TRAFFIC SEASONALITY AWARE ADAPTIVE THRESHOLD ALGORITHM FOR DETECTION OF FLOODING BASED DENIAL-OF-SERVICE ATTACKS IN IoT NETWORKS

PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Department of Electronics and Communication Engineering Specialization in Communication Systems Engineering by the A P J Abdul Kalam Technological University

by

SREELEKSHMI A N

TKM20ECCS13

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

TKM COLLEGE OF ENGINEERING

KOLLAM 691 005

JULY 2022

TRAFFIC SEASONALITY AWARE ADAPTIVE THRESHOLD ALGORITHM FOR DETECTION OF FLOODING BASED DENIAL-OF-SERVICE ATTACKS IN IoT NETWORKS

PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Department of Electronics and Communication Engineering Specialization in Communication Systems Engineering by the A P J Abdul Kalam Technological University

by

SREELEKSHMI A N

TKM20ECCS13

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

TKM COLLEGE OF ENGINEERING

KOLLAM 691 005

JULY 2022

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

TKM COLLEGE OF ENGINEERING

KOLLAM 691 005

CERTIFICATE

Certified that this project report titled ***"TRAFFIC SEASONALITY AWARE ADAPTIVE THRESHOLD ALGORITHM FOR DETECTION OF FLOODING BASED DENIAL OF SERVICE ATTACKS IN IoT NETWORKS"*** *is a bonafide record of the work done by* ***SREELEKSHMI A. N.*** *(Reg. No. TKM20ECCS13) under my supervision, in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Electronics and Communication Engineering with specialization in Communication Systems by the A P J Abdul Kalam Technological University.*

Guide & Coordinator
Dr. Nishanth N.
Associate Professor
Dept. of ECE, TKMCE

HoD
Prof. Abid Hussain M.
Head of Department
Dept. of ECE, TKMCE

Acknowledgement

At the outset, I find it my obligation to thank the Almighty God for giving me necessary wisdom to complete this project successfully.

I thank Prof. ABID HUSSAIN M., HoD, Department of Electronics and Communication, for his encouragement and support.

I express my profound and sincere gratitude to our project coordinator and guide, Dr. NISHANTH N., Department of Electronics and Communication, for his advice, supervision and patience during the course of project preparation, presentation and providing guidance and critical inputs in the preparation and presentation of my project.

I also express my heartfelt thanks to all my teachers, friends and my parents for providing the much needed support during the course of preparation and presentation of my project.

SREELEKSHMI A. N.

TKM20ECCS13

ABSTRACT

The Internet of Things (IoT) is a new technology that connects and exchanges data with other devices and systems over the internet or other communication networks using physical objects that are integrated with sensors, computing power, software, and other technology. The network nodes in a decentralised infrastructure are typically mobile and have limited resources, such as low memory, low processing power and inadequate battery backup. As a result, they are vulnerable to a variety of Denial-of-Service (DoS) attacks, of which SYN flooding, Route Request (RREQ) flooding and HELLO flooding are examples. It is crucial to identify these attacks to ensure that the network can survive an attack. However, most of the works that are now available employ fixed threshold algorithms which led to large false-positive rates. The majority of attacks vary seasonally and are challenging to detect with current techniques. This project suggests a technique for detecting flooding attacks that also consider network traffic seasonal changes.

Contents

List of Figures

Chapter 1

Introduction

The Internet of Things (IoT) is a fast-developing technology for mobile applications that satisfies ever-tighter requirements including system security, ultra-low latency and low battery consumption and data reliability for cloud infrastructure and underlying wireless sensor networks (WSNs). Services at the network's edge that are close to the user are required to meet these stringent requirements. Thus, Mobile Edge Computing (MEC) is a distributed open platform that combines networking, computation, storage and applications with key capabilities at the network's edge close to the source or data source, developed. MEC offers cutting-edge intelligent services.

Because of its non-interference, broader breadth and scalability qualities, IoT is preferred over all other traditional networks. IoT, which consists of millions of connected devices interacting with one another to facilitate easier and more efficient human processes, has grown to be the largest network in the modern world. In the entire IoT working span, many operations are performed at various levels to accomplish the desired goal of any smart application. IoT systems link the virtual and physical worlds. Fig. 1.1 displays a typical illustration of IoT architecture.

To fully comprehend how the Internet of Things functions, numerous reference models or frameworks have been put forth in the literature [1]. Fig. 1.2 illustrates a fundamental idea with three layers that stand for three distinct functionalities. Data is gathered using a range of sensing tools in the first layer including sensors, Radio-Frequency IDentification (RFID) readers, smart controllers and so forth. The data gathered here must be in a uniform format to interact with the various network

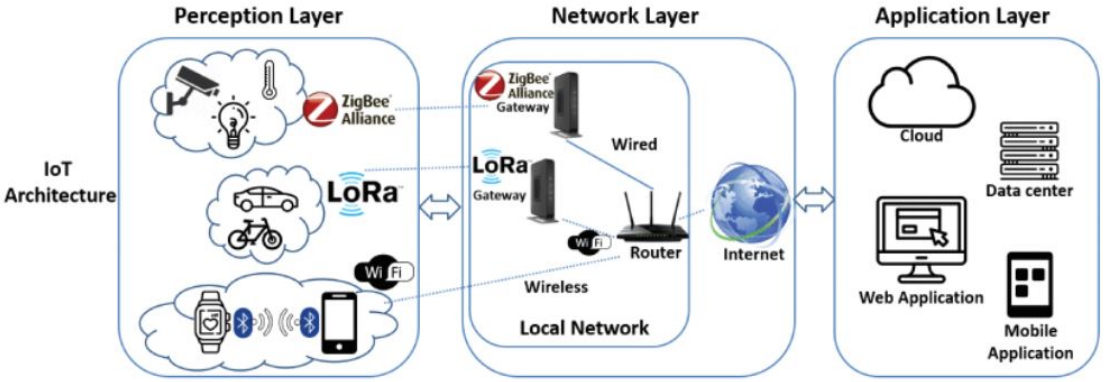

Figure 1.1: IoT Architecture [2]

protocols. This layer is referred to as a perception layer, also known as an edge device layer. The network layer, which is the second layer, is in charge of facilitating communication between the program and the edge devices utilized for data collection. The gathered data is transported through wireless technologies such as Bluetooth, WiFi, Zigbee and others. The Application layer is the final and outermost layer. IoT is being used as a platform for a variety of smart applications including smart grids, homes and cities.

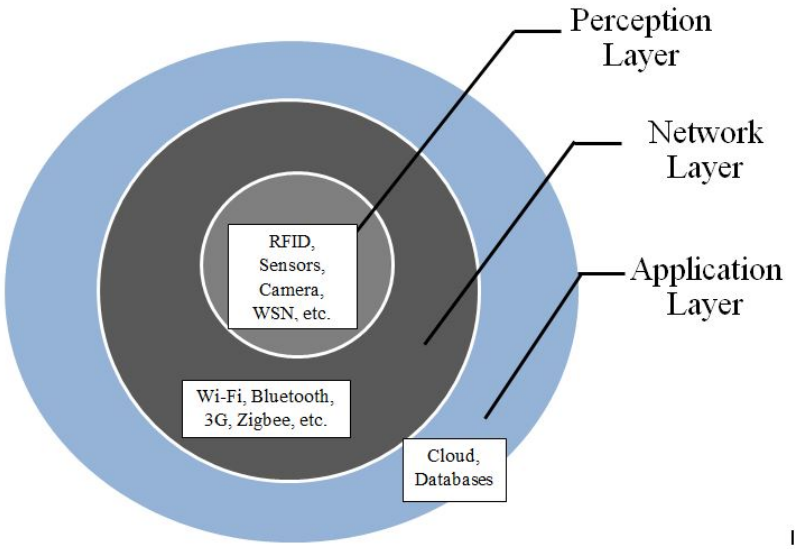

Figure 1.2: Layers of IoT

Although it has a relatively straightforward description, when it comes to problems about security and privacy, it becomes considerably more complicated and substantial.

It becomes significantly more vulnerable to different security concerns because of the unreliable network protocols employed and the lack of human intervention. Limited power and memory resources, which make it intolerant of security attacks on today's high-tech gadgets are another important factor.

Denial of service attacks (DoS), particularly flooding attacks, is susceptible to network edges. DoS attacks prohibit the victim node from going into sleep mode, which might cause the node's lifetime to degrade exponentially as a result of the attack. The identification and prevention of such attacks are crucial because they can occur anywhere, at any time, and with varied strength, especially in the networking environment of the Internet of Things. DoS attackers are easily discovered by analyzing the traffic volume or intensity flowing through a certain link. The multiple layers of the TCP/IP protocol stack can be used to launch flooding attacks. SYN flooding is a common and effective DoS attack that uses the transport layer. Hello flooding and Route Request (RREQ) flooding are examples of flooding attacks launched from the network layer. The SYN flooding-based DoS attack is discussed in this method, along with its detection technique, which is evaluated using a standard dataset.

Other flooding attacks, such as RREQ flooding, can be detected using an extended model of the detection system. DoS attacks prohibit the victim node from going into sleep mode, which might cause the node's lifetime to degrade exponentially as a result of the attack. The identification and prevention of such attacks are crucial because they can occur anywhere, at any time, and with varied strength, especially in the networking environment of the Internet of Things. DoS attackers are easily discovered by analyzing the traffic volume or intensity flowing through a certain link. The multiple layers of the TCP/IP protocol stack can be used to launch flooding attacks. SYN flooding is a common and effective DoS attack that uses the transport layer. Hello flooding and RREQ flooding are examples of flooding attacks launched from the network layer.

The majority of attacks are difficult to locate with current techniques and change with the seasons. One of the main challenges to the viability of network infrastructure is a DoS attack. This kind of attack is carried out by attackers ordering private computers that have been infected with malware to establish botnets and deliver massive amounts of network traffic to their targets. Attackers would specifically use certain

protocols, such as Network Time Protocol (NTP) or Domain Name Server (DNS), to deliver pertinent network traffic to a target. As a result of being overwhelmed by these unauthorized users, the victim's bandwidth and server resources may be depleted, preventing genuine users from using internet services.

Numerous techniques are offered to identify a DoS attack because of its significant impact; this can reduce its effects. However, they do not take into account how traffic varies in a network.One of the most useful strategies for preventing DoS attacks from both the source and victim sides. A DoS attack may result in a brief momentary increase in attack traffic that, if it rises beyond the network administrator's configured detection threshold, may result in an attack notification. A static threshold is typically effective on the victim side because attack traffic and legitimate traffic differ significantly from one another. The static threshold method, however, might not be appropriate for spotting complex source-side attack collisions that are easily masked by varying valid traffic. If the fixed threshold is set very high, small attack spikes won't be visible.

On the other hand, normal traffic surges will incorrectly be viewed as an attack if the static threshold is set too low. That is, there could be a lot of false positives. Both the previous threshold and the currently observed traffic are used to update the adaptive threshold. However, during an attack, both attack and legitimate traffic are visible, and this means that the throttling process may be indirectly impacted by the attack traffic. As a result, although having a high detection rate, this approach has a significant rate of false positives. The performance of DoS detection using the adaptive thresholding method can be enhanced if we can precisely distinguish between the exact amount of legitimate traffic and the attack traffic during the attack. For more accurate detection of complex attacks with fewer false positives, this project proposes in this project efficient DoS detection, with adaptive thresholds responsive to traffic seasonality. Threshold calculated using seasonality of traffic determined from periodic traffic statistics, adjusted for observed and predicted typical traffic.

1.1 IoT security issues

It is a challenging endeavor to defend the IoT against a multitude of potential attacks. However, when referred to under its layered design, it is somewhat manageable. Each layer has its restrictions and weaknesses that must be found to assure its security by shielding it from various forms of attacks.

A suitable security mechanism that fixes IoT device vulnerabilities is necessary to prevent such attacks. To do this, we must first focus on the concept of vulnerability and how it affects an attack. A system's vulnerability is the inability of the system to prevent an attacker from evaluating the severity of the security intrusion.

The majority of vulnerabilities are simply a result of our casual and reckless management of IoT devices. Self-awareness is a crucial strategy that we may readily put into place. Users must be thoroughly informed of all the risks involved and the steps to take if their IoT devices exhibit unusual activity. To deal with newer and better attack types, the current security system needs to be supported with features like intrusion detection systems, content filtering, firewalls, inspection technologies and application whitelisting.

1.2 DoS Attack Based on SYN Flooding

The weakness in the TCP standards is exploited by SYN flooding attacks. The connection is regarded as being in a half-open state when the source node sends a SYN packet to the server node in order to establish a connection. To keep track of all of these partially open connections, the server has a backlog queue. The server node responds with a SYN-ACK packet if it is prepared to establish the connection. The source node transmits the last ACK packet to the server node after getting the SYN-ACK packet, completing the three-way handshake. An attacker may send several spoof SYN packets (fake packets) to the victim server in a SYN flooding-based DoS attack. The three-way handshake, victim server never receives the last ACK packet from the client because the SYN request was spoofed. Since the victim server's backlog queue has a limited capacity, a flood of fake SYN requests might quickly empty it, resulting in the flooding of all incoming legitimate SYN requests. Furthermore, the

network becomes congested as a result of thousands of these flooding packets. In an Internet of Things network, the attacker sends several SYN packets to a far away node, which consumes an enormous amount of the victim node's battery backup and computing resources as well as those of the relay nodes, completely depleting their battery resources. As a result, these nodes are removed from the network, which has an impact on how the network normally functions. The attacker deployed in the current network model is taken to have unlimited resources, particularly unlimited computing and battery backup resources in comparison to other network nodes.

1.3 Distributed Denial of Service (DDoS) attack in IoT

When a server or device is the victim of a simple DoS attack, the server becomes unreachable because too much data is present in the communication channel and there is excessive bandwidth being used. Due to the attacker's purposeful use of fraudulent requests to attack the server, legitimate requests suffer as a result and are abandoned before being fully processed.

A Distributed Denial of Service (DDoS) attack is the result of such a breach in a distributed environment, where data is transferred among various network devices scattered throughout the network without any centralized supervision. This is one of those attacks that affect the network architecture's application layer as well as its infrastructure layer (i.e., the network layer and the transport layer). The actual damage brought on by an attack not only results in server failures or website floods but also in a loss of customer and business confidence.

DDoS attacks have been the subject of numerous investigations in the past. However, the majority of them go through basic DDoS attacks, their variations, and defence strategies for conventional networks. The DDoS defense literature contains a depth of knowledge that can be used to construct a baseline model for the present DDoS attack protection in IoT networks. Zargar et al. [3] offered a survey that covers the various botnet-based DDoS flooding attack types as well as other types of DDoS flooding attacks, however, it does not cover new malware variants. Latest malware

attack methodologies have entered the scene as a result of recent DDoS attacks. To create a more effective defense against such attacks, it is crucial to understand some of the recently identified malware varieties and their functions.

Over the past few years, DDoS attacks have displayed some variation in attack methodologies and are still being researched. IoT-specific DDoS attack techniques are very similar to conventional DDoS attack techniques. They employ similar methods to take advantage of weaknesses in IoT devices or traditional systems. However, because of the heterogeneity of IoT devices, DDoS attacks that are specific to the IoT are more varied and advanced. According to their attacking methods, all of these attacks can be roughly divided into three categories, as depicted in Fig. 1.3 [4].

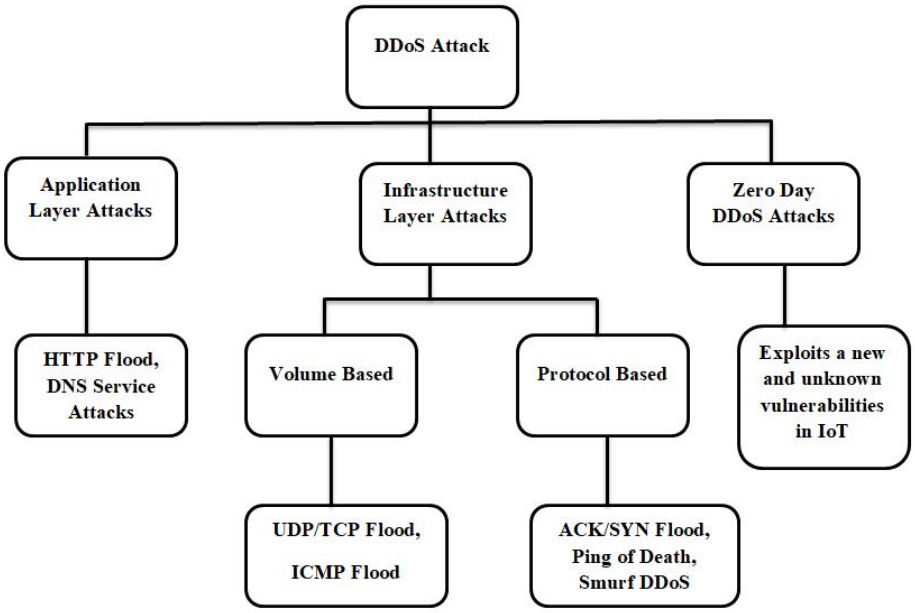

Figure 1.3: DDoS attacks in IoT network

The impact of the attack on the server-side of the IoT network is one of the crucial things to keep in mind about the classification basis for DDoS attacks. The only difference between IoT-specific DDoS attack patterns and conventional DDoS attack patterns is that the relevant network architecture is used as a reference model. Another type of DDoS attack, however, is possible and is based on sending fake requests from an IoT source to the server that is being attacked. To breach the security of application servers based on IoT networks, each layer can be specifically targeted. As a result, the assaults can be roughly divided into infrastructure layer

attacks and application-layer attacks.

Attacks on the application layer are those that attempt to penetrate the IoT network infrastructure where packets are lost at the rate of requests per second (hence measured in Rps) due to flooding of the application or web server with HTTP(Get/Post) requests and other requests that target the system software such as Windows, Apache, OpenBSD etc. Because they tend to create traffic at a lower rate and make requests that appear legitimate but start a backend process that makes the resource unavailable, these attacks are more difficult to identify and mitigate. These comprise attacks using the DNS service, HTTP floods and other methods.

Attacks on the infrastructure layer of the IoT architecture aim to make the target system inaccessible by taking advantage of vulnerabilities in the transport or network layer. These attacks can be of two types: volume-based attacks and attacks based on protocols. To start the attack, they frequently employ reflection or amplification techniques. The attacker, according to the reflection, congests the victim's network by sending the victim's request as an unexpected answer using IP address spoofing. Amplification is another way of saying that you get bigger responses for smaller queries, which also consumes bandwidth.

The real server resources as well as intermediate communication devices like firewalls, load balancers etc. are consumed by protocol-based attacks, also referred to as Resource Depletion attacks. The units of measurement are packets per second (Pps). SYNs, packet fragmentation attacks, Ping of Death, Smurf DDoS etc. are a few examples. By producing an excessive amount of data in bits per second, mass-based attacks, sometimes referred to as bandwidth-draining attacks, saturate the target system's bandwidth (Bps). These are the simplest to utilize because they start the attack using amplification and reflection techniques. According to research, up to 65% of attacks are merely volume-based attacks. UDP/TCP floods and ICMP floods are two examples of these types of attacks. Even though attacks are categorised in this way, they can include elements of the aforementioned infrastructure layer attack categories.

IoT network intruders have advanced to the point where they are testing out new attack vectors to get beyond certain well-known web servers' high-security measures. One of the more recent instances of such a situation is the Dyn DNS Outage, which

combines an application- and protocol-based attack with a volume attack to target a DNS server. Here are a few well-known DDoS attacks that are currently becoming more and more popular.

1.4 Organization of Report

This paper has the following organizational structure. Related works, such as various security attacks and DoS attack detection techniques are presented in Chapter 2. In Chapter 3, the key concept of traffic throttling based on traffic timing is described, along with fundamental aspects of the proposed DoS detection utilizing the adaptive thresholding method. The core of the suggested solution, which defines the adaptive traffic seasonality threshold for DoS attack detection, is included in Chapter 4. The project's results and discussions are included in Chapter 5. Chapter 6, finally put an end to this paper.

Chapter 2

Literature Review

Numerous studies have been done on the detection of flooding or denial-of-service attacks in IoT networks. DoS attack detection has long drawn attention in addition to several studies on traffic engineering in computer systems, including smart controlling systems. There were numerous types of detection techniques, the majority of which concentrated on victim-side detection techniques, including threshold-based techniques, packet confirmation techniques and machine learning techniques. Edge computing trust evaluation techniques are crucial for IoT flood attack detection.

Using Bayesian inference, a novel technique for simulating SYN traffic in the network was presented by N. Nishanth *et al.* [5], with the mean of the method acting as a metric for the SYN Arrival Rate (SAR) of incoming traffic. The mean of the beta distribution is calculated for each sample in order to obtain the average statistics of mean SAR in typical traffic. The mean of the Beta distribution is modified for attack detection to accommodate for variations in incoming traffic. The amount of time mobile nodes spend sleeping also increases how long their batteries last. Additionally, it is reported that without a dedicated server node, it can more accurately and with a higher TPR identify Hello flooding attacks, RREQ flooding attacks, SYN flooding attacks, UDP flooding attacks, and data flooding attacks.

The best technique for RREQ flooding attack detection was developed using Dempster-Shafer (D-S) evidence theory and Bayesian Inference, two examples of uncertain reasoning presented by N. Nishanth *et al.* [6]. This method works well for detecting both high and low rate attacks. The Bayesian Inference-based approach was unable to discriminate between pulsed attacks and seasonal fluctuations. The

frequency of RREQ packets that consistently violated their usual statistics was the main source of evidence utilized to pinpoint attacks. The D-S evidence theory, which uses frequent violations of packet drop statistics as a secondary source of evidence in addition to the primary source of evidence, significantly improved this method for recognizing low rate and high rate pulsed attacks. The proposed technique was found to find both low and high rate bursts.

To obtain the optimum mobility path, Wang *et al.* [7] suggested a Trustworthy Data Collection Model for the Internet of Things Based on Edge Computing. They created the optimal mobility path with high trust by using multidimensional trust value evaluation methodologies in comparison to previous solutions. They overcame the issue of the underlying common nodes' weak computational power and constrained storage capacity using this technique.

The clustering approach for detecting LDoS attacks is another area of critical research. The decision feature of TCP ratio (S_{DFTR}), a novel metric that may be utilized to discriminate between regular traffic and abnormal traffic during LDoS attacks, was used by Dan Tang *et al.* [8]. This method has the advantages of not requiring a predetermined number of clusters, being suitable for any cluster shape, being real-time and adaptable, and having superior robustness and detection effect to MS algorithm, but it is unable to identify the origin of LDoS attacks.

CTRUST is introduced in [9], which accurately parameterizes trust while applying belief functions to evaluate recommendations. This is because the current models do not parameterize trust and erroneously weigh trust suggestions. Sharma *et al.* [10] developed a methodology to decrease the influence of false or dishonest offers in indirect trust calculation by performing both objective and subjective evaluation.

Similar to this, Payam Mohammadi *et al.* [11] presented a strategy based on statistical analysis by maintaining the stability of the network status to defend against flooding attacks in mobile ad-hoc networks. Data packets will not be sent to a node that has been identified as malicious, and that node will be added to the detention list when this occurs. In comparison to more conventional approaches, this method has the advantage of being more effective and efficient than the DSR protocol under flooding attacks, which can increase network throughput overall. Following a logical punishment, the accused node may then be revisited as a typical node in the network.

However, this method does not take seasonal fluctuations into account. An efficient belief-based trust evaluation mechanism (BTEM) that could distinguish between malicious and trustworthy nodes and stop attacks from malicious nodes in WSNs was introduced in [12].

To overcome the resource constraints of WSNs, the exponential-based trust and reputation evaluation system (ETRES) was introduced for node trust and reputation evaluation in [13]. When direct trust is not entirely guaranteed, an indirect trust might increase the interaction information. The uncertainty of direct trust levels is expressed in terms of the entropy theory. The node's trust is considered in this type of research, but the nodes' direct and indirect trust evaluations are frequently biassed, and some studies don't provide specific evaluation indicators to evaluate the direct trust, making the evaluation inaccurate and subjective. Dempster-Shafer (D-S) evidence theory and Bayesian inference have also rarely been combined to evaluate the trust levels of observed nodes in MANETs.

The routing mechanism in Mobile Ad Hoc Network systems will exclude Nodes with low trust values. Wei *et al.* [14] developed Trust Management Using Uncertain Reasoning ways to acquire more accurate trust by taking into account direct and indirect trust computation. The proposed strategy can successfully cope with attacks, which enhanced throughput and Packet Delivery Rate (PDR), according to the experimental performance analysis. However, this approach has the drawback of increasing notification costs and end-to-end delay. In conclusion, only simple traffic is taken into account for attack flood detection, making it challenging to effectively identify floods in real-time within the network and failing to adequately manage traffic variation according to season within the network. Although flooding attacks in the IoT network have been detected in certain studies, there is no specific combination of estimates for seasonal fluctuation in traffic and the underlying application. An adaptive thresholding algorithm is thus required to handle the detection of flooding assaults in IoT networks as a result of seasonal fluctuations.

An Adaptive Threshold Algorithm (ATA) and cumulative sum (CUSUM) technique was proposed by Siris and Papagalou [15] for the identification of SYN flooding-based DoS attacks. The ATA misinterprets attack traffic as regular traffic after some samples of attack traffic owing to persistent attacks, which further results in an in-

creased false alarm rate and decreased accuracy because the threshold is set adaptively based on mean SAR. A system for detecting distributed DDoS attacks in software-defined networks utilizing CUSUM and ATA was proposed by Conti *et al.* [16]. Using data from the Defense Advanced Research Projects Agency (DARPA), this method's performance is assessed.

Gurung and Chauhan [17] proposed implementing Flood Intrusion Detection System (F-IDS) nodes in the network as a means of combating RREQ flooding attacks in MANET. The F-IDS node determined how many RREQ packets were produced by the nodes and compared that number to a dynamic threshold determined using standard deviation. The F-IDS node transmitted an ALERT packet to all other nodes alerting them to the malicious behavior if the RREQ packet rate exceeded the threshold. The blocked nodes were given three chances to rejoin the network as part of a recovery procedure, but if they continued to act maliciously, they were permanently isolated from the network

Chapter 3

Developed Method Based on Traffic Seasonality Aware Adaptive Threshold Algorithm

IoT networks need a solution for detecting DoS attacks that is adaptable and responsive to seasonality in traffic. The suggested approach uses both traffic volume and seasonality to more precisely change the threshold, even in the case of an attack. To distinguish between legitimate traffic and attack traffic and to update attack detection thresholds, it is very important to track the arrival rate of data packets in the network.

To use the DARPA dataset for seasonal flow calculations, the up to the package over time must be withdrawn. This allows it to analyze the IoT network's sampling traffic and build traffic blocks that are detected after equal intervals. The traffic block that was seen in each window must then be set each time. Additionally, check for DoS attacks by determining whether the volume of traffic exceeds the threshold.

In this time interval, a DoS attack is identified if it exceeds the threshold. Two more components, the adaptive thresholding module and the traffic seasonality learning module support the attack detection decision. The adaptive threshold module updates the threshold for the subsequent time slot in accordance with the observed traffic in the current time slot. Using traffic seasonality over a specified period, the traffic seasonality learning module estimates regular traffic on traffic observed during

an attack.

Because the threshold is changed for each time window, the developed method models traffic seasonality in terms of time windows. In other words, it is to discover the seasonality of traffic by learning the tendency of change in traffic in each time window. For the sake of this approach, we suppose that each traffic window has n time frames. If we want to make the time window's minimum length one minute. The set of observed traffic for each time window can then be defined.

This information allows it to determine the fluctuating rate of observed traffic for each traffic window between two continuous-time windows. Get such a set of changing rates assuming there is a traffic window. Then, to determine the trend of traffic change inside a time window, it can calculate all changing rates average value of corresponding for each traffic window. As a result, the suggested procedure depicted in Fig. 3.1 and 3.2 can be summed up as having two phases, as follows: The adaptive threshold method shown in Fig. 3.2 is used to first detect flooding attacks after learning the seasonality traffic from dataset Fig. 3.1. So, two algorithms were created for each of these stages.

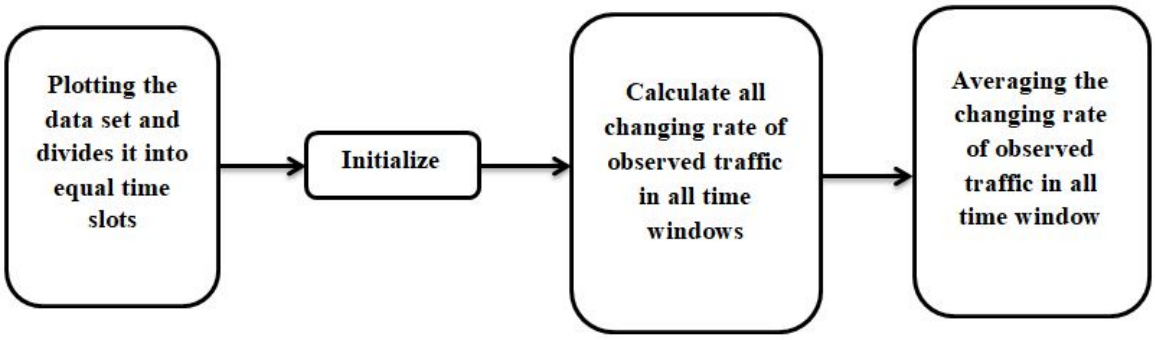

Figure 3.1: Seasonal Traffic Calculation

The first algorithm utilizes a packet arrival rate plot from the DARPA dataset as input, which contains the observed traffic for all time windows with the specified appropriate size and produces the average change rate as the algorithm's output.

Three steps make up the seasonal variation learning algorithm:

1. Initialization-The average changing rate and changing rate are initialized as 0.

2. Calculating the observed traffic's changing rate over all time frames.

3. Calculating the changing rate mean for observed traffic in all time windows.

It can then apply a different method to detect the flood that takes into account the traffic observation from the previous algorithm after computing the seasonal variance.

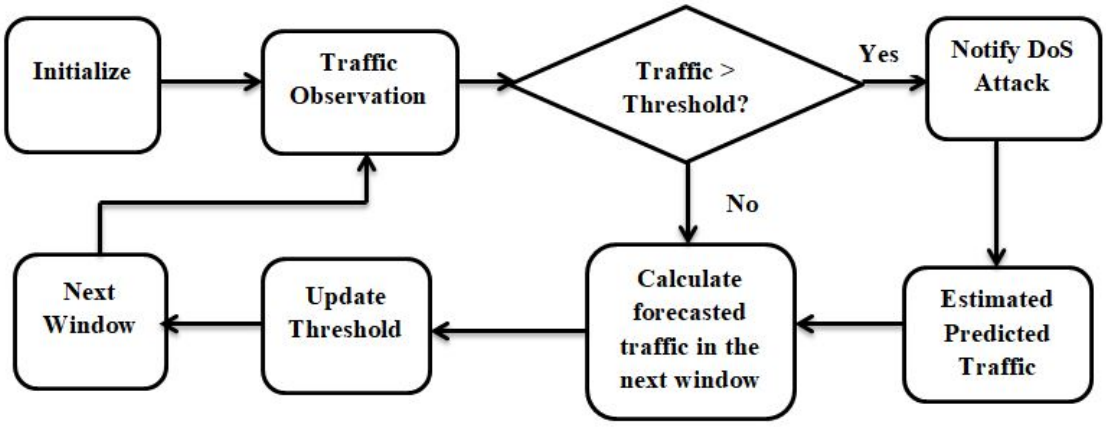

Figure 3.2: Attack Detection

The initial threshold is set to infinity in this approach, and assume that in first time window no attack is present. The minimum time window length is set at one minute.
The following five steps make up the algorithm:

1. Initialization process on every time window.

2. Traffic observation: The traffic observed is derived from a previous algorithm.

3. Attack detection: The threshold is compared to the observed traffic. An attack is detected on the appropriate time window if it exceeds the threshold.

   (a) Notification of attack: A function notifies of an attack if one is found.

   (b) Estimated legitimate traffic calculation: The estimated legitimate traffic is computed using seasonality if an attack is detected. It is determined how much estimated legitimate traffic there would be if the attack covered numerous time frames.

4. Forecasted traffic calculation: The forecasted traffic during the next time window is determined. The threshold is updated if no attack is found.

5. Threshold update: The margin value is used to compute the next time window's threshold.

The algorithm will then run for the subsequent time window.

Chapter 4

Adaptive Traffic Seasonality Aware Threshold for Detecting DoS Attacks

This project proposes the traffic seasonality aware adaptive threshold method to accomplish effective DoS attack detection. Even though there is an attack, the suggested solution better adapts the threshold by considering both traffic volume and seasonality. In particular, the traffic seasonality is gathered through network monitoring and is utilized to distinguish between legitimate traffic and attack traffic as well as to update the threshold for identifying attacks. It starts by describing the setup for gathering and monitoring network traffic on a network where SDN technology is being used. Then, go over the fundamentals of the adaptive threshold technique with fluctuating traffic. Then, discuss how to determine the traffic's seasonality and how to modify the threshold based on that information.

A local network's gateway could be a useful place to put the functionality of DoS attack detection to identify attacks close to their sources, particularly with IoT devices. The proposed DoS attack detection's total environment is shown in Fig. 4.1. The gateway has SDN capabilities, and by modifying the Monitoring policy table, an outer SDN controller could properly establish the monitoring criteria for the gateway. Out of all the internet traffic that moves from the neighbourhood network to the Internet, the traffic in the network related to the specific protocols chosen by the

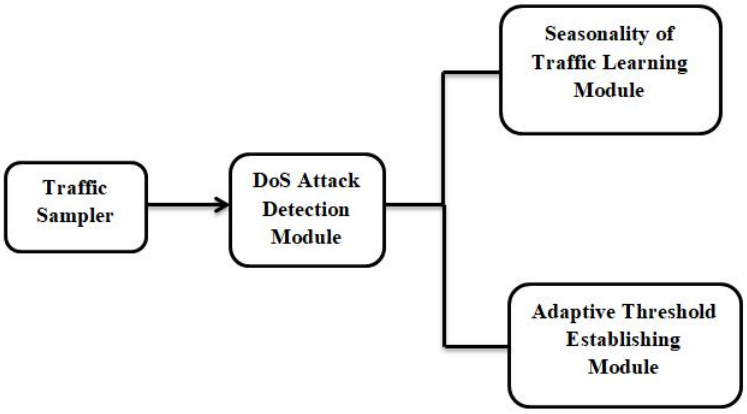

Figure 4.1: Environment of the proposed DoS attack detection

monitoring policy, such as NTP and DNS, is tracked and transmitted to the DoS detection module. In other words, one can deliberately monitor some dubious sorts of protocols by setting up monitoring settings in the described environment.

Traffic samplers that satisfy the selected protocols are present in the Dos detection module. After each equal period, a sampler generates the observed traffic block by receiving the corresponding forwarded traffic. This period is referred to as a time window and its length is tw. s_z is the definition of the observed traffic block during the z^{th} time interval. The attack detection decision module determines whether or not a DoS attack has occurred by determining whether the observed traffic volume exceeds a threshold. If it rises above the threshold, a DoS attack is detected during that window of time.

Adaptive threshold establishing and traffic learning of seasonality are two additional modules - support the attack detection decision module. Based on the observed traffic of the current time window, s_z, the adaptive threshold setting module modifies the threshold of the following time window, θ_{z+1}. The seasonality of traffic learning module uses the seasonality of traffic for a certain time, to estimate the volume of regular traffic compared to the attack-related traffic observed.

The DoS detection module's primary goal is to identify subtle attack traffic among highly fluctuating legitimate traffic. Some minor attack traffic peaks are mixed in with this highly fluctuating traffic. Since the attack happens near the attack sources

instead of the target and because the attack sources could be dispersed across multiple different subnetworks, attack traffic may be relatively low. The static threshold method cannot detect numerous minor assault peaks in the deep valley of the legal traffic, making it difficult to apply in this situation.

As a result, to detect minor attack peaks, the threshold needs to be carefully adjusted to follow the legal traffic. Additionally, a considerable margin can be employed when adjusting the threshold to avoid false detection caused by fluctuations in legitimate traffic. This threshold update is performed on the adaptive threshold establishing module for each time frame using an exponential smoothing function and the measured traffic for each time window.

To make the threshold follow the legal traffic effectively by adjusting the smoothing algorithm, preventing spontaneous attack peaks. When an attack occurs, both legal and attack traffic is present in the observed traffic during a time interval. In this situation, it is necessary to distinguish between legitimate traffic and attack traffic to properly modify the threshold. However, it is very challenging to accurately segregate this traffic in real-time, thus this project is attempt to calculate the volume of real traffic while accounting for seasonality.

In other words, one can anticipate the volume of legitimate traffic by learning the traffic changes likelihood in a specific time window from the trend of changes in traffic over a relatively long period. This procedure is carried out on the traffic learning module's seasonality. It modifies the barrier more conservatively based on the anticipated number of legitimate traffic.

4.1 Traffic Volume aware Adaptive Threshold Establishment

The threshold for the subsequent time window, θ_{z+1}, will be determined by utilizing an exponential smoothing function with the observed traffic in the immediate time window s_z, in the adaptive threshold setting module. Additionally, the margin (δ) is used to leverage the threshold.

In essence, the forecasted traffic of the upcoming time window is used to calculate

the threshold value. Consider that can acquire the observed traffic s_z, at the z^{th} time window after receiving the forecasted traffic, $\bar{s_z}$, at the $(z-1)^{th}$ time window. The predicted traffic in the $(z+1)^{th}$ time frame $\bar{s_{z+1}}$, is determined by using the exponential smoothing function, and is computed as follows:

$$\bar{s_{z+1}} = \mu * s_z + (1-\mu) * \bar{s_z} \tag{4.1}$$

In which the smoothing parameter is the coefficient μ $(0 \leq \mu \leq 1)$. The forecasted traffic for the z^{th} time frame would be given greater weight to the next one the smaller is μ. On the other hand, the greater the value of μ, the more significance the present observation would have on the following one. In this work, we maintain this coefficient's value at 0.5.

It is necessary to assign the forecasted value of the first time window, s_1, as there isn't a value before it because this forecasting uses the prediction from the previous time window as input. This initial value is what is assigned as the total number of requests throughout the first time interval.

$$\bar{s_1} = s_1 \tag{4.2}$$

The following set of equations would then represent the predicted traffic.

$$\bar{s_2} = \mu * s_1 + (1-\mu) * \bar{s_1}$$

.....

$$\bar{s_{z+1}} = \mu * s_z + (1-\mu) * \bar{s_z}$$

$\bar{s_{z+1}}$ can therefore be determined using the regression form as shown below.

$$\bar{s_{z+1}} = \sum_{i=0}^{z-1} \mu * (1-\mu)^i * s_{z-i} + (1-\mu)^z * s_i \tag{4.3}$$

With the help of the forecasted traffic, one may define a threshold that, over the long run, will adjust under the forecasted traffic. However, there may be local traffic fluctuations that mistakenly surpass the threshold, leading to false positives and misdetection. One can use a margin value of δ $(\delta \geq 0)$ to give a meaningful difference

between the forecasted traffic and the threshold to avoid these unintentional false positives. This detection method's knob parameter is the margin value. The detection method is more adaptive to local changes when the margin value is bigger, but it may miss some attack peaks.

The $(z+1)^{th}$ time window's threshold is therefore determined as follows:

$$\theta_{z+1} = \bar{s_{z+1}} * (1 + \delta) \tag{4.4}$$

The attack is then identified if the observed traffic for the $(z+1)^{th}$ time frame exceeds the adjusted threshold, which is passed to the attack detection decision module.

If there is attack traffic present in the observed traffic, this threshold change based on that traffic may not function properly. The observed traffic s_z, which includes both the attack and legitimate traffic, should not be used to determine the threshold for the $(z+1)^{th}$ time window when an attack is identified at the z^{th} time window.

If the observed traffic includes attack traffic and is used to generate the expected traffic using equation 4.1, the threshold and forecasted traffic for the next time window would be overestimated. Because of this increased threshold, some minor attacks cannot be recognized. As a result, to properly change the attack threshold, one can estimate the percentage of legitimate traffic from the observed data that consists of attack traffic using the seasonality of traffic. This is covered in more detail in the next part.

4.2 Threshold Adjustment by Learning Traffic Seasonality

The core idea of threshold adjustmentby learning traffic seasonality is to update the threshold even in the presence of an attack using estimated legitimate traffic that is obtained from traffic seasonality. The suggested technique considers the seasonality of traffic as the variable rate of traffic in a corresponding time window of a day. It is possible to see the traffic's regular seasonality in a network environment. For instance, the observed traffic in a company can significantly grow in the morning

before progressively declining from the afternoon to the evening. As a result, even with incomplete traffic information, such as observed data that is a mix of legitimate and attack traffic, may estimate the volume of traffic at a given time if we can identify the seasonality, or changing trend, of the traffic at that time.

Additionally, because each network's user behavior or system differs from the others, each network's seasonality is unique. It is the key reason why we must offer a reliable technique for determining the seasonality of traffic in a network.

The procedure of DoS detection with traffic seasonality aware threshold adjustment involves two steps: first, determining the traffic's seasonality, and second, changing the threshold in accordance. Using the traffic records, the seasonality of traffic, or the rate of change between two continuous-time frames, is determined in the first step. The second phase covers how to forecast traffic that is computed with the estimated legitimate traffic and how to adjust the threshold with that traffic. The forecasted traffic is calculated with the estimated legitimate traffic.

4.2.1 Obtaining the seasonality of traffic

It models the traffic seasonality in terms of time windows because the threshold is adjusted on every time window. In other words, one can discover the Detecting the seasonality of traffic by tracking its growth change across every time window.

Assume that there are k time windows every day and that there are n days with legitimate traffic records $D = \{d_1, d_2, ..., d_n\}$. The size of the time window determines the actual value of k. Let t_w is the length of the time window if the time window's minimum value is one minute. Following that, one may define a collection of observed traffic for the y^{th} day, s_z ($1 \leq z \leq k$), as follows:

$$d_y = \{s_1, s_2, ..., s_k\} \tag{4.5}$$

Here, the expression $d_y[s_z]$ is employed to give s_z in d_y, or the traffic that was seen during the z^{th} time frame of the y^{th} day. With the use of this information, one can determine the daily variation in the observed traffic rate between two consecutive time frames and characterize it as follows:

$$c_y = \{ch_1, ch_2, ..., ch_k\} \tag{4.6}$$

Here, the changing rate for the z^{th} time window of a day is denoted by ch_z ($1 \leq z \leq k$), and the for the y^{th} day, a set of changing rates is indicated by c_y. The changing rate for the z^{th} time window on the y^{th} day is denoted by the symbol $c_y[ch_z]$, and it is calculated as follows:

$$c_y[ch_z] = \begin{cases} \dfrac{d_y[s_k]}{d_{y-1}[s_k]} & z = 1 \\ \dfrac{d_y[s_z]}{d_y[s_{z-1}]} & z \geq 2 \end{cases} \tag{4.7}$$

When n days of logs are assumed, one can obtain a collection of changing rates $C = \{c_1, c_2, ..., c_n\}$. Calculate the average value of all changing rates that correspond to the same time windows for each day in this section to assess the trend of traffic fluctuation on a given time window. In this case, consider the average change rate in the z^{th} window of time as z and compute it as follows:

$$\Delta_z = \frac{\sum_{y=1}^{n} c_y[ch_z]}{n}, 1 \leq y \leq n, 1 \leq z \leq k \tag{4.8}$$

Get a collection of average daily changing rates as shown by the equation: $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_k\}$. This estimation is made for each time window and is based on the traffic learning module's seasonality. Algorithm 1 presents the whole algorithm for determining the seasonality of traffic.

A traffic record of n days that contains the observed traffic for all time intervals of the chosen size is provided as the method's input. The algorithm's output is $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_k\}$. There are three steps in the algorithm:

- Initialization - calculating the observed traffic's changing rate across all time windows and averaging that rate across all time windows. Initialized to zero is the a day's average rate of change Δ_z ($1 \leq z \leq k$), as well as the rate at which each time frame changes on a given day $c_y[ch_z]$ ($1 \leq y \leq n$ and $1 \leq z \leq k$).

- Equation 4.7 is used to determine all of the changing rates for the y^{th} day of the z^{th} time window ($1 \leq y \leq n$ and $1 \leq z \leq k$). There is no time frame from the previous, hence the value of $c_1[ch_1]$ is set to 0.

- Equation 4.8 calculates the changing rate mean Δ_z ($1 \leq z \leq k$) of a time frame in a day. Since the value of $c_1[ch_1]$ is set to 0, instead of dividing by n, the average value for the first time window ($z = 1$) is divided by $(n - 1)$.

Algorithm 1 Traffic Seasonality Learning, Δ

Input:$D = \{d_1, d_2 \ldots, d_n\}, d_y = \{s_1, s_2 \ldots s_k\}, t_w$
Output: $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_k\}$

Initialize
for y=1 to n **do**
    $c_y = \{ch_1 = 0, ch_2 = 0, ..., ch_k = 0\}$
    $\Delta = \{\Delta_1 = 0, \Delta_2 = 0, \ldots, \Delta_k = 0\}$
end for
for y = 1 to n **do**
    for z = 1 to k **do**
        if z = 1 **then**
            if y != 1 **then**
                $c_y[ch_z] = \frac{d_y[s_z]}{d_{y-1}[s_k]}$
            else$c_y[ch_z] = 0$
            end if
        else$c_y[ch_z] = \frac{d_y[s_z]}{d_y[s_{z-1}]}$
        end if
        $z = z + 1$
    end for
    $y = y + 1$
end for
for z = 1 to k **do**
    if z != 1 **then**
        $\Delta_z = \frac{\sum_{y=1}^{n} c_y[ch_z]}{n}$
    else$\Delta_z = \frac{\sum_{y=2}^{n} c_y[ch_z]}{n-1}$
    end if
end for

The total number of time windows in a day, k and the number of days, n, has a significant impact on the complexity of algorithm 1. Each stage of algorithm 1 comprises two nested loops that each accept k and n entities. The temporal complexity, which can be reduced to $O(n * k)$, is, therefore, $O(3 * n * k)$ for determining the seasonality of traffic. The $(n * k)$ values of $d_y[s_z]$, $(n * k)$ values of $c_y[ch_z]$, and k values of Δ_z ($1 \leq y \leq n$ and $1 \leq z \leq k$) are kept for calculation in the aspect of the space complexity of the algorithm 1 that deals with space complexity. As a result, the space complexity for calculating traffic seasonality is $O(2 * n * k + k)$, which can be reduced to $O(n * k)$.

4.2.2 Adaptive Threshold for seasonality of traffic

As already mentioned, the seasonality of traffic is utilized to determine how much of the observed traffic is authentic and how much is being targeted. Equation 4.1 is used to calculate the predicted traffic of the $(z+1)^{th}$ time window using the current observed traffic s_z, and equation 4.4 is used to change the associated threshold if there is no attack at the z^{th} time window. Instead of using the current observed traffic s_z, when an attack is detected at the z^{th} time frame, calculate the forecasted traffic $\bar{s_{z+1}}$, using the estimated legitimate traffic $s_{z\_predict}$, as follows:

$$\bar{s_{z+1}} = \mu * s_{z\_predict} + (1-\mu) * \bar{s_z} \tag{4.9}$$

Here, $s_{z\_predict}$ is determined by multiplying the traffic that was seen within the prior time window s_{z-1}, which is thought to include only valid traffic, by the change rate z of the subsequent z^{th} time window, as follows:

$$s_{z\_predict} = s_{z-1} * \Delta_z \tag{4.10}$$

Δ_z represents the typical traffic change rate between the $(z-1)^{th}$ and z^{th} time windows. and must ensure that no attack occurs during the $(z-1)^{th}$ time interval. Equation 4.4 is used to update the corresponding threshold after computing the forecasted traffic with the calculated actual traffic using the traffic's seasonality.

If an attack lasts for a longer period than a time window allows for and the attack crosses more than a one-time frame. One cannot estimate the legal traffic (i.e., $s_{z\_predict}$) using the observed traffic from the last time frame (i.e., s_{z-1}) since the observed data includes an unknown quantity of attack traffic. Use the predicted legitimate traffic of the next time window as equation 4.11 as a result of using the expected legitimate traffic in subsequent computations.

$$s_{z\_predict} = s_{z-1\_predict} * \Delta_z \tag{4.11}$$

In addition, Equation 4.10 is used to calculate predict s_{z-1} with the values of s_{z-2} and Δ_{z-1} and s_z predict is obtained using equation 4.11 if no attack is present on $(z-2)^{th}$ time window and attacks occur continually on the $(z-1)^{th}$ and z^{th} time

windows. Only when no further attack traffic is detected will the iterative calculation of the estimated legal traffic using equation 4.11 come to an end.

Algorithm 2 DoS detection with threshold adjustment for seasonality

Input:$\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_k\}$, Margin δ, Smoothing parameter μ, Time window length t_w

Output: Attack notification

Initialize
$\theta_z = \infty$
$cAttack = 0$
while true **do**
    $s_z = getObservedTraffic(z)$
    if When the algorithm begins, the initial time window is denoted z **then**
        $\bar{s_z} = s_z$
    end if
    if $s_z \geq \theta_z$ **then**
        NotifyDoSAttack(z)
        if cAttack = 0 **then**
            $s_{z\_predict} = s_{z-1} * \Delta_z$
        else$s_{z\_predict} = s_{(z-1)\_predict} * \Delta_z$
        end if
        $\bar{s_{z+1}} = \mu * s_{z\_predict} + (1 - \mu) * \bar{s_z}$
        $cAttack++$
    else$\bar{s_{z+1}} = \mu * s_z + (1 - \mu) * \bar{s_z}$
        $cAttack = 0$
        $\theta_{z+1} = \bar{s_{z+1}} * (1 + \delta)$
    end if
    $z = z + 1$
end while

The developed algorithm, which uses traffic seasonality aware threshold adjustment to detect DoS attacks, is described in algorithm 2. This algorithm's minimum time window size is one minute, and z is initially set using the current time in minutes. For each time window, the startup phase is followed by the steps of traffic observation, attack detection, attack notification, estimated legitimate traffic calculation, predicted traffic calculation and threshold change.

- The function *getObservedTraffic(z)* returns the observed traffic s_z.
- The threshold θ_z is compared with the observed traffic s_z. An attack is detected on the appropriate time frame if it rises beyond the threshold.

  The function *NotifyDoSAttack(z)* notifies the user of an attack if one is found.

If an attack is identified, seasonality is used to estimate the estimated legitimate traffic ($s_{z\_predict}$). If the attack covers many time windows (cAttack $\neq$ 0), equation 4.11 is used to estimate the legitimate traffic. Otherwise, equation 4.10 is used to calculate it.

- It is determined how much traffic is forecasted within the following time window $(z+1)^{th}$. If no attack is detected, equation 4.1 is used to adjust the threshold. If an attack is detected, the threshold is modified using equation 4.9 otherwise.

- Equation 4.4 is used to calculate the threshold θ_{z+1} of the next time frame $(z+1)^{th}$ utilizing the margin value.

The algorithm will then run for the subsequent time window. If z exceeds the daily limit, z resets to 1 for the next day. On each time window, algorithm 2 constantly runs the main loop. Algorithm 2's time complexity is just $O(1)$ because there isn't a loop inside of this main loop. The seasonality representatives, i.e., $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_k\}$, and in terms of space complexity, and a few other variables inside the main loop, are saved while the algorithm is running. As a result, algorithm 2's space complexity is just $O(k)$.

Chapter 5

Results and Discussions

5.1 Simulation environment

MATLAB: The MathWorks company created the proprietary multi-paradigm programming language and computing environment known as MATLAB. Matrix manipulation, function and data visualization, algorithm implementation, user interface building, and connecting with other programming languages are all possible with MATLAB.In this project MATLAB is used as a platform to simulate and detect DoS attack in IoT network.The proposed approach was evaluated offline using traffic statistics and then implemented in MATLAB (version R2021a). The MATLAB software is used to plot traffic statistics and the detection window for the proposed methodology for identifying flooding attacks.

Dataset: The Information Systems Technology Group at MIT Lincoln Laboratory created the DARPA dataset in 1999, which is used to evaluate the effectiveness of the proposed method. Three weeks of training data were made available for the 1999 DARPA Intrusion Detection off-line evaluation. It comprises three weeks of training data, the first and third of which are free of attacks, and the second of which is filled with various kinds of labeled attacks. It is accessible as a "tcpdump" file. These data were produced using a test network that gathered data traffic for twenty-two hours a day and were made available to assist in the training of anomaly detection algorithms. The training data for the second week include a subset of the attacks from the 1998 evaluation as well as many fresh attacks. The incoming packets were divided into TCP, UDP and ICMP categories based on the IP header's protocol field.

TCP, UDP and ICMP packets are represented by the values 06, 11, and 01 in the IP header's protocol field, respectively. Based on the flag bits present in the TCP header, the incoming TCP packets were categorized as SYN packets, FIN packets, RST packets etc. The synchronization, finish and reset packets are denoted by the abbreviations SYN, FIN, and RST respectively. The algorithm, which depends on the SYN arrival rate (SAR), is as follows:

$$SAR = \frac{Number of SYN packets}{Total number of TCP packets} \tag{5.1}$$

SAR is defined as the proportion of SYN packets to all packets that were collected. Similarly, the ratio of FIN packets to all packets recorded is known as the FIN arrival rate.

5.2 Performance parameters

A list of the many measures used to evaluate the algorithm's performance is provided below:

1. True Positive (TP) – The attack that actually occurs and is expected to occur is positive (Attack)

2. True Negative (TN) – The predicted negative (Normal) is the actual negative (Normal)

3. False Positive (FP) – The predicted positive (Normal) however actual negative (Attack)

4. False Negative (FN) – The predicted negative (Attack) became positive in the actual (Normal)

5. True Positive Rate (TPR) - It is defined as the percentage of actual positives correctly identified. The algorithm is more accurate if TPR is close to 1. It is also called hit-rate, sensitivity, or probability of detection.

$$TPR = \frac{TP}{(TP + FN)}$$

6. False Positive Rate (FPR) - The ratio of FP to the sum of FP and TN. It is also known as Fall-Out.

$$FPR = \frac{FP}{(FP + TN)}$$

7. False Negative Rate (FNR) - It is the percentage of all positive samples that were accidentally classified as negative samples.

$$FNR = \frac{FN}{(FN + TP)}$$

8. False Detection Rate (FDR) - It is the proportion of the total number of samples (n) to the sum of FP and FN.

$$FDR = \frac{(FP + FN)}{n}$$

9. Precision (P) - the proportion of TP to all samples with positive predictions.

$$P = \frac{TP}{(TP + FP)}$$

10. Accuracy (A) – The ratio of the sum of the predicted true values to the total number of observations, n, is used to measure accuracy.

$$A = \frac{(TP + TN)}{n}$$

11. Confusion Matrix (CM) - The confusion matrix is used to evaluate the algorithm's precision. Based on the classification problem, compares the performance of the actual and predicted values. The rows in the matrix represent actual class events, whereas the columns in it reflect expected class events. A 2X2 square matrix is the CM in this situation. It is provided by

$$C = \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$$

5.3 Performance Assessment of Different Existing Methods Using the Test Dataset

The DARPA data set was used to assess the effectiveness of a few recent detection approaches. Most methods had a fixed or adaptive threshold, and an anomaly was detected if the rate of incoming packets exceeded the threshold. It is performed to measure the proposed methodology with cutting-edge attack detection methods like Fixed Threshold Algorithm (FTA) and Adaptive Threshold Algorithm (ATA).

The detection window for the FTA was plotted and presented in Fig. 5.1 with an optimum threshold of 0.16.

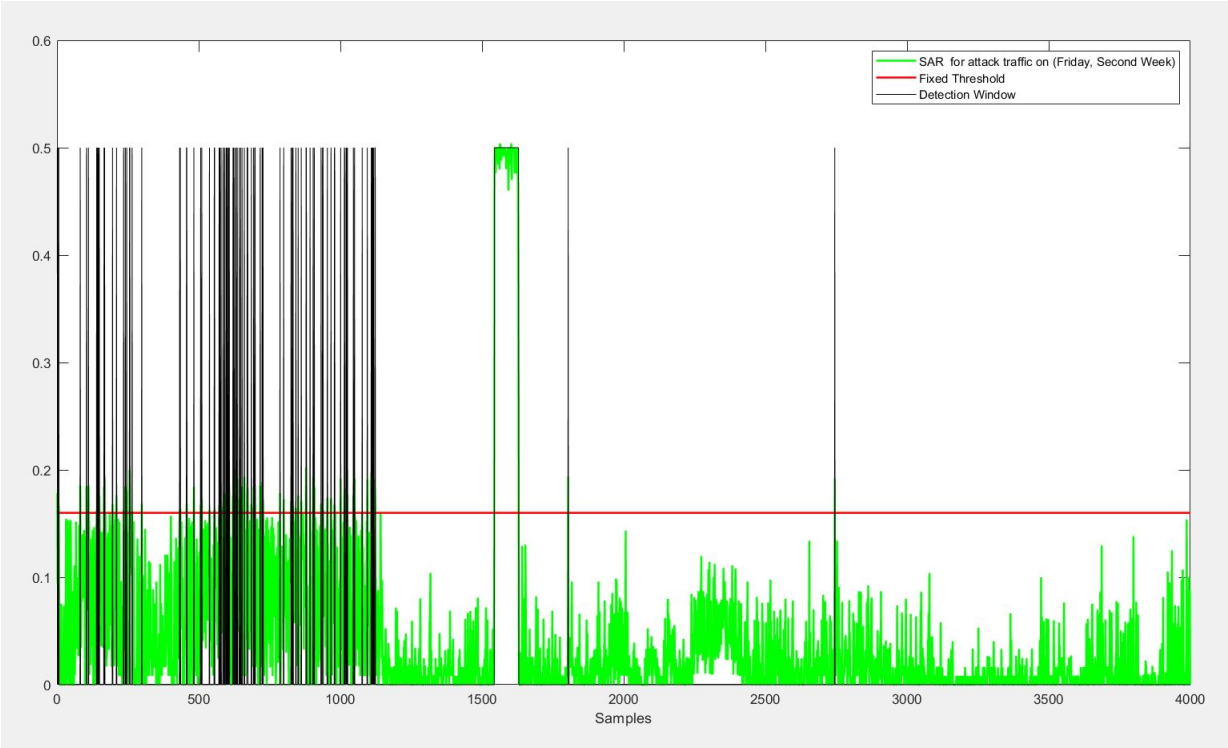

Figure 5.1: Plot for detection window using Fixed Threshold Algorithm (FTA) [18]

This scheme's confusion matrix evaluation is given by,

$$C = \begin{bmatrix} 102 & 11 \\ 55 & 4332 \end{bmatrix}$$

According to Fig. 5.1, FTA has a greater false positive rate as well as a higher detection rate. The establishment of an ideal threshold for attack detection is the

FTA's limitation. Any network's threshold must be determined over a prolonged period of continuous network monitoring. Higher misdetection or false alarm rates would follow from selecting a suboptimal value for the threshold. The fixed FTA threshold in this instance is 0.16, and it only applies to the dataset given. A new threshold needs to be established whenever the dataset changes.

For the Friday of week two's attack traffic, the detection scheme based on ATA is presented in Fig. 5.2 for the values of β= 0.08, α= 0.9, and k = 5.

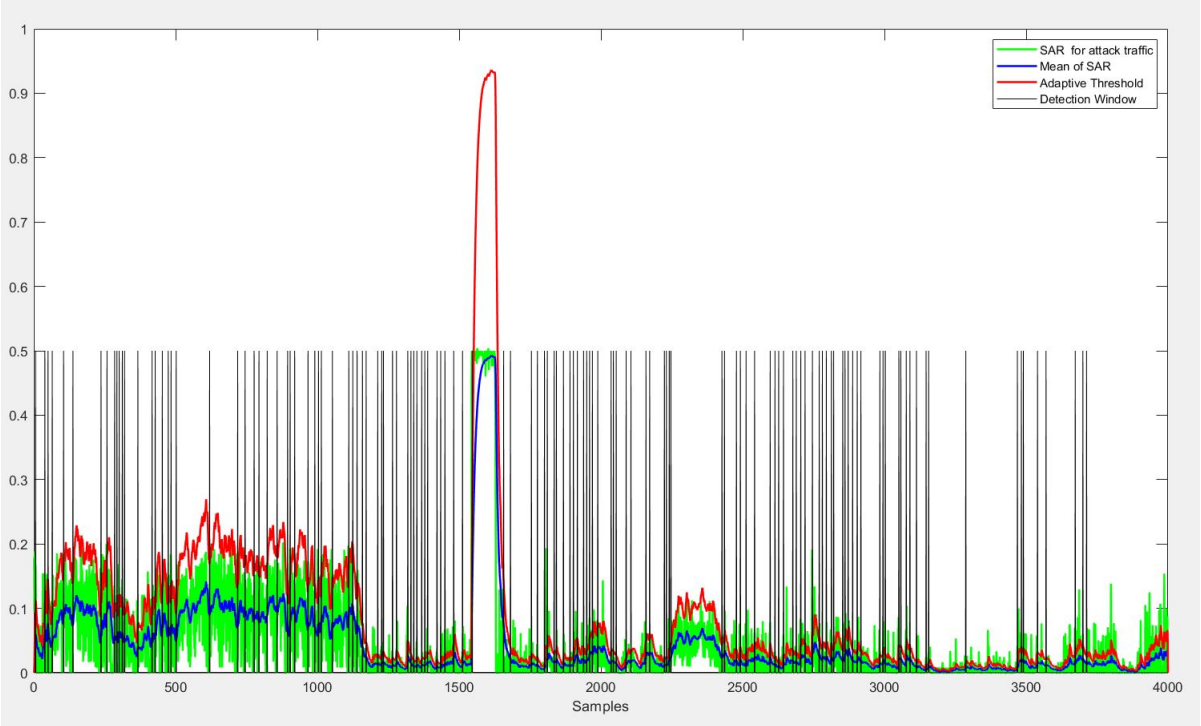

Figure 5.2: Plot for detection window using Adaptive Threshold Algorithm (ATA)

The evaluation of the confusion matrix for ATA is provided by,

$$C = \begin{bmatrix} 36 & 77 \\ 38 & 4349 \end{bmatrix}$$

5.4 Simulation results

For typical traffic, the rates of SYN and FIN arrivals are equal, and the proportion of RST packets is relatively low.The usual SAR on of the first week is depicted in Fig. 5.3.

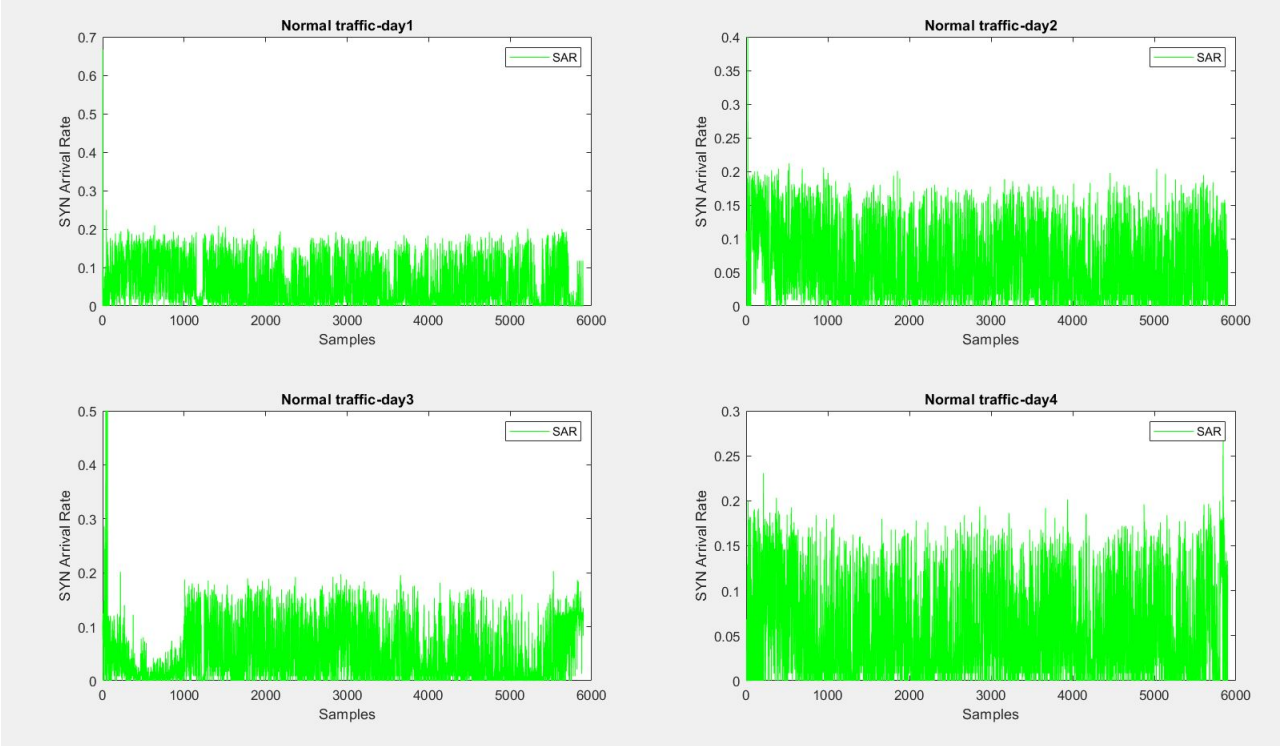

Figure 5.3: SAR statistics during of week one normal traffic

The average changing rate of the first week is shown in Fig. 5.4. The SAR statistics for the attack traffic on Wednesday of week two are shown in Fig. 5.5. When a flooding-based DoS attack occurs, a remarkable spike in SAR is seen.

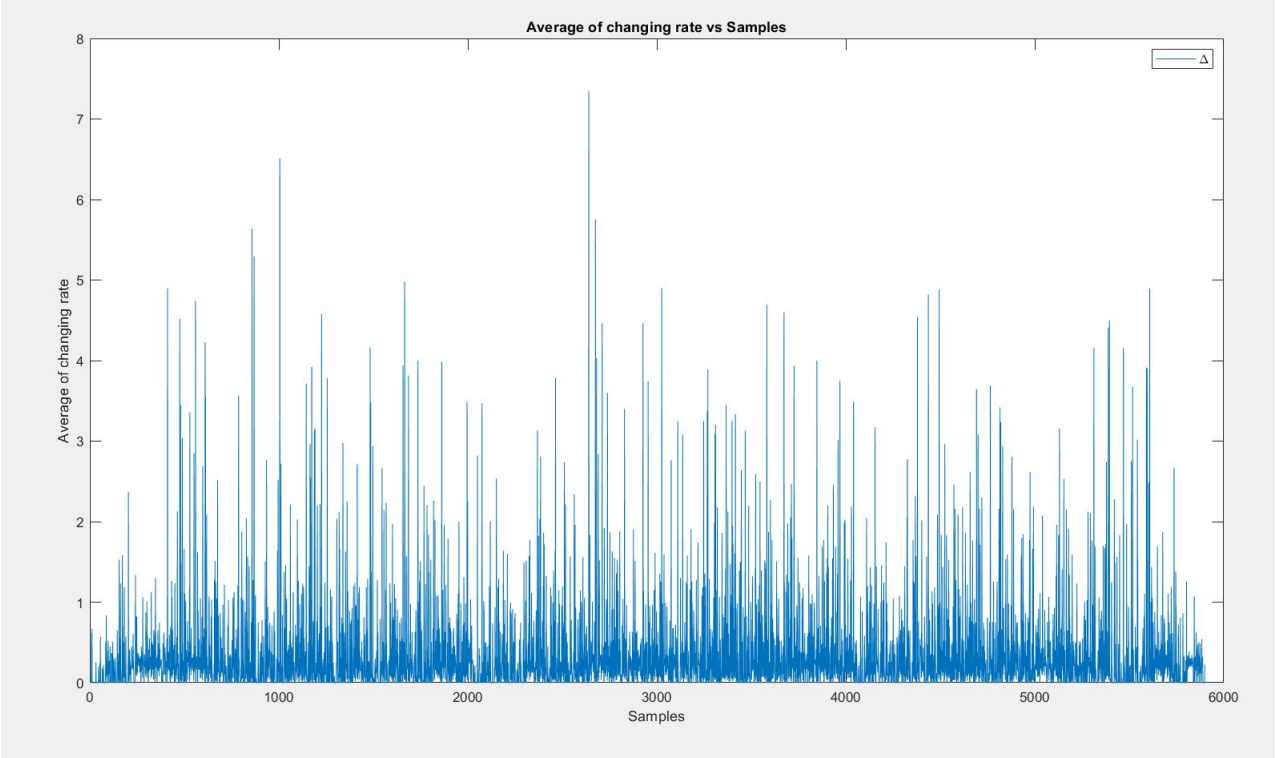

Figure 5.4: Average of Changing Rate vs Samples

5.5 SYN Flooding Attack Detection Using the Proposed Algorithm

According to Algorithm 1, an adaptive threshold algorithm with a seasonality learning approach has been utilized to detect anomalies in network traffic. Plot proposed method simulated results of attack detection is shown in Fig. 5.6

The evaluation of the confusion matrix for Traffic Seasonality Aware Adaptive Threshold Adjustment Algorithm (TSA-ATA)is provided by,

$$C = \begin{bmatrix} 84 & 0 \\ 3 & 3912 \end{bmatrix}$$

As a result, the proposed approach has successfully recognized the network traffic anomaly with greater TPR and precision. The proposed method was able to detect persistent and pulsed SYN flooding attacks with improved accuracy and fewer false-positive rates, according to the confusion matrix for the algorithm's proposed algorithm based on the Seasonality learning aware adaptive threshold. If pulsed attack

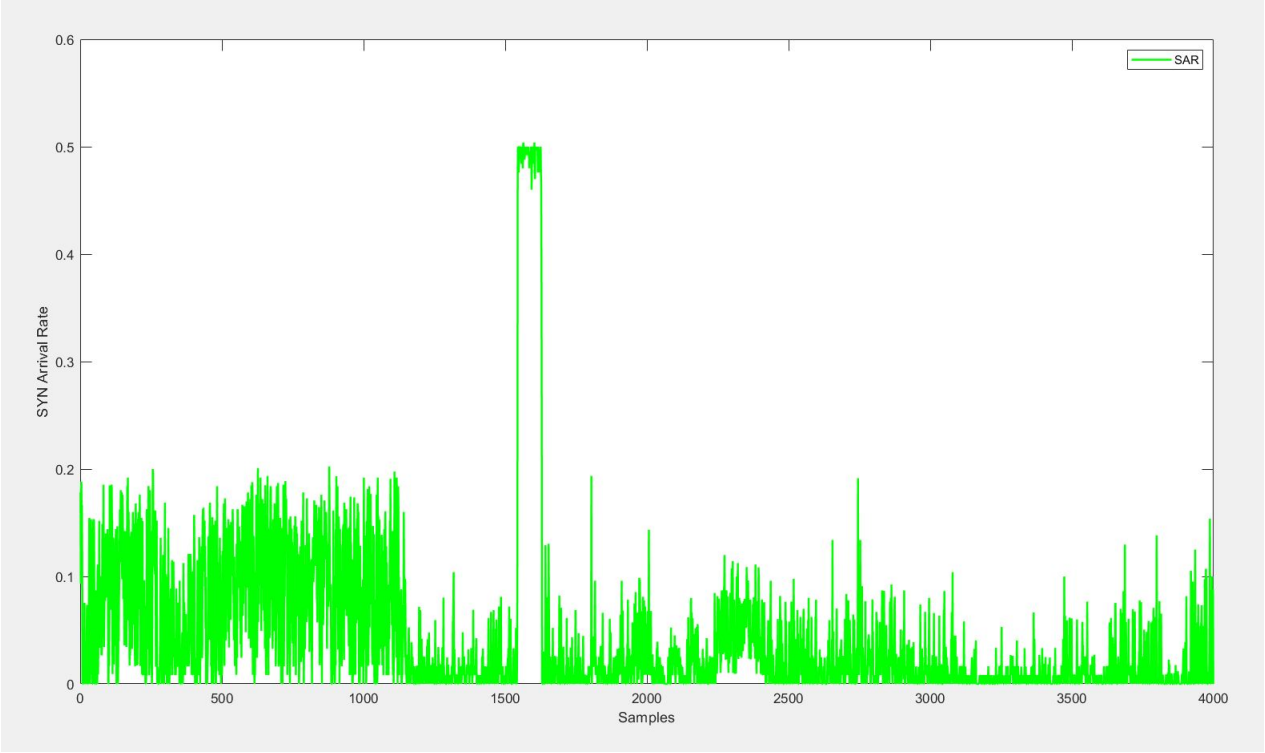

Figure 5.5: Attack Traffic vs Samples

samples were available in the DARPA dataset, packet drop statistics for these samples were also taken into account for computation, which led to a longer computation time than with other approaches. However, when compared to existing techniques, the delay in the decision-making process for the detection of persistent attacks was the same.

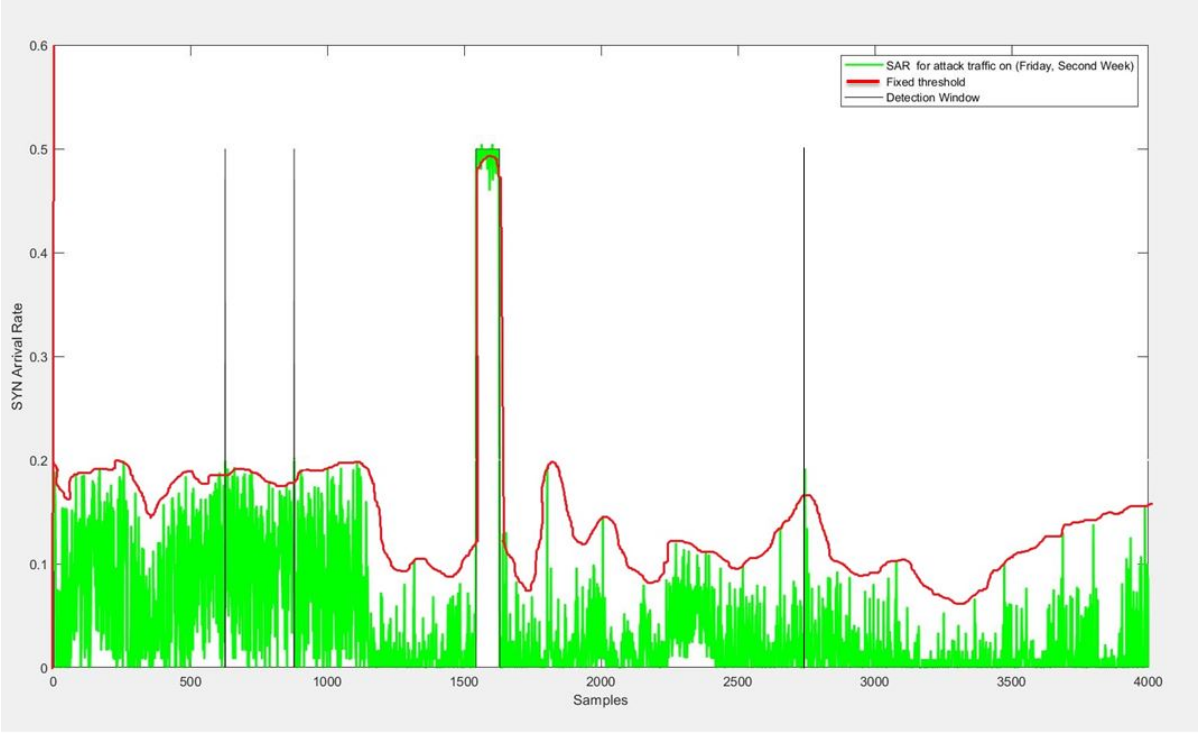

Figure 5.6: Simulation results of Attack detection

5.6 Comparison of performance parameters

The comparison of the various performance parameters obtained for the different algorithms is discussed in Table. 5.1.

Table 5.1: Comparison of the suggested method with the current methods

Detection Scheme	TP	TN	FP	FN	A(%)	TPR(%)	P(%)	FPR(%)	FDR(%)
Fixed Threshold Algorithm	102	4332	55	11	98.53	90.26	64.97	1.25	1.47
Adaptive Threshold Algorithm	36	4349	38	77	97.44	31.85	48.65	0.86	2.56
TSA-ATA Algorithm	84	3912	3	0	99.93	100	96.55	0	0

DoS attack detection uses a fixed threshold with lower accuracy than the adaptive threshold technique when comparing the detection schemes. Compared to these two methodologies, the accuracy of the proposed methodology is significantly higher.

The False Positive Rate (FPR) is a crucial parameter to consider when evaluating a detection scheme's effectiveness. The FPR ought to be as low as possible for an effective strategy. With that in mind, it is evident from the table that the proposed strategy is superior to the alternatives. The False Negative (FN) value of the proposed technique is null, whereas the others have a value to take into account that variable,

which is an interesting fact.

Since the average SAR was calculated for both legitimate traffic and malicious traffic, it was also noted that ATA suffered from a high false-positive rate and lower accuracy. Additionally, just a portion of the attack was alerted to the alarm, which increases the chance of misdetection. The suggested strategy had the lowest percentage of false positives and found every attack point in the dataset. In comparison to other strategies that were taken into consideration, the suggested method likewise provided the highest accuracy.

Chapter 6

Conclusion

Applications of the Internet of Things network, such as connected automobiles, traffic management, smart grid, and smart buildings/smart homes are affected by flooding-based DoS attacks. The SYN traffic is used in the current work to learn about traffic seasonality, and an effective algorithm for detecting persistent flooding-based DoS attacks applicable to IoT networks is built by traffic seasonality aware adaptive threshold algorithm. The developed algorithm analyses the fluctuating rate of SYN packet arrival to determine traffic seasonality. It has a higher TPR and is more accurate in detecting SYN flooding attacks. The creation of traffic volume-aware adaptive thresholds helps the suggested technique achieve higher detection accuracy of subtle network attacks. Traffic seasonality learning is very helpful in separating legitimate traffic from attack traffic and reducing the negative effects of attack traffic on the threshold updating process when attack traffic is recognized before increasing the threshold. The developed algorithm can be extended to multiple alternative ways to identify different types of DoS attack, such as Hello flooding attack, SYN flooding attack, Black Hole attack, Jelly Fish attack and so on. This is because of the limits inherent to IoT Network. The future work will focus on specific machine learning and artificial intelligence techniques for modelling traffic seasonality to more accurately set the threshold and for comprehending traffic models of various IoT sub-domains to detect unexpected network behaviour.

List of Publications

1. Sreelekshmi A N, Dr. Nishanth N, "Traffic Seasonality Aware Adaptive Threshold Algorithm for Detection of Flooding Based Denial-of-Service Attacks in IoT Networks", IEEE Systems Journal, Manuscript ID:ISJ-RE-22-14677, Submission Date: 13-Jul-2022, Status: Under Review.

2. Amal Dev S, Shanu J Panicker, Kavya Krishna, Sruthy Krishna N, Sreelekshmi A N, Nishanth N, "Detection of Flooding based Denial-of-Service Attacks in Wireless Ad Hoc Networks based on Statistical Analysis", Arabian Journal of Science and Engineering, Manuscript ID: AJSE-D-22-04738, Submission Date: 28-Jun-2022, Status: Under Review.

References

[1] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). "Internet of Things (IoT): A vision, architectural elements, and future directions". Future generation computer systems, 29(7), 1645–1660.

[2] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac and P. Faruki, "Network Intrusion Detection for IoT Security Based on Learning Techniques," in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2671-2701, thirdquarter 2019, doi: 10.1109/COMST.2019.2896380.

[3] Zargar, S. T., Joshi, J., & Tipper, D. (2013). "A survey of defence mechanisms against distributed denial of service (DDoS) flooding attacks. IEEE Communications Surveys & Tutorials", 15(4), 2046–2069.

[4] Three Types of DDOS Attacks. https://blog.thousandeyes.com three-types-ddosattacks.

[5] N.Nishanth, A.Mujeeb on "Modeling and Detection of Flooding-Based Denial-of-Service Attack in Wireless Ad Hoc Network Using Bayesian Inference", IEEE Systems Journal(vol. 15, Issue:1, March 2021),Pages 17 - 26.

[6] N.Nishanth, A.Mujeeb on "Modeling and Detection of Flooding-Based Denial of Service Attacks in Wireless Ad Hoc Networks Using Uncertain Reasoning", IEEE Transactions on Cognitive Communications and Networking(vol. 7, Issue:3, Sept. 2021), Pages 893 - 904.

[7] T. Wang, L. Qiu, A. K. Sangaiah, A. Liu, M. Z. A. Bhuiyan and Y. Ma,"Edge-Computing-Based Trustworthy Data Collection Model in the Internet of Things," in IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4218-4227, May 2020.

[8] Dan Tang, Jianping Man, Liu Tang, Ye Feng, Qiuwei Yang, "WEDMS: An advanced mean shift clustering algorithm for LDoS attacks detection, Ad Hoc Networks, vol. 102, 2020.

[9] A. A. Adewuyi, H. Cheng, Q. Shi, J. Cao, Á. MacDermott, and X. Wang, "CTRUST: A dynamic trust model for collaborative applications in the Internet of Things," IEEE Internet Things J., vol. 6, no. 3, pp. 5432–5445, Jun. 2019.

[10] A. Sharma, E. S. Pilli, and A. P. Mazumdar, "RRAR: Robust recommendation aggregation using retraining in Internet of Things," in Proc. IEEE 5th World Forum Internet Things (WF-IoT), pp. 76–80, 2019.

[11] Payam Mohammadi and Ali Ghaffari. 2019. Defending Against Flooding Attacks in Mobile Ad-Hoc Networks Based on Statistical Analysis. Wirel. Pers. Commun. 106 , 365–376, 2 May 2019.

[12] R. W. Anwar, A. Zainal, F. Outay, A. Yasar, and S. Iqbal, "BTEM: Belief based trust evaluation mechanism for wireless sensor networks," Future Gener. Comput. Syst., vol. 96, pp. 605–616, Jul. 2019.

[13] J. Zhao, J. Huang, and N. Xiong, "An effective exponential-based trust and reputation evaluation system in wireless sensor networks," IEEE Access, vol. 7, pp. 33859–33869, 2019

[14] Z. Wei, H. Tang, F. R. Yu, M. Wang and P. Mason, "Security Enhancements for Mobile Ad Hoc Networks With Trust Management Using Uncertain Reasoning," in IEEE Transactions on Vehicular Technology, vol. 63, no. 9, pp. 4647-4658, Nov. 2014

[15] V.A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks," Comput. Commun., vol. 29, no. 9, pp. 1433–1442, 2006.

[16] M. Conti, A. Gangwal, and M. S. Gaur, "A comprehensive and effective mechanism for DDoS detection in SDN," in Proc. IEEE 13th Int. Conf. Wireless Mobile Comput., Netw. Commun., 2017, pp. 1–8.

[17] S. Gurung and S. Chauhan, "A novel approach for mitigating route request flooding attack in MANET," Wireless Netw., vol. 24, no. 8, pp. 2899–2914, Nov. 2018.

[18] S. Bhalodiya and K. Vaghela, "Enhanced detection and recovery from flooding attack in MANETs using AODV routing protocol," Int. J. Comput. Appl., vol. 125, no. 4, pp. 10–15, Sep. 2015.

[19] Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy, "Network innovation using openflow: A survey," IEEE communications surveys & tutorials, vol. 16, no. 1, pp. 493-512, 2014.

[20] Junjie Zhang, Xiapu Luo, Roberto Perdisci, Guofei Gu, Wenke Lee and Nick Feamster, "Boosting the scalability of botnet detection using adaptive traffic sampling," in Proc. of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 124-134, March 22-24, 2011.

[21] N. Levy, D. Smith, and J. Schiel, "Operationalizing ISP cooperation during DDoS attacks," in Proc. of North American Network Operators' Group Meeting 71, October 2-4, 2017

[22] Hakem Beitollahi and Geert Deconinck, "Analyzing well-known countermeasures against distributed denial of service attacks," Computer Communications, vol. 35, no. 11, pp. 1312-1332, June, 2012.

[23] Nguyen, Giang-Truong, Van-Quyet Nguyen, Sinh-Ngoc Nguyen and Kyungbaek Kim. "Traffic Seasonality aware Threshold Adjustment for Effective Source-side DoS Attack Detection." KSII Trans. Internet Inf. Syst. 13 (2019).

[24] Ryoichi Kawahara, Tatsuya Mori, Noriaki Kamiyama, Shigeaki Harada and Shoichiro Asano, "A study on detecting network anomalies using sampled flow statistics," in Proc. of 2007 International Symposium on Applications and the Internet Workshops, pp. 81, January 15-19, 2007.

Appendix