. Most of these web threats were DOS and DDOS attacks. This article talks about Amplified DOS attack which is causing havoc in the past few years. DOS Amplification is the most frequently used attack methodology. An attack in 2013 on Spamhus caused a peak flow of 300 Gbps and there were other reported attacks in 2014 of which one was of 421 Gbps. The DDoS Amplification attack could be carried out using uninfected machines suffering from 'protocol or other flaws' (Sieklik, 2016). Though DDoS attacks are quite significant, the complex DDoS attacks are more of a concern as per the article (Sieklik, 2016). There are some claims made that DDoS attacks take almost about 5% of the total Internet traffic which is a huge number when it results in a bandwidth of terabytes per second regardless of the time in the day. Moreover, the survey done by Corero Network Security shows that 38% of United States enterprises experienced DDOS attacks in 2013 whereas 42% of these companies have gone through multiple attacks (Vadlamani, 2013).

Transport Layer Attacks

TCP-SYN: SYN Flooding is an attack where the client sends SYN packets continuously to every single port possible on a server using spoofed IP address. The server responds to the client with ACK packets but the client (hostile) never responds to these servers with an ACK packet. Instead, the client keeps sending SYN packets and since its IP is spoofed there is no way for the server to close the connection by sending RST packets. Its connection stays open and before there can be a time-out, the server receives another SYN packet from the hacker's machine (client) and this is SYN Flood attack which is also called half-open connection. The

reason it is called half-open is due to the fact that a TCP connection needs a three way handshake between a client and the server in order to establish a connection whereas in a SYN flood attack only SYN packets sent to the server is needed, to which the server sends an ACK packet but never receives SYN-ACK (acknowledging the request to connect) from the client. In a three-way handshake the client sends SYN packet to which the server sends an ACK packet and finally the client sends SYN-ACK packet to the server. The SYN packets consume the server to an extent that it is unable to receive any communication from legitimate clients. UDP flooding is like ping attacks where continuous flow of these packets is sent on the ports of the server to make it unavailable for the legitimate traffic (users). The maximum size of its packet could be set to 65000 bytes leading it to flood an Ethernet network in case of multiple zombies being used. UDP is considered, to be much more effective on a smaller network since the size of their packets is huge (Subramani, 2011).

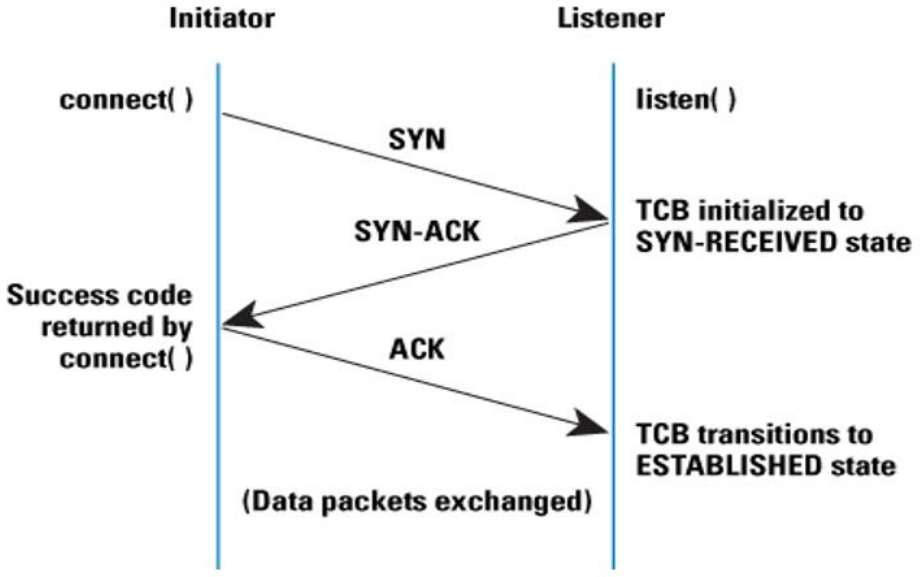

Figure 1. TCP 3-way Handshake (Eddy, 2013).

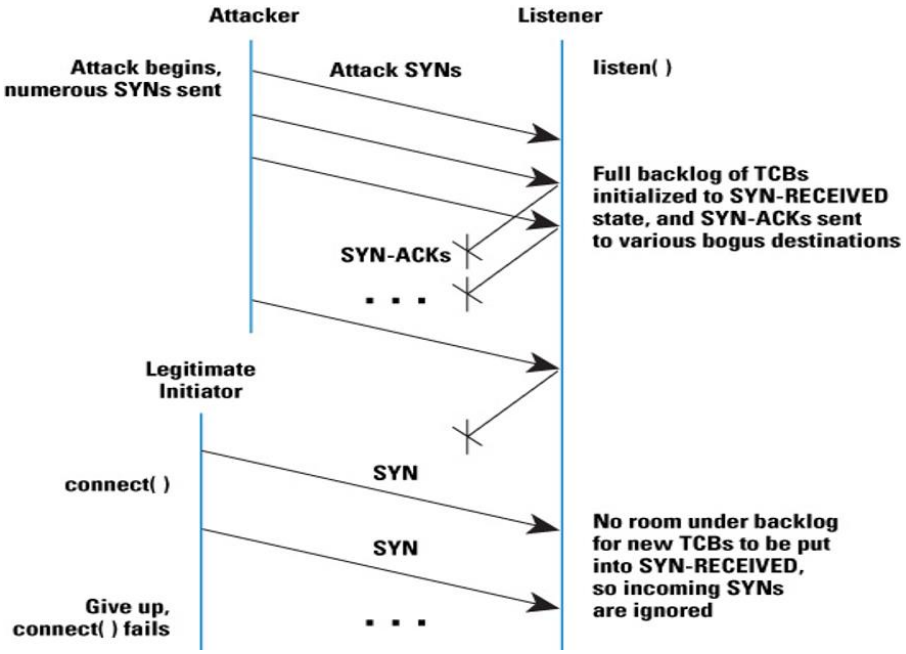

Figure 2. TCP SYN Flood Attack Demonstration (Eddy, 2013).

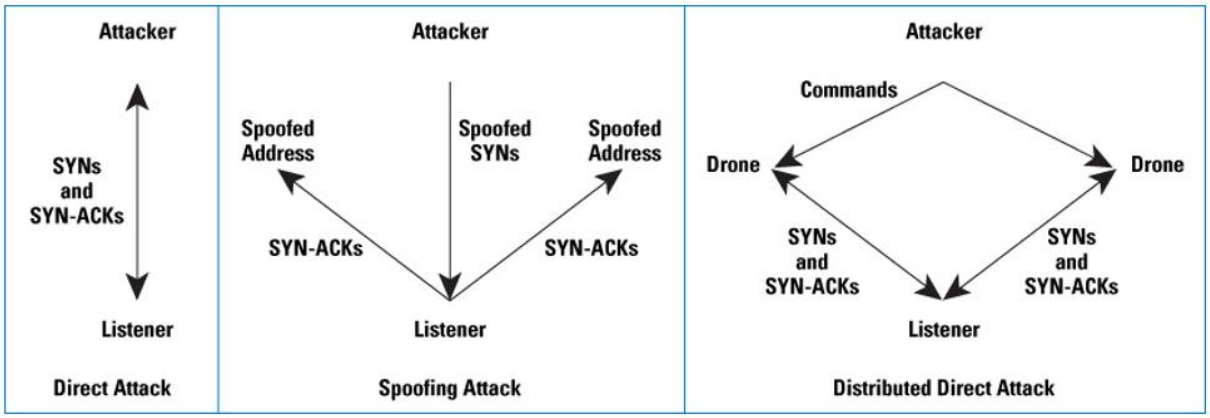

Figure 3. Variants of the Attack (Eddy, 2013).

- **TCP Reset**: Forged TCP resets are sent to the victim's server/host by sniffing victim's IP and port addresses as well as current TCP connection sequence numbers. The attacker gathers the required information to forged reset packets by tricking the victim to connect

to a malicious Wi-Fi or by sniffing WIFI packets if the victim is in close range. The attack is carried out using SCAPY in python through the tcp-reset.py script.

- **TCP Fragmentation**: The attack is carried out to bypass firewall's filtering rules by fragmenting TCP packets with the header flag field pushed into the second fragment. Another method involves overlapping fragments where the first segment passes through the filter whereas the second fragment overwrites the TCP header data with malicious payload. These fragments could also be sent at a low time-rate to avoid any timeout for any of these fragmented packets' connection which later puts load on the resources trying to assemble the packet and waiting a long time for defragmentation. At a higher layer it creates overhead in the bandwidth (for transmission) since each fragment possesses both L2 and L3 headers. In this case smaller fragments lead to higher overhead.
- **UDP flooding**: It is a little like ICMP flood attack except the server does not reply-back to the attacker's machine. It rather involves sending extensive amount of large UDP packets to either random ports or a destination address. Moreover, it is easily spoofed as it is a connectionless protocol which does not have the three-way handshake method of connecting with the server. Most of these attacks are executed at the firewall level to block the internet connection making it inaccessible for the normal users. Here is the diagram showcasing the working of the attack:

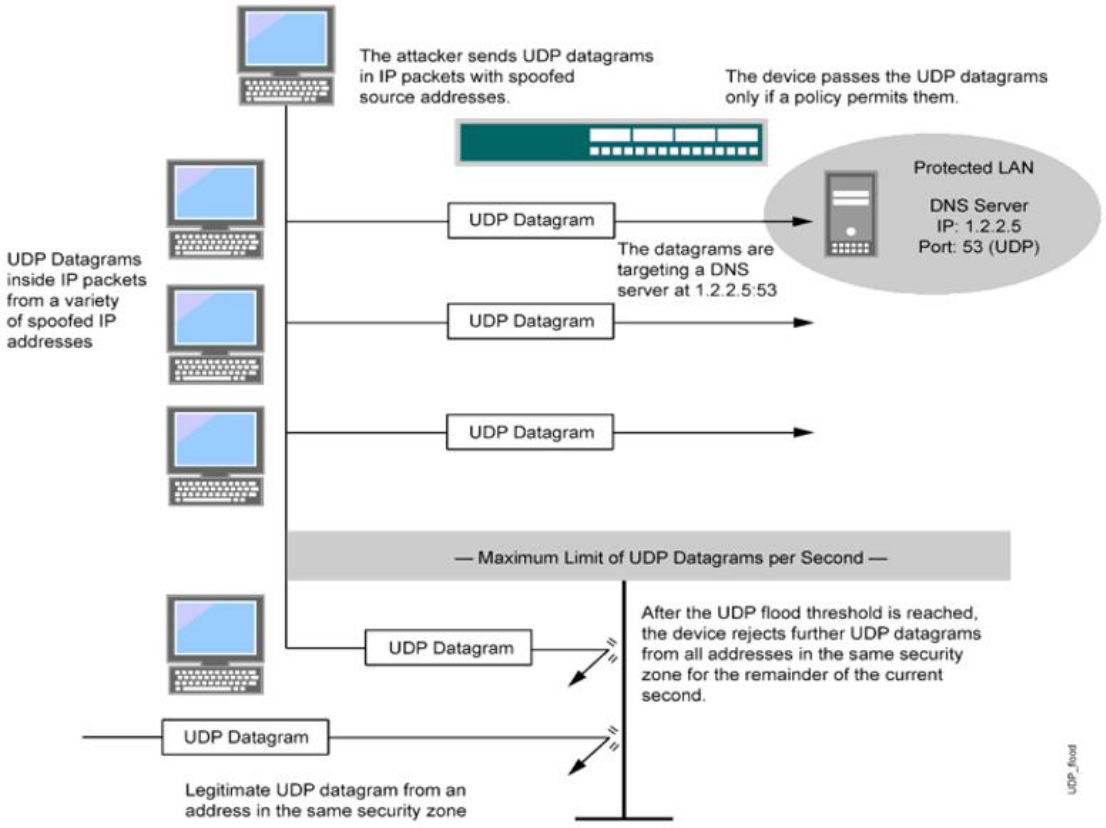

Figure 4. UDP Flood (Juniper Networks, 2016).

Network Layer Attacks

ICMP or Smurf attack. ICMP flood attack is basically ping attacks done on the victim's server which involves an attacker consuming the server's resources with ping packets (echo requests) using spoofed IP addresses. So, when the server tries to echo reply back to the user (attacker) it is unable to reach the destination as its IP address is spoofed. ICMP flooding or ping flooding is where a host (victim) gets sent thousands of ping packets. Ping of death is where the victim gets sent corrupt packets leading to a possible system crash. The article by Juniper Networks (2016) explained the workings of the ICMP attack using the diagram shown below:

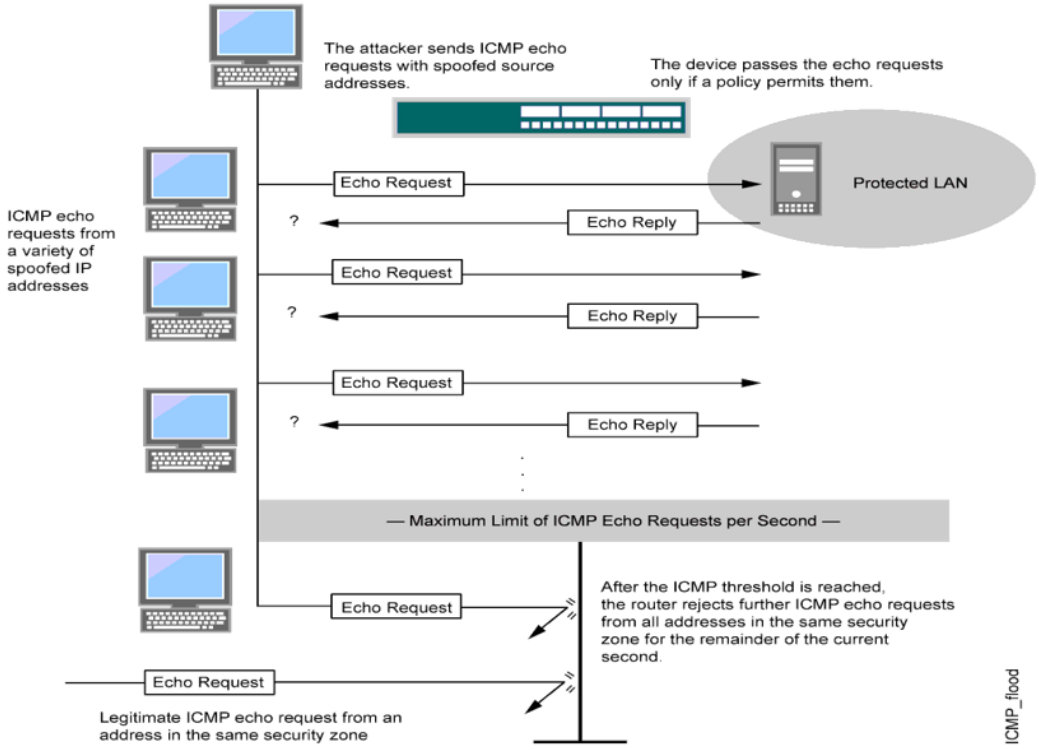

Figure 5. ICMP Flooding (Jupiter Networks, 2016).

Application Layer Attacks

Application Layer Attacks are stealthier for the main reason that it is a low rate (low bandwidth) attack requiring minimal resources which then gets accompanied by network layer attacks. It is targeted for the purpose of causing disruptions in transactions, search functions, web browser function, email services, and access to databases. This leads to consumption of available memory, server dependent requests; or acquiring sensitive information like IP address and version of the software used. This attack falls under DDoS only when more than three to five nodes on various networks are being consumed otherwise it's a DOS attack (Patrikakis, Masikos, & Zouraraki, 2015).

There are two major application layer DDoS attacks: HTTP Flood and Slowloris among others such as DNS dictionary attacks, VoIP (SIP Invite Flood attack), and SMTP buffer overflow attacks (Kostadinov, 2013). There are three HTTP flood attacks which includes HTTP GET, HTTP POST, and HTTP Slow read. HTTP GET is where the attackers try to download a file from a website using large number of botnets; thereby consuming the webpage with thousands of download requests. A legitimate user ends up waiting for infinite amount of time for the file to finish downloading since the server or database powering the web page is flooded with attackers GET request packets. In a HTTP POST flooding attack, the hackers use a form like customer feedback, questions, or login page to send large number of http post request packets. The hackers can also create an XSS script attack using http post method. This could further lead to other attacks such as SQL injection thereby corrupting or deleting the database. This attack type can also focus on exhausting the server resources like sockets, CPU, memory, disk/database bandwidth, and I/O bandwidth.

Many companies these days are implementing CDN (content distribution network) service which is caching the web content on separate servers located at different geographical locations to help provide the users a fast access to the website. CDN has its own firewall implemented but attackers can bypass these firewalls and reach the backend server. The vulnerabilities found within this system of which one out of many others is “login and authentication pages are not cached but are linked to the main server” which also stores the users’ database. Hackers can execute HTTP POST flood attack using either the login or authentication pages. This is more of a slow-send HTTP POST attack which can be executed using the Slowloris tool as it delves into tying up server’s resources through continuous flow of

incomplete legitimate HTTP request packets that is in open state waiting for the completion of the request (Gonzalez, Stakhanova, & Ghorbani, 2013). The slow read attack is where the attacker sends a legitimate HTTP request to the server and then reads the response at a very low speed. Since, the attacker sends the zero TCP window size (or close to zero) to the server, the server is in the pretense that the attacker is reading the data, thereby keeping the connection open. Its main idea is to drain the TCP receive buffer, leading to a heavy consumption of the server's resources, and thwarting normal users' requests for connection to the server.

These attacks can be mitigated by the ISP or an enterprise level organization using BGP that reroutes the traffic to a DDOS mitigation provider, such as DDOS-Guard, Redware, NetScout (just used as an example and not promoting it or concluding it to be great or bad), which are also called scrubbing centers that analyzes and scrubs the malicious traffic.

Literature Related to the Methodology

After thorough and intensive research into various methodologies like signature-based algorithms used to detect DDoS, machine learning seems to be the best way as they show the promise in terms of adapting to the varied behavior of the attacks. Overtime it can learn from its past attack patterns and attack algorithms, provided the method to detect is a supervised and/or semi-supervised machine learning algorithm (since unsupervised algorithm though could be effective, the results might not always be as desired and the algorithm's adaptive learning strategy is still under development). After doing intensive research and study on machine learning algorithms, the unsupervised algorithm is the most unpredictable in terms of detecting and has the potential to generate high false positive rate.

To be able to implement ML algorithm (especially creating a new one) it is recommended that one has enough knowledge and proficiency in statistics, calculus, and python to be able to understand the workings and implications; so as to ensure its efficiency. One can still use the ML algorithms that are already released and proven to be effective, but a basic understanding of statistics and python can go a long way in terms of understanding the use of parameters (supervised and semi-supervised) in determining the outcome of the results. Some of the well-known ML algorithms used for DDOS detection are SVM, KNN, K-means, Fuzzy C Means, and Naïve Bayesian.

Another subset of machine learning is deep learning which has the potential to look for minor anomalies within the data-packets which can be conducted using structured supervised and/or semi-supervised implementations.

Transport Layer Detection Techniques

Research Article 1. As per the article by Hoque, Bhattacharyya, and Kalita (2016), a covariance analysis model is considered to be very useful in detecting SYN flooding DDoS attacks or any other attacks which shows activity for even minimal abnormal behavior (DDOS attack). Another method proposed by Peng (2004) (another source used within this article (Hoque, 2016) is a sequential nonparametric change points that uses IP addresses to analyze network traffic and detect the bandwidth DDoS attacks. This methodology is very useful as it detects the attack way before it is executed on the server. For low rate DDoS attack such as ‘generalized entropy and information distance metric’(Hoque, 2016), a trace-back methodology which adjusts the value of order α of the generalized entropy and information distance metrics

could be useful according to Xiang. Furthermore, the article talks about its own methodology for detecting the attacks called FFSc measure that analyzes every single network traffic sample and object. There are two phases of this methodology i.e. '(i) normal traffic analysis phase and (ii) captured traffic analysis phase' (Hoque et al., 2016). Figure 6 shows the description of the methodology:

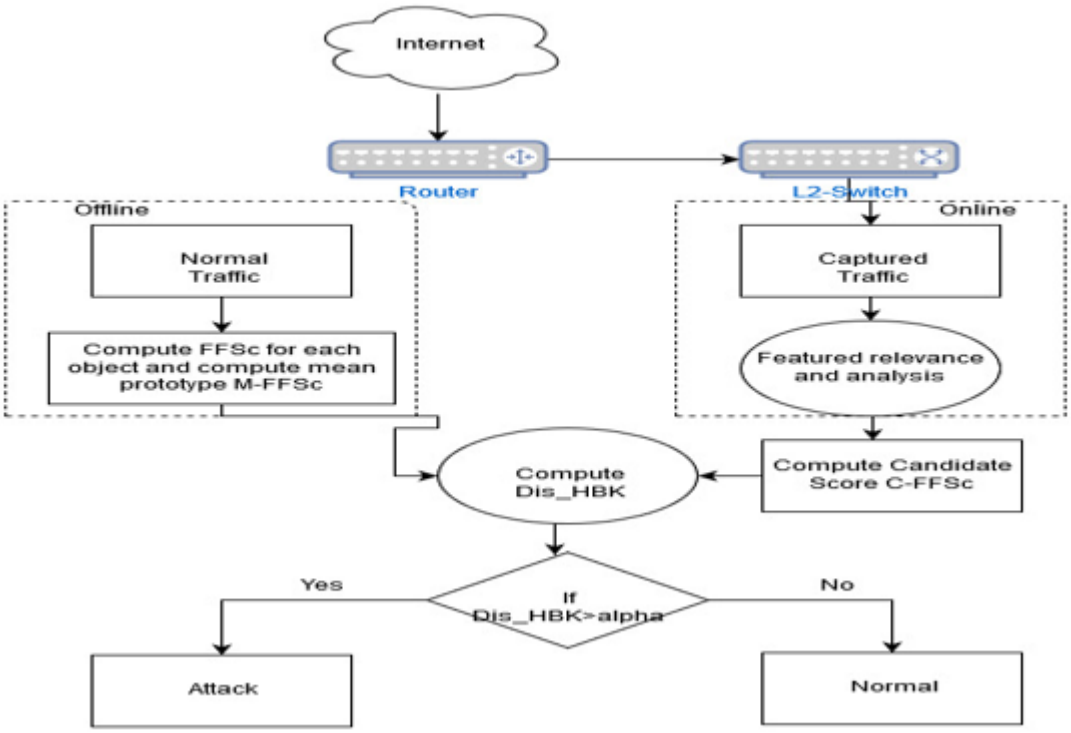

Figure 6. FFSc Methodology to Detect Attacks (Hoque et al., 2016).

Research Article 2. This article mentions using SVM as the detection technique for DDoS attacks. SVM is used as a classifier to help get the best margin between training patterns and decision boundaries. It is a part of kernel methods to prevent the quadratic growth in the memory. They used feature selection method by implementing genetic algorithm created as a parallel algorithm using MapReduce program focusing on extracting source and destination ports, sequence and acknowledgement numbers, the SYN and ACK-TCP flags, and TTL for the

classification of anomalous vs. normal traffic. In respect to extracting the anomalous and normal traffic they used TShark from the pcap files; where their specific study had total 7126 feature vectors out of which 4999 were DDoS and 2127 were normal traffic. Later this data obtained by SVM was evaluated using cross-validation method that divides data into various partitions; and these partitions are separated to be used either for testing or training. This study created an equal 10 partitions where one partition was used for testing and the remaining nine were used for training. Later this process was repeated for about 10 times to get the average of the cross-validation method. Additional measures like sensitivity, specificity, average classification accuracy, F-measure and ROC curve were taken to evaluate the accuracy of the metrics (Masetic, 2017).

Sensitivity is a measure that identifies the performance of its classifier in determining true positives; which is defined by the formula $\frac{TP}{TP+FN} * 100$ (TP=number of true positives, FN=number of false negatives) (Masetic, 2017).

Specificity is a measure that identifies the performance of the classifier in determining true negatives; which is defined by the formula $\frac{TN}{TN+FP} * 100$ (TN=number of true negative, FP=number of false positives).

Average classification accuracy looks into the average of both the True Positives (Sensitivity) and True Negatives (Specificity); which is defined by the formula $\frac{Sensitivity+Specificity}{2}$.

F-measure checks for the performance of the model itself; which is defined by the formula $\frac{2TP_i}{2TP_i+FP_i+FN_i}$ (TPi=true positive rate of ith class, FPi= false positive rate of the ith class, FNi= false negative rate of the ith class).

ROC curve is a measure that identifies the ability to differentiate various statistical methods by plotting true positives on y axis and false positives on x-axis. Later the classification performance is measured by calculating the various means of "area under the curve" (Masetic, 2017). The size of the area directly corresponds to the effectiveness and efficiency of the classification model (large area=more efficient model).

Using all these factors and analytical methods they were able to generate results that showcased that their algorithm was able to detect true positives and true negatives with the accuracy of 100%.

It seems to be very promising in the training environment but when this method was tested in the actual environment where attacks were performed by various outside (similar to a production environment) members than this study proved to be less accurate and efficient. This proves that the algorithm though effective needs a long duration of learning time for the researchers to have the ability to modify the methodology design as necessary.

Research Article 3. This research focused on using three different clustering methods namely K-means clustering with single arbitrary set of initial k-means, K-groups of clusters, and K-means clustering with optimal initial means. The ML training datasets used for this study was MIT DARPA, Auckland-I, Auckland-II, and SETS.

As we know about the TCP protocol, it has dependencies when in communication between two parties in respect to the flags generated. For example: SYN/ACK is followed by a

SYN packet received from the client and an ACK packet is followed by the SYN/ACK received from the server to the client. It is important to ensure that combinations like SYN/ACK and FIN/ACK are not considered malicious as it is part of normal traffic and communication. The idea of this research in respect to detecting TCP protocol DDoS attacks were to look within the abnormal combination of TCP flags (generally never seen in a normal traffic-like XMAS (FIN/PSH/URG) attack, SYN/FIN, SYN/FIN/PSH, SYN/FIN/RST, SYN/FIN/RST/PSH, and others (total of 15 combinations)), windows size burst error caused by zero-windows-size, fragmented packets less than 400 bytes for IPV4 and less than 1280 bytes for IPV6, acknowledgement number, sequence number, and the payload within the traffic received by the server. There were also parameters for detecting valid flag combinations that are rarely seen in traffic (13 combinations in this research).

Using these parameters, a probability distribution was created as follows:

1. Observables were grouped into 6 (T1 to T6).
2. Discrete Random Variable Xiper observable group i= number of packets of type Ti observed in a window.
3. Outcome set Oi= {0,1,2,3. . . N}, where N= window size.
4. The probability of observing ci packets of type Ti in a window is P(Xi=ci).
5. 5.P(Xi=ci) =number of windows with ci packets/T, where T= Total number of training windows.
6. Probability of a window W with c1 packets of T1, c2 packets of T2…c6 packets of T6 (c1+⋯c6 = window size N) =P(X1=c1) ∗⋯P(X6=c6), by NB independence assumptions. (Vijayasarathy, 2012)

Research Article 4. Multiple algorithms were used in this research paper to study the effects of Transport Layer attacks (namely UDP and TCP). Classifiers such as Naïve Bayes, MLP-ANN, Decision Trees, and SVM were used to compare and analyze the accuracy, precision, recall rates of both DDOS and normal packets (Alkasassbeh, 2016).

Decision trees are non-parametric supervised learning method widely used in data mining for classification and regression analysis. The main idea of using this methodology in this study is to create an efficient model predicting the value of a target variable through continuous learning of simple decision rules that are implied using the data features. The structure of decision tree contains three nodes: root node which is the top node that has no input but zero to multiple outputs, internal node has one incoming input and two or more outgoing outputs, and leaf node has one incoming edge (input) and no outgoing outputs. The parameter used to categorize transport layer traffic into normal, anomalous, or other is source port, destination port, number of SYN flags from a source to a destination port, packet id, packet type, packet size, flags, FID, sequence numbers, number of packets, number of bytes, packet-in, packet-out, packet-rate, byte-rate, packet average size, utilization, packet delay, packet send time, packet reserved time, first packet sent, last packet received, node name from and to, and from and to a node.

Naïve Bayes was used by first setting a set a variables such as $X = \{x1, x2, x3, \dots xn\}$ for the event C_j where $C = \{c1, c2, c3, \dots cn\}$. Based on Bayes theorem we get $p(C_j|x1, x2, x3, , xn) \propto p(x1, x2, x3, , xn|p(C_j).P(C_j))$; where p($C_j|x1, x2, x3, , xn$) is the probability, such that X belongs to C_j. Based on the Naïve-Bayes algorithm which defines the conditional probability of independent variables to be statistically independent, the original

equation ("product of terms") is rewritten from $p(X|C_j) \propto \prod_{k=1}^{d} p(x_k|C_j)$ to $p(C_j|X) \propto p(C_j)\prod_{k=1}^{d} p(x_k|C_j)$, where C_j represents jth class of classes {1,2,3…v}, x_k represents feature vectors of kth sample of samples {1,2,3…w}, $p(X|C_j)$ is the probability to observe the pattern x given that it is in class C_j, $p(C_j|X)$ is the probability of pattern/event x occurring given that it belongs to class C_j; and a new case X is now labeled with class C_j, thus achieving the highest posterior probability. Posterior probability is the probability of a sample i belonging to a class j given its observed feature-vector $x_k$values.

Based on another research article's (Riadi, 2017) explanation of the ANN algorithm which is used to detect transport and network layer DDOS attacks, can help support the results of this research study. So, in brief ANN is a computational model that is inspired by biological neural networks found in human brain composed of neurons with various processing capability. ANN can have either single or multiple layers of neurons. The study by Sofi, Mahajan, and Mansotra (2017) used confusion matrices as part of the ANN algorithm; which is basically a form of visualization that helps in understanding the performance of the neural network. The paper also used SVM to help in distinguishing and separating IP addresses based on different classes (normal and anomalous). The specific SVM used is the (C-SVC) c-support vector classification for training the algorithm and then testing the normal and attack traffic datasets. The datasets were created by the researchers using the process of collecting and auditing network traffic, preprocessing it to remove redundant data, and then using feature extraction to identify relevant parameters for detection.

The dataset is evaluated using the WEKA 3.8 tool in Ubuntu 16.04 LTS platform. The algorithms are trained using the 66% collected data and the rest 34% using test data created in

the experiment phase. They also used k-folds (here k=10) cross validation, where the original sample is randomly partitioned into 10 equal sized subsamples. Out of the 10 subsamples, one subsample is reserved as the validation data for testing the models and the remaining nine samples are used as training data. The class label of training data was eliminated through the conversion of csv file format to ARFF file format. Including the class label there are 28 attributes present, which is evaluated for performance using the confusion matrix as presented in Tables 1, 2, and 3 (Alkasassbeh, 2016).

Table 1

Confusion Matrix for MLP (Alkasassbeh, 2016)

	Normal	***UDP-Flood***	***Smurf***	***SIDDOS***	***HTTP-FLOOD***
Normal	657961	0	0	70	20
UDP-Flood	6767	61765	0	0	0
Smurf	2817	0	1396	132	10
SIDDOS	115	0	0	2136	0
HTTP-FLOOD	0	0	0	86	1352

Table 2

Confusion Matrix for Random Forest (Alkasassbeh, 2016)

	Normal	***UDP-Flood***	***Smurf***	***SIDDOS***	***HTTP-FLOOD***
Normal	653545	3117	1258	112	19
UDP-Flood	6728	61794	10	0	0
Smurf	2798	10	1414	121	12
SIDDOS	160	0	75	1972	44
HTTP-FLOOD	8	0	11	77	1342

Table 3

Confusion Matrix for Naïve Bayes (Alkasassbeh, 2016)

	Normal	*UDP-Flood*	*Smurf*	*SIDDOS*	*HTTP-FLOOD*
Normal	646612	0	10375	981	83
UDP-Flood	6705	61765	59	3	0
Smurf	2717	0	92	140	1406
SIDDOS	115	0	16	2120	0
HTTP-FLOOD	0	0	0	86	1352

The Confusion Matrix is calculated using the following equations for accuracy, precision and recall:

Accuracy is where it measures the rate of the correctly classified attack instances of both classes"(Alkasassbeh, 2016) represented as $Accuracy = \frac{TP+TN}{TP+TN+FN+FP}$

"Precision is the ratio of the number of relevant attacks retrieved to the total number of irrelevant and relevant attacks retrieved" (Alkasassbeh, 2016) represented as $Precision = \frac{TP}{TP+FP}$

Recall is a ratio of the number of relevant attacks retrieved to the total number of relevant attacks, which is represented as $Recall = \frac{TP}{TP+FN}$

Network Layer Detection Techniques

Research Article 1. To detect DDOS attacks, Saied (2016) talks about using Artificial Neural Network (ANN) Algorithm. This algorithm is focused on detecting known and unknown attacks in real time while separating the DDOS attack traffic from genuine traffic to make sure that the genuine traffic is not dropped. Here is the image showing the design of the methodology of ANN:

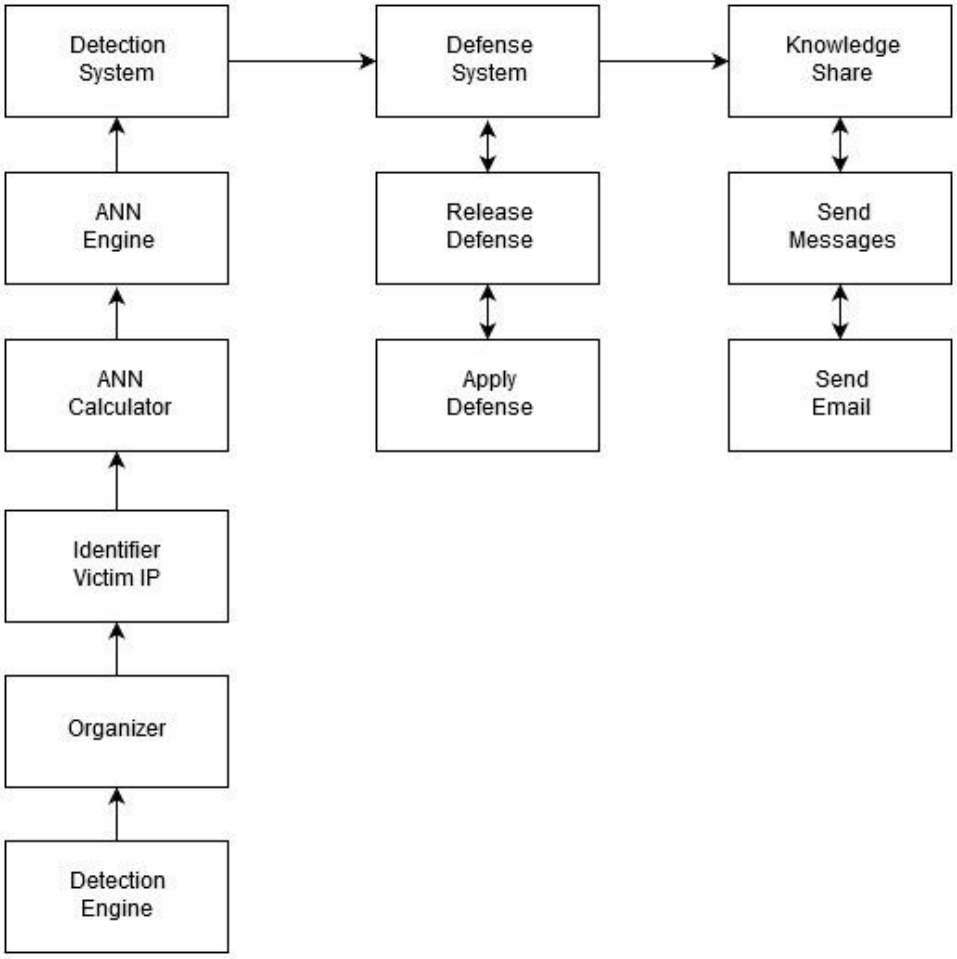

Figure 7. Design for ANN Methodology (Saied, 2016).

Following is the explanation of how this methodology shown above works as per Saied (2016):

1. "DDoS detectors are installed on different networks.
2. Each detector registers the IP address of all neighboring DDoS detectors to inform and send encrypted message when DDoS attacks are detected.
3. Detectors continuously monitor their networks for abnormalities.
4. Abnormalities are flagged when the number of passing packets is greater than a predefined threshold for each protocol.

5. If the number of packets is greater than the threshold, then:
    a. The organizer sorts the packets accordingly.
    b. The IP identifier identifies the victim's IP addresses.
    c. The ANN calculator calculates retrieved patterns and prepares them for the ANN engine
    d. The trained ANN engine takes the patterns as inputs and produces one output (1=attack or 0=normal).
    e. Before activating the defense system, the steps under point 5 above are repeated twice more, producing a total of 3 outputs.
6. The defense system receives the outputs from the detection component and:
    a. If the outputs are (0,0,0), then no action is required by the defense system, as the traffic is clean.
    b. If the outputs are (1,1,1), (1,1,0), (1,0,1) or (0,1,1), it activates defense and stops the attack while allowing genuine traffic to pass through.
    c. If the outputs are (1,0,0), (0,1,0) or (0,0,1), the solution repeats point 5. If the outputs of the new retrieved traffic are:
        I. (1,1,1), (1,1,0), (1,0,1) or (0,1,1), activate the defense system.
        II. (1,0,0), (0,1,0) or (0,0,1), deploy mitigation as this is considered to be a low rate attack.
        III. (0,0,0), no actions are required.
    d. However, if the output is none of the above, then the system generates value 2. This means the traffic is unidentified (not used in training) by the ANN. At this

point, the solution checks its local record to learn if the same traffic is received and detected by neighboring DDoS detectors. If the received traffic from the neighboring detectors is 1 or 0 then the algorithm is outdated since its detection was 2 while other detectors identified the traffic as 1 or 0. This means the algorithm on the local DDoS detector needs to be retrained (off-line) with both existing and new patterns. Otherwise no action is taken.

7. The knowledge sharing component sends encrypted messages containing the type of the attack; destination IP address and protocol involved to all registered neighboring DDoS detectors. Such information is also composed by the email element and sent to the security offices to take countermeasures if required or for the purpose of logistics or forensics." (Saied, 2016, pp. 385-393)

If this methodology gets proven to be successful and is implemented, then it could help in minimizing killing all traffic (incl. genuine traffic) while trying to protect the system from DDOS traffic.

Research Article 2. The proposed method for detection of ICMP attacks in this research experiment was Semi-Supervised Fuzzy C-Means Clustering. The process used to detect and for learning stages is as follows:

I. Learning stage involves:
    a. Forming a set of parameters based on past and current knowledge obtained from various researches and the IT security industry that are relevant in the detection of DDOS attack.

b. Showcasing the obtained information about these attacks based on the parameters studied to identify the attack.

II. Detection stage technique involves:

a. Collecting inbound and outbound traffic for our dataset

b. Extracting relevant parameters from the traffic that can be used for the identification of a potential DDOS attack and to create parameter-vectors.

c. Building the parameter-vectors from the information available in the network traffic.

d. Implementing the algorithm as per the parameter-vectors used to label it as either attack traffic or normal.

e. Isolation of hosts that are affected by the attack.

The parameters/features used to help identify an attack are p (transmission protocol), f_{IO} is a Boolean feature to identify if an inbound traffic has a correlated outbound traffic, p_{OD} is the number of packages that are transmitted on its journey from origin to destination, b_{OD} is the "number of bytes transmitted from origin to destination" (Lysenko & Savenko, n.d.), d_C is the time-duration of the connection, d_{EL} is the duration of the connection as it is observed from the starting of the inbound and/or outbound traffic until the end of the connection, l_p is the average payload length of each connection, b_{TC} is the total number of bytes transmitted per connection, b_{EH} is the total number of bytes per connection while eliminating the header, n_{PSF} is the number identifying various sized packets transferred to total number of framer per connection, p_s is the total number of packets in a session, b_s is the total size for each session in bytes, d_{PSB} standard deviation of packet size within the session and it is measured in bytes, v_{OBP}, v_{IBP} is the "velocity

of outbound/inbound traffic which is measured in bytes per packet" (Lysenko & Savenko, n.d.), v_{OBS}, v_{IBS} is the "velocity of outbound/inbound traffic measured in bits per second" (Lysenko & Savenko, n.d.), v_{OPS}, v_{IPS} is the "velocity of outbound/inbound traffic measured in packets per second" (Lysenko & Savenko, n.d.), o_{SS}, i_{SS} is the "self-similarity of the outbound/inbound packets in the session, determined by examining the variance in size of the outbound/inbound packets using the Hurst component" (Lysenko & Savenko, n.d.), n_{DP} is the total number of packets denied, n_{NAT} is the "number of packets in the NAT/PAT table" (Lysenko & Savenko, n.d.), n_{ARP} is the number of ARP requests, f_{TCP} is the invalid TCP flag values present in the session, f_{GEO} is the geo-location parameter based on an IP address, p_R is the value of the router's processor time represented in %, m_R is the size of the memory used by a router in megabytes, and s_{RT} is the server's response time in milliseconds. The feature vectors are labeled to create clusters that are separated based on the information collected from the parameters to define attack vectors. The presence of the feature vector x_k in the i-th cluster either shows a presence of DDOS attack or absence of it. To compute the centroid of the i-th cluster, they need to assume the labeled data v_i. "Each feature vector x_k of the labeled data belongs to one of the predefined clusters" (Lysenko & Savenko, n.d.). The algorithm is basically dependent on the minimization of this objective function: $J_k = \sum_{i-1}^{c} \sum_{k=1}^{N} \mu_{ik}^{p} d_{ik}^{2} + \alpha \sum_{i=1}^{c} \sum_{k=1}^{N} (\mu_{ik} - f_{ik} b_k)^p d_{ik}^2$, where "N is the total number of the feature vectors that will need to be clustered (labeled and unlabeled feature vectors), μ_{ik} is the membership value for the k-th feature vector in the i-th cluster, f_{ik} is the membership value of the k-th labeled feature vector in the i-th cluster, d_{ik} is the distance between the k-th feature vector and the prototype of the i-th cluster, b=[b_k] which is the Boolean

indicator that distinguishes the labeled and unlabeled feature vectors such that b_k= 1, if feature vector x_k is labeled and 0 otherwise.

The centroid of the i-th cluster, vi, and the partition matrix μ_{ik} are calculated using the formula:

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^2 x_k}{\sum_{k=1}^{N} u_{ik}^2}, \quad u_{ik} = \frac{1}{1+\alpha}\left\{ \frac{1+\alpha\left(1-b_k\sum_{l=1}^{c} f_{ik}\right)}{\sum_{l=1}^{c}\left(\frac{d_{ik}}{d_{lk}}\right)^2} + \alpha f_{ik} b_k \right\},$$

, where ∝ is the scaling factor to maintain a balance between the supervised and un-supervised component within the optimization mechanism" (Lysenko & Savenko, n.d.).

The next step involves collecting the inbound and outbound traffic to be sent to the classifier for further analysis. Later, the local hosts are localized based on the MAC and IP addresses that are shown to be malicious. In this experiment, 80 hosts were used to carry out the DDOS attacks and it lasted for 24 hours. Network traffic was obtained through tcpdump functionality, where 15% of feature-vectors were labeled for the training dataset.

Application Layer Detection Techniques

Research Article 1. According to Ni, Gu, Wang, and Li (2013), the entropy of HTTP GET requests per source IP address (HRPI) is best suited for the application layer DDoS attacks since these attacks consist of the distribution of source IP address and HTTP GET request frequency. The precision of this methodology lies in using HRPI time series as a multidimensional vector through estimation of adaptive autoregressive model parameters using

the Kalman filter. Moreover, this detection technique also uses SVM (support vector machine) trained by AAR parameters of HRPI time series and then applied to identify the current state of network traffic and application layer DDoS attacks. The experiment was carried out in two groups; one consisting of detecting the attacks in normal traffic and the other in flash crowd. The attack itself was carried out in a NS-2 network simulator on Linux platform. The simulation consisted of 50 zombie machines and a web server. In this case, the attack rates were 20, 30, until 60 HTTP Get requests which mimic the attack rates of the worm "Mydoom" where every attack lasts for 1800sec. This methodology has shown results of high efficiency in detecting the DDoS attacks in flash crowd as well as in normal traffic. The detection time on average is 10seconds which is efficient considering that the sampling interval time is 0.1 sec and the HRPI time series of length N is 100. One of the main disadvantages of SVM algorithm is its difficulty to understand and analyze accurately as there are variations in the data set obtained and the choice of the kernel. In training and testing there are certain speed and size limitations (takes up more time to train for large data sets) which make it in-efficient for a volumetric DDoS attack. The memory required for detection of a volumetric DDoS attack is very high rendering its detection methodology almost useless for such attacks.

Research Article 2. This paper by Sreeram and Venkata (2019) focused on the bio inspired bat algorithm (developed by Xin-She Yang in 2010) which has proven to be very efficient as it can analyze a large set of data. It implements a sort of stochastic optimization technique where a set of randomly generated values are analyzed to come up with a solution. It can come up with a better solution using few computational resources/efforts as compared to the optimization algorithms or simple heuristics. Since the http flood attacks are randomly generated

and quite difficult to detect; this methodology is particularly efficient because it deals with randomly generated data. This research article focuses on the usage of bat algorithm methodology and machine learning metrics for detecting the HTTP flood DDoS attacks. It is considered a much more efficient technique as it can solve single and multiple objective optimization problems. It is derived from the study of bats' behavior using frequency modulated signals for echo location (this algorithm mimics the technique of echo location to detect anomalous behavior in a http flood traffic) which is a population based evolutionary algorithm.

Research Article 3. The article by Liao, Li, Kang, and Liu (2015) discussed a methodology which uses "cluster with label based on sparse vector decomposition and rhythm matching" (Liao et al., 2015, p. 3111) to detect these types of attacks. There are various methodologies mentioned in the paper invented by other researchers but has ample number of flaws and are difficult to implement it in real time. The methodology the paper chose to pursue is in relation to the four categories of application layer attacks which are single URL repeated attacks (it sends multiple repeated requests that can be conducted at fixed speed or not), multiple URL repeated attacks (this can have fixed speed or a variation in the speed), scanning of target websites and finding all page links of which random page links are selected for attacks, and session repeated DDoS attacks (considered the most difficult of application layer attacks to be detected). The methodology focuses on the user behavior patterns based on the extraction of certain features in server web log. It uses feature vectors to differentiate between normal users and attackers which is called SVD-RM (Sparce vector decomposing and rhythm matching). So when a user accesses a web page this methodology will take in affect where only useful information like request URL suffixes (HTML, HTM, and server parsed HTML), static and

active server pages, java server pages, common gateway interface, and dynamic page information (hypertext preprocessor) will be obtained and the rest of the redundant data will be filtered out.

The first algorithm is created to distinguish attackers from normal or hybrid users and it is used while assuming that users take more time browsing a web page compared to attackers as their requests are program generated. This screening filters out only 100% real users and the rest of records will have attackers and other real users. So, if it is 100% guarantee of a record to be from a real user then only it would pass through the first stage. Its structure looks something like this:

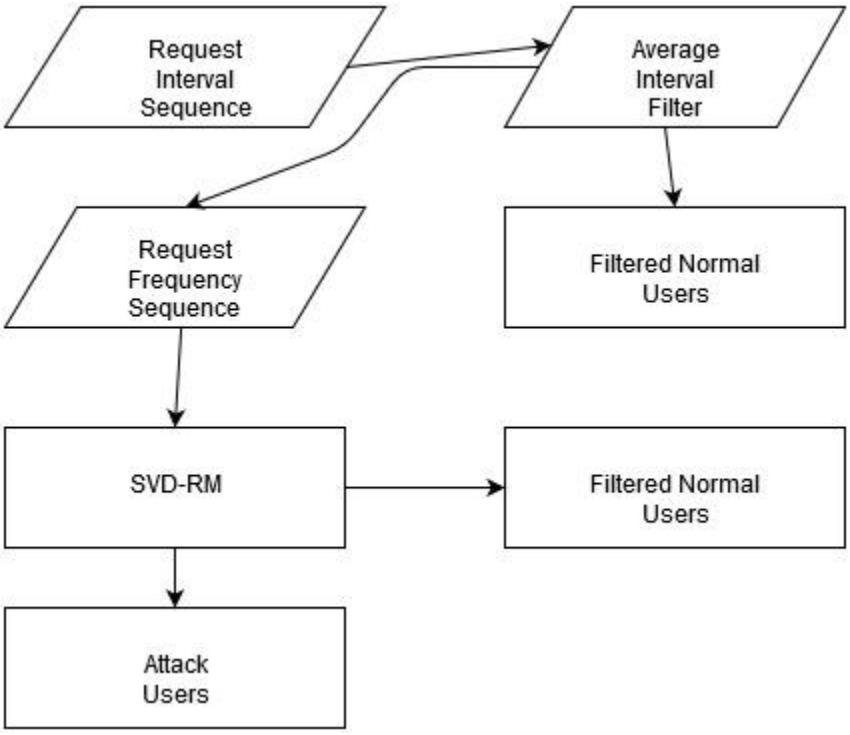

Figure 8. First Part of SVD-RM Detection Architecture (Liao, et al., 2015).

The mechanism (equation used) for this first part is as follows: “Let T represent the threshold of average interval, x_i be the i-th record, y_i be $x_i's$ label, and t_i be the average interval of x_i; where $y_i = 0, t_i \geq T$” (Liao et al., 2015, p. 3112). The next step is considered the most important step as it would determine if this methodology stands up to what it claims or fails to provide the results. Here it applies the complete SVD-RM methodology. It uses the element in SVD-RM vector algorithm where the value 0 for the DDoS application layer attack detection shows group access behaviors rather than just single user behavior helping to distinguish based on the pattern of the 0 values. SVD-RM looks into each user request frequency sequence as sparse vector while decomposing it into two parts i.e. order vector and value vector (Liao et al., 2015). The SVD-RM forms multiple clusters by using the order vector rhythm matching; since the DDoS application layer attacks would have program generated accesses and attackers' behavior similar to each other; whereas the normal user will have diverse behavior patterns helping in distinguishing between attackers and normal users. Here is the figure showing Step 2 of the SVD-RM methodology:

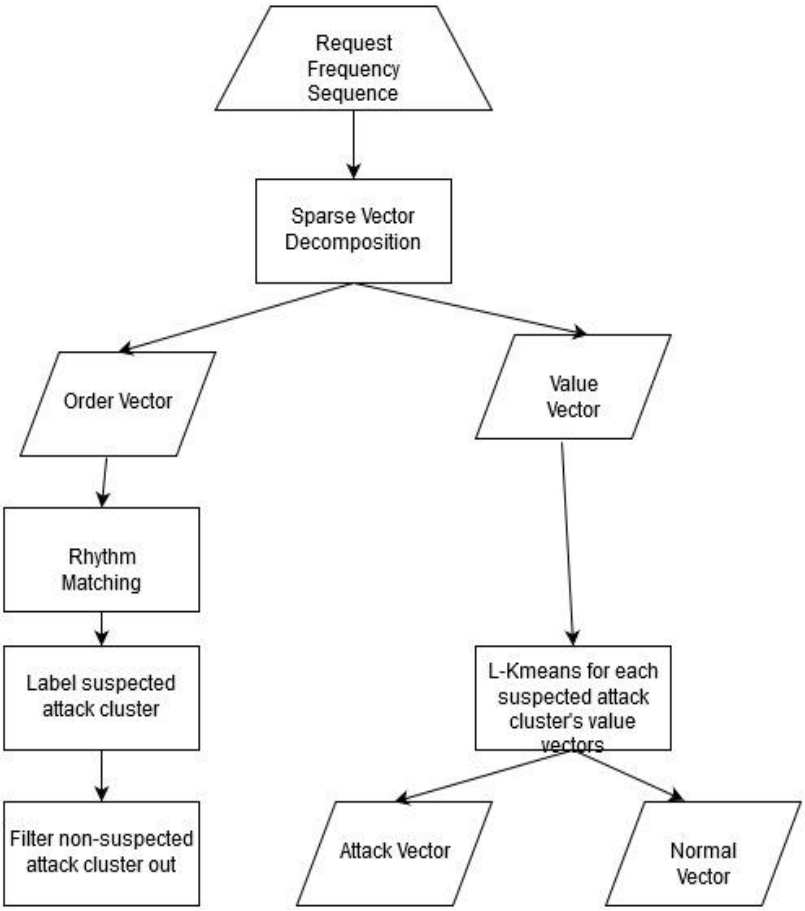

Figure 9. Second Part of SVD-RM (Flowchart) (Liao et al., 2015).

The result of the SVD-RM using these two steps as methodology is that the first three types of attacks are detected quite well; whereas the fourth type of attack detection is not efficient as the attackers' behavior imitates the normal users' behavior. Here are the four types of attacks mentioned with the rate of frequency:

The above-mentioned attacks can be classified as Type 1, Type 2, Type 3, and Type 4 attacks respective to their order in the table.

Figure 10 shows the accuracy of SVD-RM algorithm methodology between these four types of attacks:

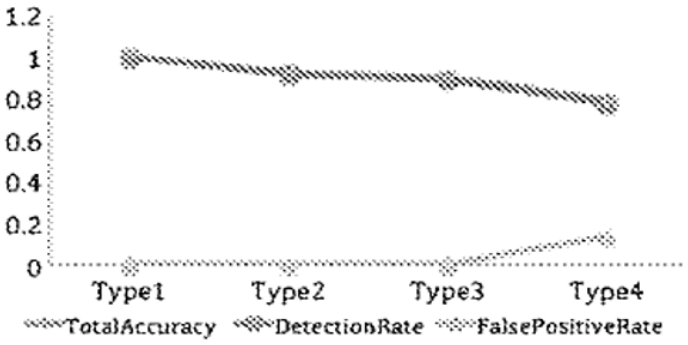

Figure 10. Classification Result of the Four Types of Application Layer DDOS Attacks (Liao et al., 2015).

As per the tests mentioned in the journal article, this methodology seems to work out quite well and its performance compared to the other methodologies are shown to have worked out quite good as well in terms of application layer DDoS attacks. Here is the table comparing the current methodology with the methodologies created by other researchers:

Table 4

Comparison with Previous Works (Low, Medium, and High) (Liao et al., 2015)

	Accuracy	Scalability	Complexity
HSMM	High	Low	High
Botz-4-Sale	Unknown	Medium	Medium
Traffic Analysis	Medium	High	Low
Wavelet	Medium	Low	Medium
Covariance	Medium	Low	Medium
Backbone	High	High	Medium
SVD-RM	High	High	Low

Research Article 4. Another article focuses on the Botnets to analyze the effectiveness of the various Botnet agent allocation strategies and estimating victim resistance probability against DDoS attacks (Ramanauskaite, 2015). Figure 11 shows an example of active botnets in Lithuania over the past six years representing analysis of everyday scenario:

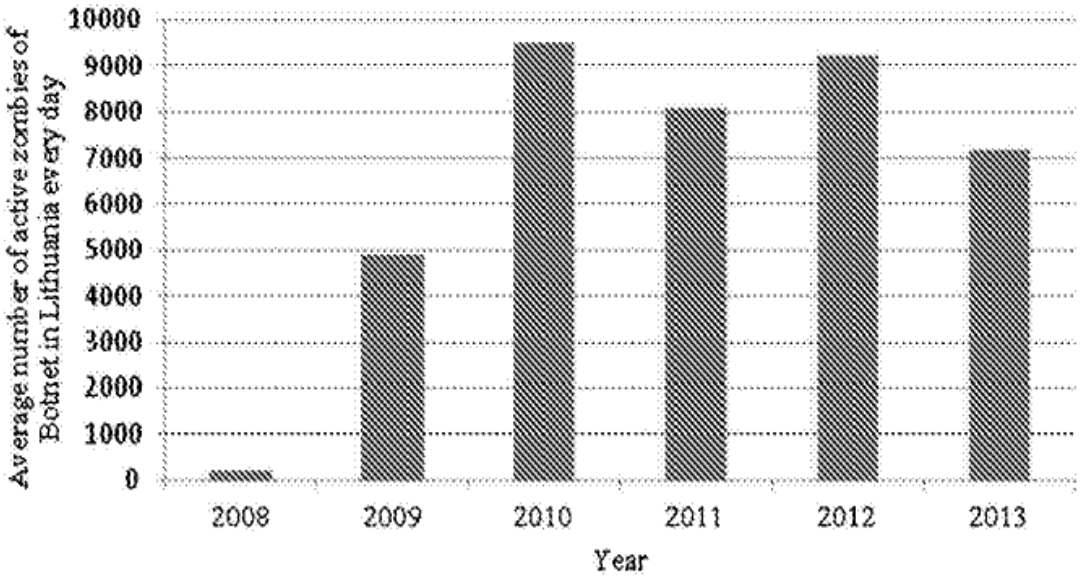

Figure 11. Changes in the Number of Unique Botnets in Lithuania (Ramanauskaite, 2015).

The article proposes to use a DDoS attack simulation as a way of finding the Botnet agent attack patterns to prevent the attacks by increasing the resources and filtering attack traffic. Their methodology is focused mainly on thinking like an attacker and coming up with different strategies of DDoS attack patterns using different botnet allocation methods. They found out that changing the agent allocation speed changes the attack power and therefore the attacks can never be conducted in a stable form. The major issue with this methodology is that it is easy to block one zombie computer but there are other million other infected zombies (botnets) which all have different viruses and trojans and are difficult to detect and prevent them to be a part of the attack.

Moreover, the IP addresses of these zombies could be spoofed making it difficult to detect them and stop or block them.

CAPTCHA is considered to be a good security measure by some of (Umarani & Sharmila, 2014)the research articles like (Umarani & Sharmila, 2014; Vadlamani, 2013). It seems to be a good solution for DDOS attacks as most of them are executed using botnets (automated computer bots) but it is not a security measure in preventing a slow rate application layer attack. The slow rate attack does not need a botnet or bot and can be executed by a malicious user whose sole purpose is to infect the backend server's database. Therefore, CAPTCHA can be considered a good solution for mitigating HTTP GET flood attack but not quite for the HTTP POST attack.)

Research Article 5. This paper uses the http traces from 1998 world cup website for detecting the Layer 7 DDOS attacks. Three methodologies have been used namely KNN, PCA-Naïve Bayes, and Naïve Bayes. The data set belongs to the timeline of April 30, 1998 to July 26, 1998 which consists of total 1,352,804,107 http requests obtained from 33 various world cup http servers from four locations (Paris-France, Plano-Texas, Herndon-Virginia, and Santa Clara-California). The parameters used to differentiate between normal and attack traffic is converted into binary to save space and processing time. The parameters are timestamp (Epoch time-converted to GMT), ClientID, object-ID (unique identifier for the URL), size (number of bytes), method (found in client request), status (the first 2 bits are the http version and the rest 6 bits are the response status code), type (type of file that was requested), server (the specificity of the server that handled the request).

They also used PCA Naïve Bayes which focuses on reducing the number of variables that are uncorrelated. By using these variables, the average vector is computed through the formula $\bar{\mu} = \frac{1}{T}\sum_{t=1}^{T} \bar{a}_t$ where T is the total number of variables in the data set and $\bar{a}_t$ is the variable t. Later this PCA methodology is used to minimize the data with multidimensions through the computation of characteristic value (eigenvalue) associated with characteristic vector of linear transformation (non-zero vector or eigenvector) which are then used in singular value decomposition (SVD). In the geometrical terms the eigenvector that belongs to a non-zero eigenvalue goes in the direction at which it was stretched through the transformation, and this eigenvalue is a factor used for the stretch.

After conducting the test using Naïve Bayes, it was found that the detection rate of Naïve Bayes excelled by 2.56% and 1.7% in comparison to KNN and PCA-Naïve Bayes. For the PCA-Naïve Bayes, the detection rate only improved by 0.9% and its False positive rate by 4.11%.

Research Article 6. OC-SVM is a methodology where only one class is used to train a specific set of data. It will be fed only good/normal traffic as a training data set for the algorithm to recognize normal users. Through this, if it detects any behavior that represents even a small chance of abnormal/attack traffic then it would place it into another class (the attack traffic class). It is an anomaly-based detection technique in which any traffic that deviates from the normal traffic (the training data fed to OC-SVM) gets labeled as anomalous traffic. It is used to essentially map the data points from input space to feature space through the method of non-linear kernel function. The training data would end up in one class, once mapping the data in feature space is completed. Later, a boundary is created in feature space that separated normal and anomalous traffic (She, Wen, Lin, & Zheng, 2017).

The next step involved defining certain techniques/methodologies that are part of the feature/parameter selection process to identify normal vs attack traffic.

First, resource popularity is introduced which is considered to use Zipf-like distribution. The popularity of resource (i) is identified as $POP_i = \frac{AC}{AC_{all}}$ where AC_i is the number of resource (i) accessed in a certain time period, and AC_{all} is the number of all the resources accessed in that time. As per this, A would be more popular than B, if A's frequency is higher than B.

Second is transition probability, which is the probability of going/transitioning from one state to another. In this case it is the probability of transitioning from one resource to another; which is represented by $P_{i,j} = \frac{Trans_{i \to j}}{Trans_{i \to all}}$, where the numerator is the transition time from i to j and the denominator is the transition time from 'i' to 'all other resources'.

The third one is history transition matrix, which is transitioning from one resource to any other using matrix form, represented as (She & Wen, 2017):

$$M_{trans} = \begin{bmatrix} t_{11} \dots & t_{1i} \dots & t_{1n} \\ t_{i1} \dots & t_{ii} \dots & t_{in} \\ t_{n1} \dots & t_{ni} \dots & t_{nn} \end{bmatrix}$$

The parameters used for analysis are $N_{session}$ which is the total number of requests in a session (which is highly important in detecting application layer DDOS attacks); $POP_{session}$ is the average popularity of all requests in a session and since $POP_{session}$ of a session generated by random request attack is low in comparison to normal user session, this is a good parameter for it (some sophisticated Layer 7 DDOS uses random session requests as an attack methodology which can be detected using this parameter); $P_{session}$ is the average transition probability of all adjacent requests in a session and has the same popularity level as $POP_{session}$; $Size_{session}$ is the total size of all requests in a session where an attacker requests a large packet (slow-rate attack)

to consume big chunk of resources; $D_{session}$ is the duration of each session which identifies a stealthy attacker based on the fact that normal users don't stay on a webpage for too long; $ReplyCode_n$ is the http codes (200, 400, 500) generated at the web server based on the frequency of the generated code to identify it as an attack; $dynamic$ parameter looks into the webpages that involve working with database querying and/or injection/insertion (sophisticated script computations) attacks and therefore it is a highly important feature in the detection of stealthy layer-7 DDOS attacks.

These features/parameters are used in the OC-SVM algorithm for training it with the normal-user traffic dataset. Later, this is used to create boundary between normal and anomalous traffic through an optimization model represented as $min\frac{1}{2}\|w\|^2 + \frac{1}{nv}\sum_{i=1}^{n}\xi_i - \rho$, where n is the number of data points, ξ_i is the non-negative slack variable of i, v is the regularization parameter controlling the fraction of outliers, w and ρ define the decision boundary. Based on the mathematical computation of the algorithm's equations' steps for every new session; where a result of one for a session is considered normal traffic and the result of -1 is considered anomalous traffic. Any user session that is identified as anomalous gets its IP added to the blacklist. During the detection phase many HTTP requests are received which are added to the respective user sessions.

The datasets for this study were the weblogs of the website of "Sun Yat-sen University" collected on October 4, 2013; consisting of 20978 IPs, 1,281,876 requests, and 2480 total resources with timestamps (of 1 second precision). The beginning and ending of the requests are between 0seconds-86382seconds. The attack was randomized between 40000seconds-55000seconds, mixing it with the normal traffic. Later, entropy of the dataset is computed every

10 seconds and based on the graphical representation shown below, the entropy is normal before 40000sec and spikes till 55000sec (She et al., 2017):

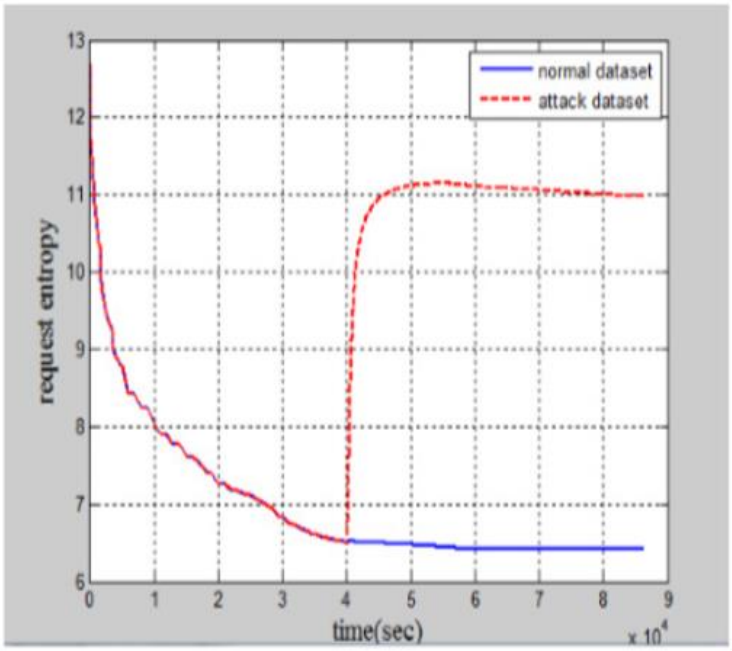

Figure 12. Normal and Attack Request Entropy (She et al., 2017).

Research Article 7. This research article uses almost the same methodology as the one above i.e. SVM supervised learning algorithm. The only difference is that this one is using ESVM with string kernels. This methodology is effective considering that it uses SVM kernel functions such as linear function, polynomial function, sigmoid function, radial basis function, and string kernels with weighted features. The issue with kernel functions is that it treats all inputs equally which could lead to high false positive and/or false negative rate; and this problem is solved by assigning weights to these inputs/features. This is considered as an enhanced function which has the capacity to distinguish between normal and attack traffic. String kernel function has string inputs that produces efficient classification result for string parameters.

Parameters that deviate from the normal traffic behavior is assigned high priority and the ones mimicking close to normal traffic behavior is assigned low priority.

The parameters used to identify variations in the traffic patterns are HTTP request rate which is the number of HTTP requests from client to server in specific duration of time, session rate which is the number of established sessions from client to server within a certain time frame, and time spent on a webpage. For application layer DDOS detection's training data set, three classifications are created normal, HTTP flood, and session flood. Every training instance is assigned a weight (Ramamoorthi, Subbulakshi, & Subbulakshmi, 2011).

In respect to detection and reducing the storage space, only attributes that are occurring most frequently are taken into consideration with their respective ratios. Namely two approaches are used, static threshold and adaptive threshold. In static threshold, the values included are those who frequency is higher than a preset threshold ratio (e.g., x percentage). Those attributes that are not present in the iceberg profiles use upper bound (x percent) as its ratio. The adaptive threshold uses most frequently occurring attribute values constituting a preset coverage of the traffic. The corresponding y percent cutoff threshold for a given coverage serves as adaptive threshold and it is used as a default ratio for the absent attributes. The nominal profile for such iceberg-style profiles have the capacity to be stored to a manageable size.

In the attack scenario, the http requests are higher based on the duration of the visit to a website page. This is classified to the best of its ability using ESVM with string kernels. In an event of an attack, number of users vs. number of requests is used to identify normal and anomalous behavior (HTTP flooding attack). In a normal traffic, the number of users and the requests increase together in a gradual fashion. In an attack scenario, the number of users is not

that high, whereas the number of requests show a drastic increase. In a session flood attack, number of users vs. number of sessions is used; where increase in sessions when the number of users is low in a given time-duration is an attack and vice versa (Ramamoorthi et al., 2011).

When this methodology is compared with existing approaches, ESVM shows to be more efficient than others as shows in this graphical representation (Ramamoorthi et al., 2011).

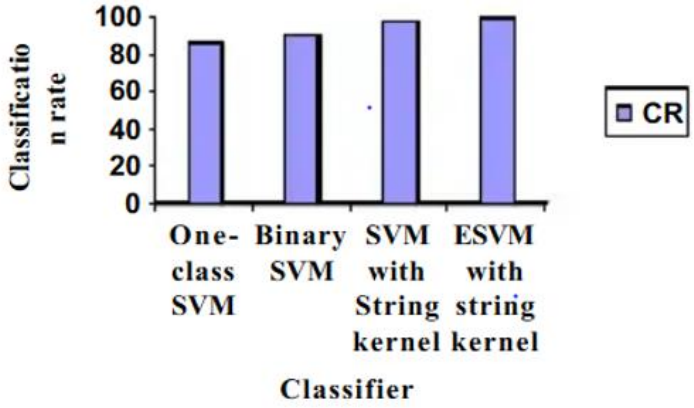

Figure 13. ESVM vs. Existing Mechanism (Ramamoorthi et al., 2011).

Summary

This part of the paper dealt with the literature related to the methodologies used for the detection of DDOS attacks at Transport (Layer 4), Network (Layer 3), and Application Layer (Layer 7). It explained the feature selection process and the modified algorithms/methodologies that were used to identify and analyze the network traffic at the respective layers. Using tables and graphs when necessary to explain the functionalities or the process of the feature selection and data-set analysis. Moreover, the environment in which each of these experiments were researched and tested to get the desired results, were also discussed.

Chapter III: Methodology

Introduction

Supervised machine learning has perceptrons (inputs) where each input has its own weight and based on the dataset, we are trying to distinguish to get the correct results. For example: An input x has the output in single binary value f(x), b is the bias that gives this model a little flexibility (learn itself) in an event we don't have enough inputs, and w is the weight. The equation would be f(x) =1 if w*x+b>0 or else f(x)=0. The summation of all the perceptrons that would lead to a result uses the equation $\sum_{i=1}^{m}(w_i x_i)$ where m is the number of inputs. The output is later graphed using various functions and there are three top used functions sigmoid $g(x) = \frac{1}{1+e^{-x}}$ and $g'(x) = g(x)(1 - g(x))$, hyperbolic tangent $g(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ and $g'^{(x)} = 1 - g(x)^2$, and rectified linear unit (ReLU) $g(x) = \max(o, x)$ and $g'(x) = 1\ if\ x > 0\ or\ else\ g'(x) = 0$.

Based on this, if we wanted to identify a human face, then it would have multiple inputs where each input would identify a small part of the face (using pixel data) like an eye, ear, nose, mouth and various angles, curvatures, or lines. All these inputs would be put together to finally create an output that would identify the pixel data into a full face. Same logic would be applied when trying to detect anomalies within network traffic where each input (with its weight (e.g., w_1)) defined would look into IP addresses, IP address locations, IP address frequency, time-interval between connections, TCP header, IP header, and other relevant criteria (based on the protocol used for the attack and the OSI layer on which it was carried out) to separate normal from attacker traffic. Later these inputs are sent to and processed by the hidden layers with its own weight (e.g., w_2) assigned to it based on the various behavioral patterns observed by the algorithm (Note: hidden layer is not observable or directly manipulated by the user). Both the

inputs and the hidden layers are computed together to formulate an output (result) and this process is repeated with different inputs and in various scenarios to train the algorithm so that it can create desired/factual (probabilistically accurate/correct) results. It is important to keep in mind that we do not over-train the data which leaves no space for it to learn/adapt and we also don't want it to be under-trained (under-fitting) which means that it does not have enough data (inputs) to get the correct outcome/results.

SVM. Support Vector machine is a supervised learning algorithm with an output on a plane divided by a line (best decision boundary (maximum margin hyper-plane)) that determines the separation of two labeled classes (graphed as scatter plot) obtained through trained data. Margin/Hyper-plane/Line needs to be equidistant from both the classes to ensure that maxim area is provided for left and the right classes to ensure that data is not misrepresented (incorrect). SVM would work perfectly for detecting a slow Application layer attack as it is efficient/capable in handling small amount of data, robust to outliers, and can swiftly manage complex data transformations. It is highly important that in an event where application layer attacks are too complex (acting as normal traffic) which would make the data plot overlap as the "attacker traffic" would get mixed with "normal traffic"; we need to tune regularization, kernel, and gamma parameters (these are SVM classifiers) to achieve accurate results by forming non-linear classification hyper-plane(line). Regularization ensures that misclassification of network attack traffic (false positives and false negatives) is avoided by creating a flexible hyper-plane to separate classes accurately. Kernel (kernel like linear, poly, rbf, sigmoid, pre-computed, or callable to be used in an algorithm) is involved in predicting new inputs (pattern analysis) using the equation $f(x) = B(0) + \text{sum}(a_i * (x, x_i))$ where x=input, B(0)/ai=coefficients estimated by

training data (algorithm in learning stage), and xi=support vector. Gamma parameter analyzes the prediction and determines how far we are from the hyper-plane based on the training data sets; where high means that “only nearby points must be considered” and low means “points far away are also considered”. You can implement SVM using python’s scikit-learn library by importing the library, creating objects, and then fitting the model to its intended purpose and predicting as needed. One of the disadvantages of using SVM is the fact that it is unable to make probability estimates and the way of doing it is quite expensive.

ANN. Artificial neural networks are more of something based on the learning algorithm methodology where the inputs are activated using hidden layer nodes’ activation functions and these hidden layers are linked to the output which activates the output function. The learning happens at the hidden layer where the inputs are evaluated by the algorithm over time to come up with an accurate result. ANN’s have great memorization and are highly over-fitting making them perfect for cracking fixed rule systems like Chess, Go, Checkers, and other similar strategy games and/or applications. It is advisable to use supervised or semi-supervised learning since there are anomalies within network traffic that are not attack traffic which could only be separated with human intervention. Due to its over-fitting nature it is not considered a good choice for predictive analysis where the behavioral patterns are too flexible (unpredictable). It would be a great fit for detecting network and transport layer flood DDoS attacks as ANN can handle large volume and is efficient in recognizing learned pattern.

K-NN. K Nearest Neighbors is one of the simplest algorithms and quite straightforward. It investigates the entire dataset and chooses a number “K” of its neighbors. It then takes the K nearest neighbors of the new data point (based on Euclidean/Manhattan/Minkowski/Hamming

distance or any other distance based on the data we are studying). The next step is counting the number of data points in each category. The last step is assigning a new data point to the category which is the closest in similarity to that data. In terms of identifying the network traffic if we see an anomaly that seems to be like attack traffic, but it is not, then we will use our specific rules to assign that anomaly to normal traffic.

Euclidean distance works best when all the input variables are similar in type for example high bandwidth ICMP ping attacks which can be separated based on the number of ping requests and frequency of similar IP addresses. Manhattan distance would be effective when dealing with different input variables, for example, in slow-rate application layer DDOS attacks.

Euclidean distance: $Distance\ between\ A_1\ and\ A_2\ = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Manhattan distance:

K-Means. This algorithm is used for unlabeled data i.e. unsupervised learning. It is categorized based on groups (clusters) where each group has a centroid. It uses Euclidean distance between data points and the centroid to create new labeled data. The methodology of k-means relies on selecting the number of clusters and then assigning data points to each cluster. After assigning data points to a Cluster X, one needs to find the centroid and then compute the distance with other centroids. After carefully recomputing the distance between two centroids, we need to ensure that the data point closest to the centroid of a Cluster X, gets assigned to Cluster X and not Cluster Y. Each data point belonging to a cluster needs to be closest to the centroid of that cluster. One of the drawbacks of k-means clustering is the varying results after recomputing the distance each time. For this reason, the centroids assigned to a sample and the samples that have a centroid assigned to them needs to be recomputed until there is no change

seen in the clusters in respect to the centroids. Another drawback is the fact that if the number of clusters are not created correctly, then it can completely change the results of the test. K-means won't work well for detecting application layer DDOS attack in real time due to the rigorous re-evaluation of clusters' centroids and samples which increases the time it takes to run the algorithm. The only problem with the clustering methodology of this algorithm is that it does not have the capacity to accurately distinguish between network packet attributes since it relies on calculating the mean of the cluster's data points. It can work well only for detecting a specific type of DDOS attack that does not have the complexity of the parameters involved for detection but rather quite simple (like ICMP (ping) DDOS attack).

- The first step involves identifying the cluster centroids like C0, C1, C2 from the available data points.
- The second step involves choosing the closest data points to the cluster centroid (based on Euclidean distance) to define each cluster.
- The third step is to calculate the distance between data points and the cluster centroid e.g., x0, x1, x2.
- The fourth step is to recalculate the cluster centroid by adding all the points assigned to one cluster and dividing it by the total number of data points belonging to all clusters (C0, C1, C2).
- The fifth step is to repeat step third and fourth till we do not see any significant changes in the distance between the data points and the cluster centroid of each cluster.

The equation to calculate and solve it using mathematics (SSE=Sum of Squared Errors):

Note: This is the Euclidean distance metric to define the k means algorithm that involves minimizing "the within-cluster SSE, also called cluster inertia" (Raschka & Mirjalili, K-means clustering, 2017)

$$\text{SSE}=\sum_{i=1}^{m}\sum_{k=1}^{K} w_{ik}\left\|x^{i}-\mu_{k}\right\|^{2}$$

Where $w_{ik}=1$ if x^{i} data point is in cluster k or else $w_{ik}=0$. Moreover, μ_{k} is the centroid of the cluster, i is the number of m records, k is the number of K clusters, and w is the weight defined for each cluster.

Fuzzy C Means. Fuzzy C means is another clustering methodology that can be used for multidimensional data. This means that every data point has the probability to belong to any k clusters and not just to one cluster like in K-means algorithm. So, it is about assigning data points that have the possibility of belonging to multiple clusters by using probability and not the distance. It deals with centroids that are assigned weights based on the probability of it belonging to a cluster. This is the reason why this algorithm is also called the soft-k-means algorithm. The algorithm was first concocted during 1970s to improve the k-means methodology. Instead of having either 1 or 0 (like on or off) to identify a data point belonging to a cluster or not; it will be in the range of 0 to 1 to represent the probability of the data point belonging to a cluster or two/more clusters.

- The first step would be to choose the number of k (cluster) centroids and assign clusters based on each data point.
- The second step involves calculating the cluster centroids μ_{j} , $j\epsilon\{1,\ldots k\}$
- The third step is recalculating the probability of the data points belonging to those clusters

- The fourth step is about repeating second and third step until there is no significant changes in the probabilistic values of those data points assigned to k cluster (Raschka & Mirjalili, 2017).

The mathematical notation to represent this algorithm is quite similar to k-means:

$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} \mu_{ij}^{m} \|x_i - c_i\|^2, 1 \leq m < \infty$

Naïve Bayesian. This algorithm is based on Bayes theorem. Bayes theorem uses conditional probability to justify the occurrence of an event. Naïve here means that each feature/parameter is independent of each other and has no effect on others. It is easy, scalable, can work fast in real time, and great for applications in the real world. This algorithm, if trained using supervised learning, can produce efficient results, by feeding parameters using the maximum likelihood estimation (MLE).

The mathematical representation of this algorithm is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where: P(A) and P(B) are the probabilities of the individual events A and B simultaneously

A and B are the events

P(A|B) is the probability of the event A to occur if the event B is true.

P(B|A) is the probability of the event B to occur if the event A is true.

Due to its scalability, it can work well with complex DDOS attacks like Layer 7 (application layer) attacks. It is great as a text classification methodology that can dive deep into the adapting behavior of the attack vectors (e.g. cache bypass http flood and WordPress XMLRPC floods) of an application layer DDOS attack; due to its ability to efficiently process

new data. Another methodology related to this is called Multinomial Naïve Bayes (MNB) which works great specifically when the number of parameters/features representing an attack are very few.

Design of the Study

The study will mainly focus on quantitative analysis with some emphasis on the qualitative comparative study. Quantitative analysis will focus on comparing the performance metrics such as false positive and negative rates, computer resource consumption, response time, and high/low rate traffic detection accuracy.

Quantitative analysis would help in determining the efficiency of the methodologies in the training environment.

Data Collection

Data was collected by the researchers of each research articles either from DARPA, CAIDA datasets, university sources, normal traffic datasets from the university websites, custom created datasets in their individual test environments, and other means like survey methods. It will focus on the parameters involved to detect and analyze the likelihood of an event to be either false positive, false negative, true positive, or true negative. Each of the data sets are either used entirely or a technique of 10-folds is applied to create sub-samples, for the purpose of reducing the storage space and processing power. For network layer attack detection, the data was obtained from the routers and switches that included ICMP packets; both with normal and attack traffic (which was randomly generated in between a time-duration of the normal traffic). For application layer attack detection and for learning phase of the algorithms, the data was collected from the webservers that had HTTP traffic.

Tools and Techniques Used

The tools and techniques solely belong to the researchers that were part of the research articles that are being used for this paper (studying the varying effects of the algorithm in determining and identifying attack traffic). Varied techniques were implemented by the researchers based on the attack vectors determined after using the available knowledge sources determining the features involved in identifying and distinguishing between normal and attack traffic.

The researchers mostly used Linux systems and/or servers to conduct their experiments using the data from the above-mentioned list.

Summary

This section of the paper discussed about the methodologies used in detection of these attacks, the techniques related to the detection, and the datasets used for both machine learning and

Chapter IV: Data Presentation and Analysis

Analysis Techniques Used

The detection algorithms were analyzed using mainly three quantitative parameters i.e. Detection Rate which can also be called as the accuracy rate of the algorithm's detection capability, false positive rate meaning that an event was recognized as malicious but was instead normal traffic, and false negative rate meaning that an event was marked as normal traffic but was actually attacker traffic.

There were many parameters used to calculate the effectiveness of each of these algorithms and it may have some variation in the efficiency detected based on those parameters' values and the parameters used in this paper for the quantitative analysis. The variation however is not so far apart as compared to the results (analyzed data) discussed in this chapter. So, a reader can be quite comfortable in knowing that the algorithms efficiency in detecting these distinct attacks based on the datasets used, is relevant to that particular attack scenario. Keeping in mind that attacks conducted using one protocol (e.g., UDP) will vary in comparison to an attack conducted using another (e.g., TCP); the comparison of the algorithms efficiency in detection is mainly done for either one protocol or the other. There are some algorithms that were used to detect both TCP and UDP together, for which, each of those algorithms have used all the parameters found in both these protocols that can be shown as a proof of malicious traffic.

Moreover, I have segmented the analysis into three sections i.e. Transport Layer Detection Analysis, Network Layer Detection Analysis, and Application Layer Detection Analysis. This will keep things clearer and simpler to understand in respect to the efficiency of the methodologies. It is said in the machine learning industry that any algorithm that shows to

have 100% detection rate could have its result skewed due to not so rigorous testing (the algorithms may not have been tested and trained till the point of failure). If the algorithms were not tested till the point of failure, then it may miss out on detecting some attack behavior patterns. This does not mean that the researchers mentioned below did not test it or have cheated in showing 100% accuracy rate. It could be a coincidence or a mere fact that it worked well in detecting an attack related to a certain protocol-attack (might be a result of an overfitted detection methodology/algorithm). If it is an overfitted algorithm, then the likelihood of it performing to the same standards as before may be less.

Transport Layer Attack Detection-Quantitative Analysis

As you can see below, the first five algorithms show to have either 100% accuracy or close to it.

The first algorithm was tested for both high-rate and low-rate attack traffic where the accuracy for CAIDA datasets for high-rate is 97.35% with an FPR of 0.0087%, and for the DARPA datasets for high-rate is 98.6% with an FPR of 0.12%. Based on the table in the research by Hoque et al. (2016), the attack accuracy changed from 97.35% to 100%, for which there was no explanation in the article itself. This creates a rather confusing scenario or a possibility that it was either rounded to the nearest hundred, the new test showed 100%, or the result skewed by the testing algorithm itself.

The dataset involved in the second research article was created by using TShark to extract fields from the packets captured in wireshark. The total instances used were 7126 (it had both normal and attack traffic). The number of instances is low since it was a DOS attack, but the protocol used to conduct the attack were the same as the first one. Both the first and second

methodology was tested against high-rate attacks. There is high likelihood that the second algorithm was not tested to the point of failure, since it there is no explanation of their rigorous testing in different scenarios.

The third method was tested in various scenarios using DARPA, SETS, and Auckland-I datasets and can be compared well with the first method. This method for DARPA dataset shows accuracy of 100% with varying t (error percent-FPR) for different datasets and 99.95% with an adjusted t value of 0. This researcher did a rigorous testing, which makes its results more reliant compared to other algorithms' results.

For the fourth study, the dataset used had 7,020 instances which is like the second study. Therefore, we can compare the second and fourth one much better. Both the second and fourth methods show 100% accuracy detection rate and making it difficult to know which one fared better. The fourth method did a recall calculation to check if it performs well in positive sensitivity value which looks into the percentage of detecting true positive over both TP+FN and it came out to be 90% (it is good). But, the second method did not do a sensitivity value calculation, so there is no way of knowing if one was better than another.

The fifth, sixth, and seventh studies were all done by the same researcher. If we compare them, Naïve Bayes shows to be the most efficient in detection accuracy and in FNR.

Table 5

Quantitative Analysis of the Various Methodologies for Transport Layer Attacks

Numbers	Methods	Detection Rate	False Positive Rate	False Negative Rate
1	FFSc using CAIDA dataset (Hoque et al., 2016)	100%	0	0
2	SVM (TCP AND UDP attack) (Masetic, 2017)	100%	0	0
3	SVM using DARPA dataset (UDP attack) (Vijayasarathy, 2012)	99.95%	0	0.05%
4	Naïve Bayes-UDP (Sofi et al., 2017)	100%	0	10%
5	Decision Trees-UDP (Sofi et al., 2017)	100%	0	0
6	MLP-ANN (UDP) (Sofi et al., 2017)	96.3%	0	12%
7	SVM (UDP) (Sofi et al., 2017)	97%	0	11%

Network Layer (ICMP) Attack Detection-Quantitative Analysis

Looking into network layer detection which is all about ICMP Flood attack analysis and detection. The parameters used that identify it to be either normal or anomalous traffic are quite similar across various methodologies used to train and test these algorithms.

The first and second algorithm create their own datasets using various tools and techniques and are quite similar in the number of instances used to train and test the algorithm's effectiveness. As we can see that the first one seems to do well in all three aspects namely detection rate, FPR, and FNR.

The third one seems to be better at detection than the second one and it can be compared well with the second since they are both using ANN as the detection algorithm. The datasets may vary a little in terms that the third method is about detecting ICMPv6 flood attacks which may differ in the attack behavior by a small margin. But even then, we can say that the third one did a little better even for detecting a little more complex protocol (ICMPv6 compared to ICMPv4) attack.

The fourth and fifth both use DARPA and CAIDA datasets making it easier to compare. Based on the results we can say that Semi-Supervised K-means performed better than Data-mining centroid based method.

Table 6

Quantitative Analysis of the Various Detection Methodologies for Network Layer Attacks

Numbers	Method	Detection Rate	False Positive Rate	False Negative Rate
1	Fuzzy C means (Lysenko & Savenko, n.d.)	100%	0	0
2	ANN (Devi, 2018)	97%	3.9%	0
3	ANN-Back propagation (Anbar, 2015)	98.3%	0	0
4	Semi-supervised K-means (Wang, 2019)	99.59%	1.38%	0
5	Data Mining-Centroid based method (Manna, 2014)	97.67%	3.23%	0

Application Layer Attack Detection-Quantitative Analysis

The first method got its dataset from traffic archive of Changzhou university's web server. The second, fourth, fifth, and sixth method used CAIDA datasets since it was all done by same researchers but testing different algorithms/methodologies. If we compare them all then the PCA-Naïve Bayes has done just a little better compared to Naïve Bayes.

The third method used requests generated on 1998 World cup website (April 30-July 26). Compared to all other methodologies, it seems to come in the second last place.

Comparing the last seventh, eighth, ninth, and tenth method's results we can see that "ESVM with string kernels" algorithm have fetched the best detection rate efficiency.

Table 7

Quantitative Analysis of the Various Detection Methodologies for Application Layer Attacks

Numbers	Method	Detection Rate	False Positive Rate	False Negative Rate
1	Time Series analysis (SVM-HRPI)-HTTP GET flood (Wang, 2013)	98.06%	1.47%	1.02%
2	Machine Learning-Bat Algorithm-HTTP Flood (Sreeram & Venkata, 2019).	94.8%	1.8%	3.1%
3	KNN-HTTP GET flood (Umarani & Sharmila, 2014).	93.03%	1.74%	0
4	KNN-PCA-HTTP GET FLOOD (Sreeram & Venkata, 2019).	94.25%	1.86%	3.8%
5	Naïve Bayes with PCA-HTTP GET FLOOD (Sreeram & Venkata, 2019).	95.95%	2.01%	1.97%
6	Naïve Bayes-HTTP Flood (Sreeram & Venkata, 2019).	95.47%	1.9%	2.8%
7	E-SVM (String Kernel) HTTP Flood (Ramamoorthi et al., 2011).	99.32%	n/a	n/a
8	E-SVM (Linear) Ramamoorthi et al., 2011).	93%	n/a	n/a
9	E-SVM (Polynomial) (Ramamoorthi et al., 2011).	96.45%	n/a	n/a
10	E-SVM (Radial Basis) (Ramamoorthi et al., 2011). \	97.15%	n/a	n/a

Chapter V: Recommendations and Conclusions

Recommendations

After conducting rigorous research and talking to many industry experts in various conferences and settings, I have come to three main recommendations for future researchers and enthusiasts. First, is to test and train the machine learning algorithm over a long period of time in multiple different business process network settings (in development environments) and conducting multiple different test scenarios, to find different point of failures of the algorithm and improve it on the go.

Second, the machine learning algorithms are still in development and have their own flaws (as discussed with machine learning experts in the ISSR&D 2018 conference). This means that there is a lot of work that need to be done in respect to the algorithms itself to make it effective in different settings. These algorithms prove to be effective under overfitted scenarios where the outcome can be mathematically computed (by a super or quantum computer), but it cannot function at the complexity of a human brain. This reduces the capacity it has in performing highly complex tasks. It does not by any measures mean that it is not useful in complex environment, but just that it may need either longer time of training with various datasets and/or improvement in the overall algorithm itself. Again, to clarify that I am not a machine learning expert and I have read a lot of machine learning books, journals, articles, online lectures and instructions to be able to concoct this research paper and discussed about it through surveys and one-on-one discussions with industry experts in the field of machine learning and security.

Third, the only ways that are known about preventing or stopping DDOS attacks, in the IT security industry is based on the discussions I had with multiple network and network security engineers at the Secure360 conferences in year 2016 and 2017. One way is where the network traffic is stopped by the ISP; which leads to even normal users unable to access the network resources. ISPs can implement network ingress filtering based on BCP-38/RFC-2827 to limit amplification attacks. They can also limit the response rate that is available in the BIND DNS server applications. Another method is conducted by third party DDOS mitigation service providers that use big data centers with high network bandwidth to reroute the DDOS traffic towards them; which are also called scrub centers. So, when a big corporation detects a DDOS attack, it reroutes its traffic to these scrubbing centers which can handle the high-rate flood traffic.

Conclusions

This paper's focus was on detecting attacks on Network, Transport, and Application Layers. Various machine learning and security related books helped in writing this paper and describing the components and techniques involved in the study. It was meant for a new researcher or an enthusiast to read this paper and get a better understanding of the workings of these attacks, detection methodologies, machine learning concepts involved in this study, and using it for further research and new developments in this subject.

References

Alkasassbeh, H. E. (2016). *Detecting distributed denial of service attacks using data mining techniques*. Retrieved from The Science and Information Organization: https://thesai.org/Downloads/Volume7No1/Paper_59-Detecting_Distributed_Denial_of_Service_Attacks_Using_Data_Mining_Techniques.pdf

Anbar, M. E. (2015, November). *An intelligent ICMPv6 DDoS flooding-attack detection framework (V6IIDS) using back-propagation neural network*. Retrieved from Research Gate: https://www.researchgate.net/publication/283339435_An_intelligent_ICMPv6_DDoS_flooding-attack_detection_framework_V6IIDS_using_back-propagation_neural_network

Devi, S. E.. (2018, April). *ICMP-DDoS attack detection using clustering-based neural network techniques*. Retrieved from Springer Link: https://link.springer.com/chapter/10.1007%2F978-981-10-7814-9_16

Eddy, W. M. (2013). Defenses against TCP SYN flooding attacks. *The Internet Protocol*. Retrieved from http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-34/syn-flooding-attacks.html

Gonzalez, H, Stakhanova N., & Ghorbani, A. A. (2013). *The impact of application layer denial of service attacks.* Retrieved from University of New Brunswick: http://www.cs.unb.ca/~natalia/ApplDDoS.pdf

Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2016). FFSc: A novel measure for low-rate and high-rate DDoS attack detection using multivariate data analysis. *Security and Communication Networks*.

Juniper Networks. (2016). *Attack detection and prevention feature guide for security devices: understanding ICMP flood attacks*. Retrieved from Juniper Networks: https://www.juniper.net/documentation/en_US/junos/topics/concept/denial-of-service-network-icmp-flood-attack-understanding.html

Kostadinov, D. (2013). *Layer seven DDOS attacks*. Retrieved from Infosec Institute: http://resources.infosecinstitute.com/layer-seven-ddos-attacks/#gref

Liao, Q., Li, H., Kang, S., & Liu, C. (2015). Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching. *Security and Communication Networks*, 3111-3120.

Lysenko, S., & Savenko, O. (n.d.). *DDoS botnet detection technique based on the use of the semi-supervised fuzzy c-means clustering*. Retrieved from CEUR Workshop Proceedings: http://ceur-ws.org/Vol-2104/paper_251.pdf

Manna, B. (2014). *A proactive DDoS attack detection approach using data mining cluster analysis*. Retrieved from Research Gate: https://www.researchgate.net/publication/282867829_A_Proactive_DDoS_Attack_Detection_Approach_Using_Data_Mining_Cluster_Analysis

Masetic, K. E. (2017). *SYN flood attack detection in cloud computing using support vector machine*. Retrieved from TEM Journal: http://www.temjournal.com/content/64/TemJournalNovember2017_752_759.pdf

Ni, T., Gu, X., Wang, H., & Li, Y. (2013). Real-time detection of application layer DDoS attack using time series analysis. Journal of *Control Science and Engineering*, (5).

Patrikakis, C., Masikos, M., & Zouraraki, O. (2015). Distributed denial of service attacks - The Internet Protocol Journal. *International Journal of Distributed Sensor Networks*.

Peng, T. (2004). *Defending against distributed denial of service attacks.* Retrieved from Semantic Scholar: https://pdfs.semanticscholar.org/c305/db8325dcf9e3bc69861334819521dc6b2c18.pdf

Ramamoorthi, A., Subbulakshi, T., & Subbulakshmi, T. (2011, June). *real time detection and classification of DDoS attacks using enhanced SVM with string kernels*. Retrieved from IEEE: https://www.academia.edu/24576437/Real_Time_Detection_and_Classification_of_DDoS_Attacks_using_Enhanced_SVM_with_String_Kernels

Ramanauskaite, G. (2015). Modelling influence of Botnet features on effectiveness of DDoS attacks. *Security and Communications Network*, 2090-2101.

Raschka, S., & Mirjalili, S. (2017). Hard versus soft clustering. In S. Raschka & S. Mirjalili, *Python Machine Learning* (pp. 354-355). Birmingham, UK: Packt Publishing.

Riadi, M. E. (2017, August). *Neural network-based DDoS detection regarding hidden layer variation*. Retrieved from Journal of Theoretical and Applied Information Technology : http://www.jatit.org/volumes/Vol95No15/27Vol95No15.pdf

Saied, O. R. (2016). Detection of known and unknown DDoS attacks using artificial neural networks. . *Neurocomputing, 172*, 385-393.

She, C., Wen, W., Lin, Z., & Zheng, K. (2017). *Application-layer DDoS detection based on a one-class support vector machine*. Retrieved from IJNSA-International Journal of Network Security: http://aircconline.com/ijnsa/V9N1/9117ijnsa02.pdf

Sieklik, M. B. (2016). Evaluation of TFTP DDoS amplification attack. *Computers and Security, 57*, 67-92.

Sofi, I., Mahajan, A., & Mansotra, V. (2017). *Machine learning techniques used forthe detection and analysisofmodern types of DDoS attacks* . Retrieved from International Research Journal of Engineering and Technology: https://www.irjet.net/archives/V4/i6/IRJET-V4I6200.pdf

Sreeram, I., & Venkata, P. K. (2019). *HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm*. Retrieved from Science Direct: https://www.sciencedirect.com/science/article/pii/S2210832717301655#t0005

Subramani, S. (2011). *Denial of service attacks and mitigation techniques*. Retrieved from SANS: https://www.sans.org/reading-room/whitepapers/detection/denial-service-attacks-mitigation-techniques-real-time-implementation-detailed-analysi-33764

Umarani, S., & Sharmila, D. (2014). *Predicting application layer DDoS attacks using machine learning algorithms*. Retrieved from World Academy of Science: https://panel.waset.org/publications/10000388/pdf

Vadlamani, N. S. (2013). *A survey on detection and defense of application layer DDoS.* Retrieved from University of Nevada: http://digitalscholarship.unlv.edu/cgi/viewcontent.cgi?article=3034&context=thesesdissertations

Vijayasarathy, R. (2012). *A systems approach to network modelling forDDoS attack detection using na`ıve bayesclassifier*. Retrieved from IIT-Department of Computer Science & Engineering: https://www.cse.iitm.ac.in/~ravi/papers/Vijayasarathy_thesis.pdf

Wang, L. E. (2013). *Real-time detection of application-layer DDoS attack using time series analysis*. Retrieved from Hindawi-Journal of Control Science and Engineering: https://www.hindawi.com/journals/jcse/2013/821315/

Wang, L. E. (2019). *Semi-supervised K-means DDoS detection method using hybrid feature selection algorithm*. Retrieved from Research Gate: https://www.researchgate.net/publication/333182800_Semi-supervised_K-means_DDoS_Detection_Method_Using_Hybrid_Feature_Selection_Algorithm