M.Sc. Engg. (CSE) Thesis

MITIGATING DDoS ATTACK IN NAMED DATA NETWORK

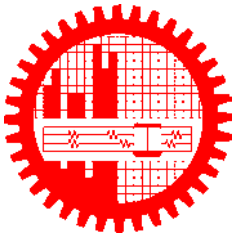
Submitted by

Nadia Akbar Tumpa

0413052087

Supervised by

Dr. Mahmuda Naznin



Submitted to

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

September 2019

Candidate's Declaration

I, do, hereby, certify that the work presented in this thesis, titled, "MITIGATING DDoS ATTACK IN NAMED DATA NETWORK", is the outcome of the investigation and research carried out by me under the supervision of Dr. Mahmuda Naznin, Professor, Department of CSE, BUET.

I also declare that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Nadia Akbar Tumpa

0413052087

The thesis titled "**MITIGATING DDoS ATTACK IN NAMED DATA NETWORK**", submitted by Nadia Akbar Tumpa, Student ID 0413052087, Session April 2013, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents on September 30, 2019.

Board of Examiners

1. Dr. Mahmuda Naznin — Chairman (Supervisor)
   Professor
   Department of CSE, BUET, Dhaka

2. Dr. Md. Mostofa Akbar — Member (Ex-Officio)
   Professor and Head
   Department of CSE, BUET, Dhaka

3. Dr. Shohrab Hossain — Member
   Associate Professor
   Department of CSE, BUET, Dhaka

4. Dr. Muhammad Abdullah Adnan — Member
   Assistant Professor
   Department of CSE, BUET, Dhaka

5. Dr. Lutfa Akter — Member (External)
   Professor
   Department of EEE, BUET, Dhaka

Acknowledgement

I would first like to thank my thesis supervisor Dr. Mahmuda Naznin. The door to Prof. Mahmuda Naznin's office was always open whenever I ran into a trouble spot or had a question about my research or writing. She consistently allowed this paper to be my own work, but steered me in the right the direction whenever she thought I needed it. I would also like to acknowledge my thesis committee, and I am gratefully indebted to them for their valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents and to my spouse for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Lastly, I thank almighty Allah for the opportunities I was given that made it possible to undertake this research.

Dhaka
September 30, 2019

Nadia Akbar Tumpa
0413052087

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Named Data Networking is a new concept in modern internet technology instead of conventional TCP/IP architecture where nodes are addressed based on the interest of the the consumers. Internet security is a very challenging research area due to the huge amount of traffic generation. NDN's built-in structure emphasizes better privacy and security protection which enables more scalable networking but it is also vulnerable to many security threats including denial-of-service (DoS) or distributed DoS (DDoS). DDoS attack can be initiated by various methods, this thesis focuses on mitigation of one special type of DDoS attack, the interest flooding attack with respect to NDN architecture.
Distributed Denial of Service (DDoS) attacks are very prevalent now a days. Our proposed procedure represents the first step towards learning and possible mitigation of DDoS in NDN. We capture the network traffic to classify and learn about legitimate and malicious interests. Moreover, we propose attack detection based on selected features computed by SVM machine learning-based approach. We have simulated our method extensively with ns3-NDN simulator and we have provided experimental results to support the performance of our method.

Chapter 1

Introduction

Emerging technology and advancements are attracting researchers to provide better internet architecture and protocols in terms of security, privacy, scalability, flexibility, performance, resilience to handle the exponential increase in network traffic. Modern internet architecture is going through a major shift of transformation which directs to the emergence of Information Centric Networking (ICN) [3–7]. One of the implementation of modern Internet architecture is supported by Named Data Networking (NDN) technology [8]. *Named data* plays the main role in NDN instead of traditional IP based hosts [9]. However, data oriented interenet protocols suffer from different attacks and security issues. In this thesis, we study security challenges in NDN, we identify the attacks and severity for NDN, and propose a methodology which is basically machine learning based *DDoS Attack* detection and mitigation, aimed at the flowing of legitimate traffic and reducing the malicious traffic that passes through NDN router. We have implemented our mechanism and compared to existing state-of-the-art DDoS detection strategies. We summarize the the contributions of this thesis as follows.

- We design an attack detection model for detection of Interest Flooding Attack (IFA).
- We analyze features that affect IFA and selected the most prominent features using k-means clustering.
- We use machine learning for detection of attack.
- We provide an attack mitigation model.
- We compare the performance of our model with the state of the art approaches. For this we use the special simulator ndnSIM [**?**].

1.1 NDN Architecture

In this section we discuss the powerful hourglass architecture designed for NDN. NDN represents hourglass-shaped architecture at the thin waist it names content instead of communication endpoints Figure 1.1. This thin waist allows lower and upper layer technologies to innovate without unnecessary constraints.

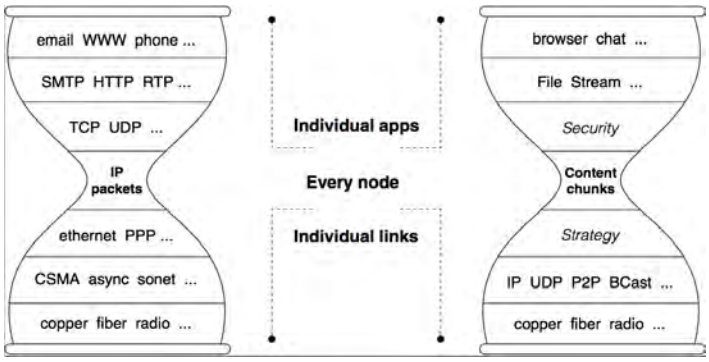


Figure 1.1: Internet and NDN Narrow-Waist Hourglass Architectures [1].

In NDN, communications between hosts works by creating two distinct types of packets. This is an important part of NDN that replaces the IP address identifier that is currently used in TCP/IP. NDN packet types are indicated in Figure 1.2. Any node sending desired *interests* for data acts as a content *consumer*, whereas a node that can provide data packets behaves as a *producer*.

Interest packet: The Interest packet is created at the beginning, which holds the requested content of a consumer. An Interest packet has three sections: *content name, selector* and a *nonce*. *Content name* is the most important part of Interest packet. Content name holds the desired searched name that identifies the wanted content.The *Selector* field uses to refine content searching a series of fields is useful such as order performance, scope etc. The *Nonce* field is used to locate and identify same or equal interest packets.

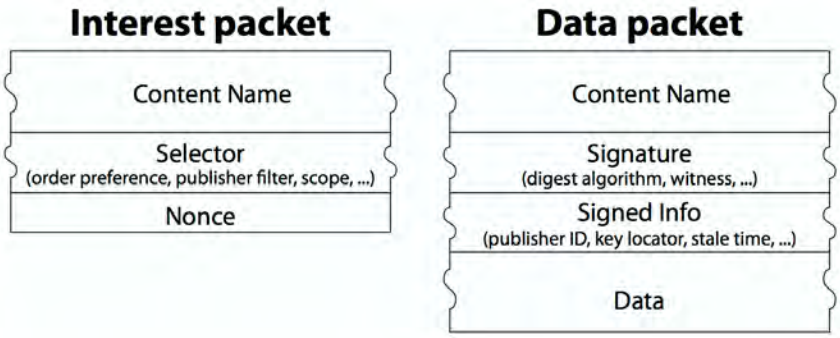


Figure 1.2: NDN Interest Packet and Data Packet [1].

Data packet: It is the most essential component, which acts as the answer carrying the requested content for the consumer. Data packet has four distinct sections: *content name*,

signature, *sign Info* and *data*. Data packet represents the answer of requested interest packet together with its matched name and a signature by the producers key which binds the two. The required data follows the reversed path which was taken by the Interest to go from the requesting consumer. Signed Info holds useful information to verify the accuracy and correctness of the signature.

1.1.1 Naming in NDN

Network architecture of Named Data Network (NDN) is based on the contents name. NDN embraces a hierarchical structure where the content name consists of one or more variable length components, each of which is separated by the "/" symbol and it indicates the boundary between name components. Names are used to identify content and consists of multiple parts: *origin of the content, the file name, the version number of the content, the segmentation number*. A consumer and a provider can always construct the same content name with the help of hierarchical name spaces and naming conventions to fetch desirable content. *Data packets* are immutable and cannot be changed, once it is published under a given name. A new name must be generated for a new version of the same data packet whenever an application needs to published. *Content segmentation* in a sequence of Data packets is supported for large data object [10]. For example : /tutorial.com/python/example/1/3, where tutorial.com is a global routable name, python/example is the object name, the version number is 1 and the segmentation number is 3. Two additional components version number and segmentation number are added to the end of an NDN name.
Furthermore, not all the content names need to be globally unique, it depends on the visibility of the content. *Global uniqueness* is required only those names of a content that are retrieved from globally. For local communication names are based on local context and needed only local routing to find corresponding data.

Packet Routing Once the interest packet is created, it follows a specific path through certain section in a previously configured router and those are the three sections.

- Content Store (CS)
- Pending Interest Table (PIT)
- Forwarding Information Base (FIB)

Content Store

Previously fulfilled Interest packets' content are stored here. Content Store is able to contain cache and are used for content retrieval [11]. Before forwarding each and every packet, NDN

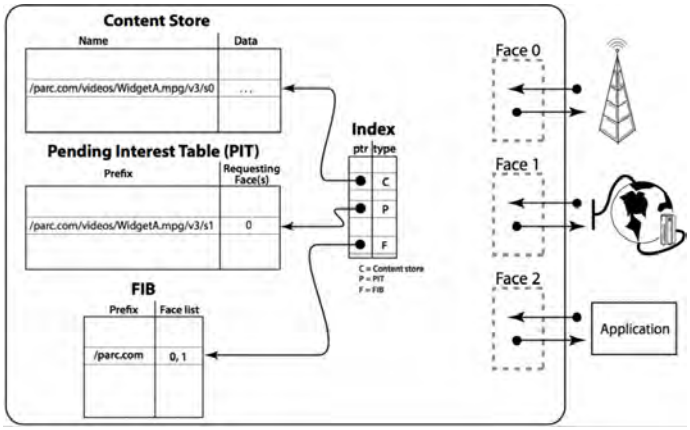


Figure 1.3: Routing path of data packets .

router checks at each hops' CS for locally cached data [12]. If it exactly matches the content name or take the content name as its prefix, the matched data packet forwards back to satisfy the interest.

Pending Interest Table (PIT)

Pending Interest Table contains currently forwarded Interests and waits for matching corresponding incoming interfaces data. PIT entries need to be timed out quickly to maximize the usage of the PIT [13]. PIT table records the Interest's name, incoming interface(s) and outgoing interface(s). For each requested name, a PIT entry is created [14]. If the same forwarded Interest comes, PIT assigns random identifier named nonce and discards the identical Interests.

Forwarding Information Base (FIB)

FIB is populated by a name-based routing protocol which is very similar to the one currently used with TCP/IP, except FIB has name prefixes rather than IP prefixes and it allows for a ranked list of outgoing faces rather than a single best next-hop [15]. Routing table contains name prefixes and corresponding outgoing interfaces. In FIB, using longest prefix match, the Interest name of corresponding Interest is looked up by the forwarder.

Datagram State

NDN router contains "datagram state" of every pending interest packet in its PIT by maintaining an entry. This state lead the way to loop free, two-way symmetric packet flow. Each and every Interest packet pulls one and only data packet thus maintaining one-on-one flow balance.

1.2 Security Challenges in NDN

Security is one of the major concern of NDN design. Content publisher signs every data packet using a particular key to bind securely name to the content. Thus every data packet is publicly authenticated via digital signatures and moves securely from hosts to hosts. This type of verification is useful against different kinds of network attacks but it does not completely cover all security, privacy and trust issues. NDN Data packets do not contain any identity of the sender but this is not sufficient to ensure better privacy because both packets can leak the name of the contents and can result into remarkable privacy concerns.
Although it has built-in security architecture which enables more collaborative, scalable networking for content diffusion but it has several security challenges regarding data confidentiality and access control to protect sensitive user information. To provide a reliable and secure internet architecture, NDN must have capability to fight against current and emerging threats or attacks. These attacks are *Interest Cache Privacy Attack, Interest Flooding Attack (IFA), Cache Pollution Attack, Content Poisoning Attack*, etc. Like current Internet architecture, denial-of-service (DoS) or distributed DoS (DDoS) can happen in NDN. DDoS attack, in particular, Interest flooding attack as well as content/cache poisoning represent the most serious threats and have severe impact on overall network performance. We try to solve this problem using Machine Learning based attack detection.

1.3 Contributions

Our contributions are listed as follows.

- We propose a framework which is capable to mitigate DDoS attack for NDN.
- In detection phase, we learn continuously network traffic data and updating SVM by training. We give better system to detect DDoS attack.
- In mitigation phase, we give a set of rules based on context which are followed by NDN Controller for further action.

The rest of the thesis is organized as follows. In Chapter 2, we discuss the background study of the problem and related research work, research gap and literature review of the existing works in terms of security in NDN. It includes Denial of Service (DDoS) Attack, Interest Flooding Attack etc. We also discuss how to mitigate these attacks. Chapter 3 focuses on the feature extraction, detection of DDoS using SVM classifier related to the problem domain. We also discuss our proposed solution in this chapter. We provide results and analysis in Chapter 4. Finally, Chapter 5 concludes the thesis with a summary discussion and the direction to future development.

Chapter 2

Related Work

In this chapter, we study some of the major research work related to DDoS attack in Named Data Networking.
DoS and *DDoS* attacks create a major security hole in Named Data Networking. These attacks are primary concern in network security till now. The adversary invokes a large number of interest requests that are distributed closely in space, controlling with a large set of zombies. When lots of interest packets are sent at aiming to overflow PITs in routers, preventing them from handling legitimate interests, to swamp the specific content producer(s) because PIT has a certain capacity of receiving interests. This type of attack is known as DDoS attack see. 2.1

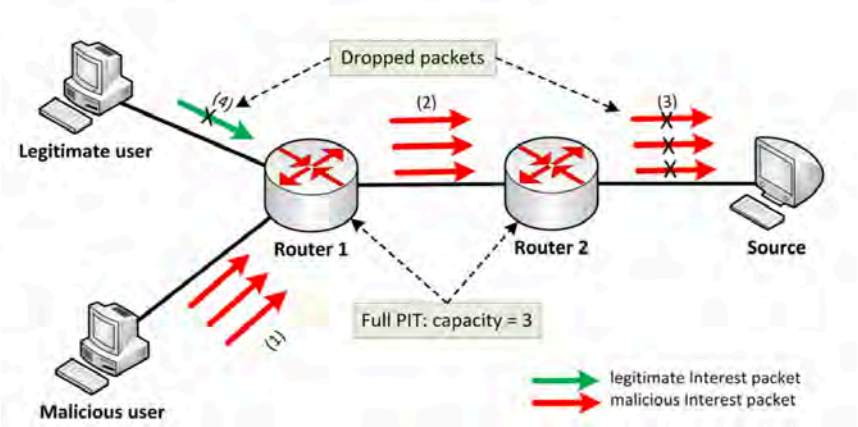


Figure 2.1: Interest Flooding Attack on NDN [2].

2.1 DDoS Attack in NDN

DDoS attack can be generated in the following situations.

- Using compromised systems.
- Numerous *Interest packets* with spoofed names by controlling with a large set of zombies.
- Making bad use of forwarding rules

2.1.1 DDoS Attack Categories

DDoS attacks can be grouped into following categories:

- Single-target DDoS Attack.
- Interest Flooding Attack.
- Content/Cache Poisoning Attack.
- Interest Spoofing Attack.

2.1.2 Single-target DDoS Attack

In this type of attack, *Longest Prefix Match* rule while looking up Interest names in the FIB is used to forward packet into destination [16]. Due to name suffix is forged, content provider does not have any content corresponding to this name to provide. As a result, unsatisfied Interest packet will stay in the PITs and hold memory and computing resources on routers until PIT entries expire. Attacker repeats this process with a large amount of different spoofed suffixes.

2.1.3 Interest Flooding Attack

It is a unique and common DDoS attack for NDN. An attacker can utilize two distinctive NDN features named CS and PIT [17]. To generate large number of Interest packets with full forged names, the adversary uses a large set of zombies, which are possibly geographically distributed and requests existing or non- existing content to overload PIT table in routers [14]. Unmatched Interest packets with FIB should be broadcast [16] or discarded.

2.1.4 Content or Cache Poisoning Attack

Content Poisoning Attack (CPA) is a critical threat where malicious data are filling up by compromised routers or collaboration between consumer and bad providers. CS has

legitimate content names but those data were corrupted or altered and spread to as many users as possible [18]. The main goal of adversary cache misses for honest consumers and to disrupt cache locality by increasing link utilization [19].

2.1.5 Interest Spoofing Attack

An attacker forwards Interest packet with spoofed name consisting a legitimate prefix concatenated with a forged suffix so that the producer cannot provide the data and it remains in the server and exhausts PIT until expired [14].

Spoofed Name = Existing Prefix + Forged Suffix

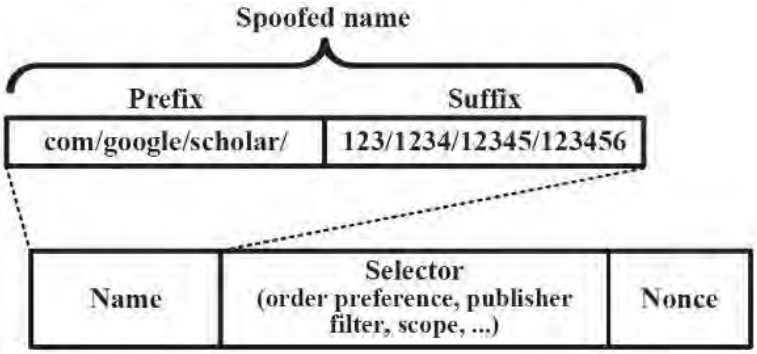


Figure 2.2: An Interest packet with spoofed name: a valid prefix concatenated with a forged suffix

2.2 Interest Flooding Attack Content Types

Gasti et al. observed there are three types of Interest flooding attack based on whether the requested content exists and how the content produced, then identified [11]:

- Existing and static
- Dynamically generated
- Nonexistent (i.e. unsatisfiable interests)

Existing or Static Content:
The impact of this type of attack is restricted because of NDN in-network content caching mechanism provides a built-in countermeasure by automatically block subsequent similar interest. Following requests are satisfied by cached copies and thus subsequent request can not propagate to the producer(s).

Dynamically Generated Content:
This type of attack uses the PIT states. The adversary, using fake requests, floods the specific content producer(s) and fills the PIT space. The adversarys origin is very difficult to find [20]. The impact on NDN routers relies their distance from the targeted content producer(s). The closer to the producer, the greater effect on its PIT.

Non-existent Content:
NDN infrastructure is resilient to this type of contents, though network and application-layer functionalities suffer [20]. Attackers generate distinct, unique and unjustifiable Interests for a non-existent content so that names cannot match any FIB entry and propagate to other NDN nodes until they reach the hosts at the edge of the network. These Interest packets remain in the PIT up to the expire time. If attacker duplicate such a great amount of Interests, computational cost will increase.

2.3 Background Work

As discussed by Gasti et al. in [11] discussed DoS and DDoS for the first time. They have Proposed two type of countermeasures:
1. *Router Statistics*: Statistical approach is suggested for the detection of the attack. It is used to maintain balance between Interests and Contents by limit flow rate.If the routers receive too many request from the same domain, they can drop some of them.
2.*Push-back Mechanisms*: The router applies a push-back mechanisms to trace back for malicious source detection and separate the attack right at source.

Limitations of this work:

- The authors only discussed about attack and its possible countermeasures.
- The strategy of dropping interests, may affect popular content as well.

Atanasyev et al. [21] proposed three mitigation algorithms by interest spoofing of interest flooding.
1.*Token bucket with per-interface fairness*: represents a fair mixture of Interests, received from neighboring nodes forwarded by a router on each interface.
2.*Satisfaction-based Interest acceptance*: calculates Interest satisfaction ratio, the ratio between the forwarded Interest requests to number of satisfied requests, for accepting legitimate(forwarding) or rejecting an incoming malicious Interest.
3.*Satisfaction-based push back*: where the value of incoming Interest limit is set by each router, directly depends on the proportion to the calculated Interest satisfaction ratio.

Besides, routers have to announce this limit to their downstream neighbors, thus in accordance with the announced limit of the upstream node, downstream routers can reconfigure their own Interest acceptance limits. Satisfaction based pushback algorithm effectively detect and mitigate the attack and make sure that all the interests are form a legitimate user.

Limitations of this work are as follows.

- All of three algorithms utilize their respective information to stop Interest flooding attacks. Some improvements was shown in token bucket with per-interface fairness. Among three algorithms, satisfaction-based Interest acceptance was less effective.
- Based on probability applied a filter on the malicious interface, legitimate interest also gone thorough this filter, thus suffer.

Compagno et al. [22,23] suggested a defense mechanism named Poseidon. It is a framework strictly used for detection and mitigation of distributed and local Interest flooding attack for non-existing contents.
A set of algorithms named Attack Detection and Message Processing runs on the routers continuously monitoring the overall traffic.
1.*Attack Detection* uses two parameters, the first one is the ratio of incoming interest packets and outgoing content packets and the second parameter is PIT space used per interface within time interval. If both of the parameters exceeds a predefined threshold value, then the attack is detected on that specific interface.
2.*Message Processing* algorithm sends an alert message back to the interface(s) of the attack. A filter is set on the annoyed interface(s) decreases the threshold value of the above parameters. In the most testing cases during the attacks, authors simulated model has been able to use around 80 - 90 percentage of the available bandwidth.

Limitations of this work are as follows.

- Static Threshold value is used here for evaluating ISR and PIT usage for attack detection.
- Each router applied a filter thus valid legal requests also suffer from this attack.

Choi et al. [24] addressed an overview of threats for strictly non-existent contents on NDN for Interest flooding attack. Authors simulated and explained how the Interest flooding DDoS attacks effect, deteriorate on the overall quality of services and the difficulty of getting a solution for flooding attacks in the PIT.

Limitations of this work are as follows.

- Only analyzing DDoS attacks and their countermeasures.

Karami et al. [17] introduced an intelligent hybrid algorithm for mitigating Interest flooding attack in NDN. A two-phase framework has been proposed as follows.

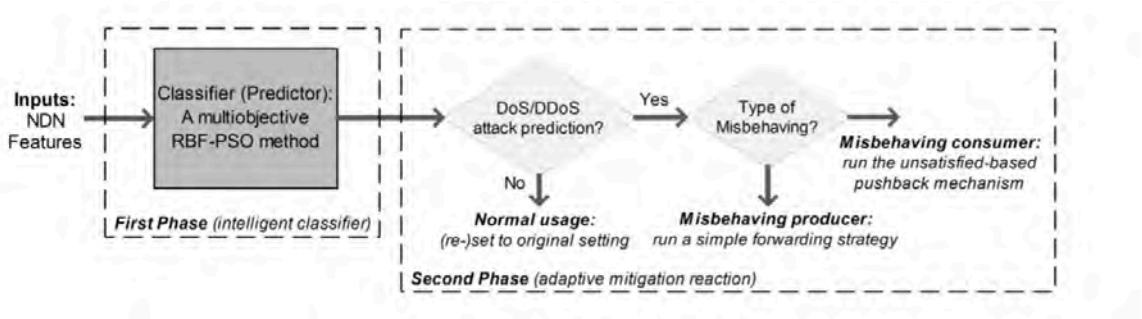


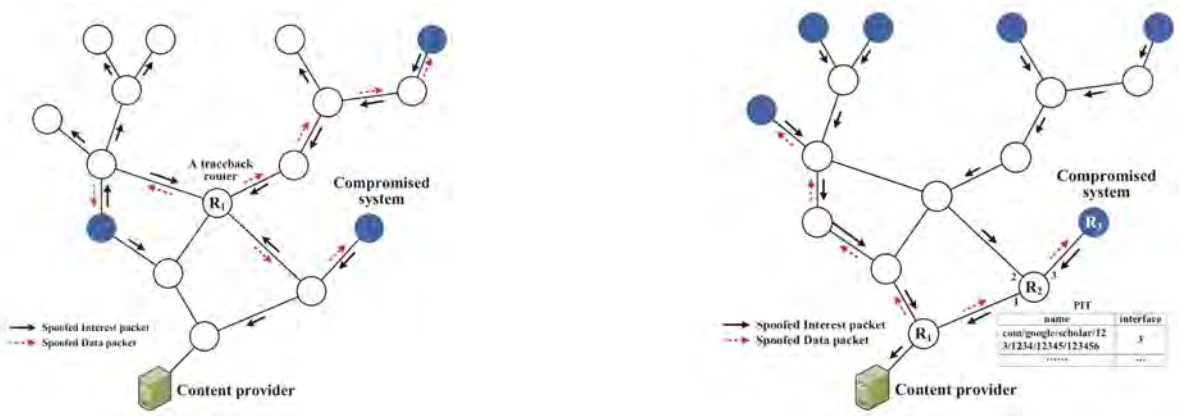
Figure 2.3: The overview of the proposed Proactive detection and Adaptive reaction.

1. *Proactive detection*: In order to improve the accuracy of DoS attack prediction, using a combination of multi objective evolutionary optimization algorithm and Radial Basis Function (RBF) neural network can provide solution.
2. *Adaptive reaction*: In the reaction phases, an adaptive mechanism by enforcing explicit limitations against adversaries is used to mitigate the attacks.

Limitations of this paper are as follows.

- Applies a hybrid approach and only investigates inter-domain DoS attacks.
- To detect an attack, features are used, but they have not done any feature analysis of detection parameters.

Dai et al. [16] proposed **Interest traceback** solution generating a spoof data packet to satisfy the long-unsatisfied interest in the PIT to trace the Interest originators. The collaboration of the router and the content producer is the basis of this solution.



(a) Compromised systems send out attacking Interest packets.

(b) Three Interest traceback paths to trace the attacker.

Figure 2.4: Mitigate DDoS Attacks in NDN by Interest Traceback.

Limitations of this work are as follows.

- To detect the attack, a threshold value is used as a metric. If the threshold goes beyond predefined value, the traceback algorithm is applied.
- This procedure does not work for attack when Interest uses a random prefix
- According to [17], the solution is not proactive. Moreover makes overhead in the network by increasing made spoofed contents for the interest depleting the bandwidth of the network and creating traffic.
- Another limitations of this process is that it accepts long-unsatisfied Interests in the PIT as adversary interest and other unsatisfied Interest are legitimate usages. So as a result any long incoming interest packet which may be a legitimate interest might drop and thus valid interest forwarded decreases from that interface.

Chapter 3

Problem Domain

In this chapter, we give the details of the problem formulation and the outline of proposed solution.
In a conventional network scenario, there is a consistency in overall traffic. But if there is any deviation, different features appear in the network traffic which can help to detect the anomaly. In the proposed system model, we learn learn traffic behavior and from there we detect the anomaly. There are several parameters which can be monitored by a router to determine any probable attack. We aim to detect *Distributed Denial of Service (DDoS)* attacks within NDN by using a Support Vector Machine (SVM) for classifying network traffic as *normal* or *anomalous*. For learning the traffic behavior we use the k-means clustering and single-class SVM classifier algorithm. If there is a significant deviation from the learned behavior of the traffic then that can be considered as *DDoS attack*.
DDoS attack can be generated by different ways. But here, we focus on only one major type of DDoS attack, *Interest Flooding Attack* (*IFA*).

3.1 Preliminaries

We now provide some preliminaries and definitions.

- *InDataPackets*: It is the number of arrived data packets in a router.
- *InInterestPackets*: It is the number of arrived Interests packets in a router.
- *Interest Satistaction Ratio (ISR)*: It is the ratio of the number of data packets received by the consumer node to the total number of interest packets sent.
- *In-ISR*: It is the number of satisfied interest packets where interface was part of the incoming set.

- *Out-ISR*: It is the number of satisfied interest packets where interface was part of the outgoing set.
- *InData*: It is the number of arrival data packets to an interface.
- *OutData*: It is the number of sent data packets from a router.
- *PIT*: Pending Interest Table which contains information about pending interests.
- *PIT Usage*: It is the number of PIT entries on router.
- *NACK*: NACK provides negative acknowledgement.

We now talk about some assumptions.

- The attacks only exploit PIT.
- Interests can both be satisfactory and unsatisfactory.
- If PIT is full, all subsequent interests are dropped.
- PIT keeps unsatisfied interests.
- The adversary requests dynamically generate contents since NDN routers can store static contents. Thus malicious interests do not reach the producers for the static ones.

3.1.1 Goal of the Attackers

NDN routers are very different from traditional routers, and it has CS and PIT which are unique in maintaining data structures used to store state. There should be process and technique to protect them, otherwise adversary misused them to implement DoS or DDoS attacks. The goals of the adversary are-

- The consumption of routers' resources: to jam PIT table in routers, preventing them from handling legitimate Interests.
- To overburden with the requests on the target content producers: Interest requests for existing contents, non existing contents or those require more computation from the providers.
- To create network service disruptions and congestion: Asymmetry in size between Interests packets and Data packets creates the saturation of network links.
- Bandwidth depletion: The maximum rate of malicious interests does not depend on the bandwidth allocated by the victim to content packets or ability to receive content.

3.2 Problem Formulation

PIT holds the information about pending interests. If the PIT is completely full, then all incoming interests are dropped and cannot accept new interests, until the pending ones are either *satisfied* or *expired*. Flooding to a router occurs by the adversaries to saturate the PIT with many distinct interests. Under this attack, incoming interests is higher than the rate of packet expiration or content satisfaction. In interest flooding attack, two scenarios may arise. These are stated as follows.

- The adversary produces interests with spoofed names.
- The adversary sends interests for existing content.

3.2.1 Interest with Spoofed Name

In the first type of attack [21], the attacker sends spoofed named interest packets created by existing prefix and forged suffix with a random string. These packets create entries in PITs of routers along with the path. Since there is no data producer that has produced data packet corresponding to interest packets, these packets are dropped by the publisher. Entries of PIT remains until expired.

For more accuracy and faster detection, we should analyse more feature parameters that give correct and precise results. Here, we use six feature parameters as a countermeasure for this scenario.

3.2.2 Interest with Random String

In the second type of attack [11], malicious nodes produce random strings as interest packets to perform the attack thus exhausting producers' maximum capacity. These interest packets have no FIB entry, thus interests are broadcast from all the interfaces and seriously affect network rather than a particular server. To mitigate this type of attack, a *negative acknowledgment*(NACK) can be sent to the router from which the interest packet is received and it can be decided not to forward interest packet to other routers [22]. Tough NACK is used but this has severe effects on network. NDN network performance may deteriorate due to the possibility of FIB corresponding to the interest packets can be found after multiple hops.

Thus it is necessary to continuously learn network traffic and create a model for attack detection.

3.3 Our Approach

The proposed method consists of *offline* and *online* phases. A new method is used for classification of network traffic. Support Vector Machine (SVM) is the most popular method among machine learning approaches regarding to classication and regression.
SVM can be efficient in identifying malicious data from network traffic, but this algorithm is memory and process intensive. Training such algorithm for each and every Interests is very expensive in terms of resources of routers. Therefore, our approach was to first cluster the Interests based on the features using unsupervised learning algorithm, such as k-means, and then applying SVM on the clusters to decide their boundaries [25]. Comparing to SVM, k-means algorithm is faster and consumes fewer resources. Generally router has less memory and less processing power, so this will be better choice for routers.

3.4 Machine Learning

According to [26], it is a computer program which is said to learn from an experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P improves with experience. Machine learning algorithms solve problems from many domains without detailed domain-specific knowledge and builds a hypothesis using a training set as an input.
Let $f : S^d \rightarrow S^d$ be the function that we need to guess from input vectors $x^1, x^2, ..., x^n$, also called as input variables or feature vectors. Let Ξ be a set of such input vectors. Let n be the number of input vectors in training set Ξ. Let G be any set of functions. Let $g : S^d \rightarrow S^d \in G$ be the hypothesis about function $f \in G$. We select g based on training set $X \subseteq \Xi$, of m input vectors. In supervised learning, we know the values of f for m samples, in training set X. We assume, if we can find hypothesis g that closely agrees with f for the members of X, then this hypothesis will be a good guess for f when X is large. In unsupervised learning, we simply have a training set of vectors without function values for them. The problem in this case, typically, is to partition the training set into subsets, $X_1, .., X_N$, in some appropriate way [27].
Most common categorization of machine learning algorithms are Supervised and Unsupervised learning. K-means algorithm is an unsupervised learning algorithm and SVM is a supervised algorithm.

3.4.1 Feature Scaling

Feature scaling is a method used for pre-processing of data to normalize the range of independent variables or features of data in a fixed range. Feature scaling is generally

performed by subtracting the mean and scaling the feature to a unit variance value. Feature scaling helps to bring all values to same magnitudes and speeding up the calculations in an algorithm.
Standardisation equation is: $x' = \frac{x-\bar{x}}{\sigma}$
where x represents a feature vector, $\bar{x}$ is the mean and σ is its standard deviation.
For example, consider two vectors (5, 3, 1000) and (4, 3, 500). The Euclidean distance between these two vectors using formula $\sqrt{\sum_{i=1}^{n}(m_i - n_i)^2}$, using the formula we can calculate the distance is $\sqrt{(5-4)^2 + (3-3)^2 + (1000-500)^2}$. A training set example as a vector, then feature will be each coordinate in the vector. Mean and standard deviation was calculated, to standardize the vector for the set of input vectors. By subtracting the mean from each feature vector a new vector was created and feature vector is dividing by its standard deviation.

3.4.2 Clustering

Cluster analysis is a technique of grouping a set of training vectors as input in such a way so that objects in the same group are more similar to each other than to those in other groups. For creating clusters, k-means [28] clustering is one of the most efficient algorithms. This algorithm groups elements in training set based on how close they are from the centroid. Taking any k randomly chosen points as centroid $\mu_1, \mu_2, ..., \mu_k$, from training set X = $x^1, x^2, ..., x^m$, $x^i \in S^d$, i = 1, 2, ..m. After every cluster assignment the centroids are recomputed. Lloyds algorithm is the most popular heuristic algorithm for k-means clustering. Algorithm 3.6.1 represents the k-means clustering, where input data will be matrix of points with dimensions m and the desired number of clusters k.
An arithmetic mean of all the points in a cluster is called centroid vector. To determine the number of clusters and centroids, *Elbow* method in Figure 3.1 is used. It can find the optimal value of k in our training set. The elbow method checks the portion of the variance explained by a function and plots the value of the cost function produced by different values of k.

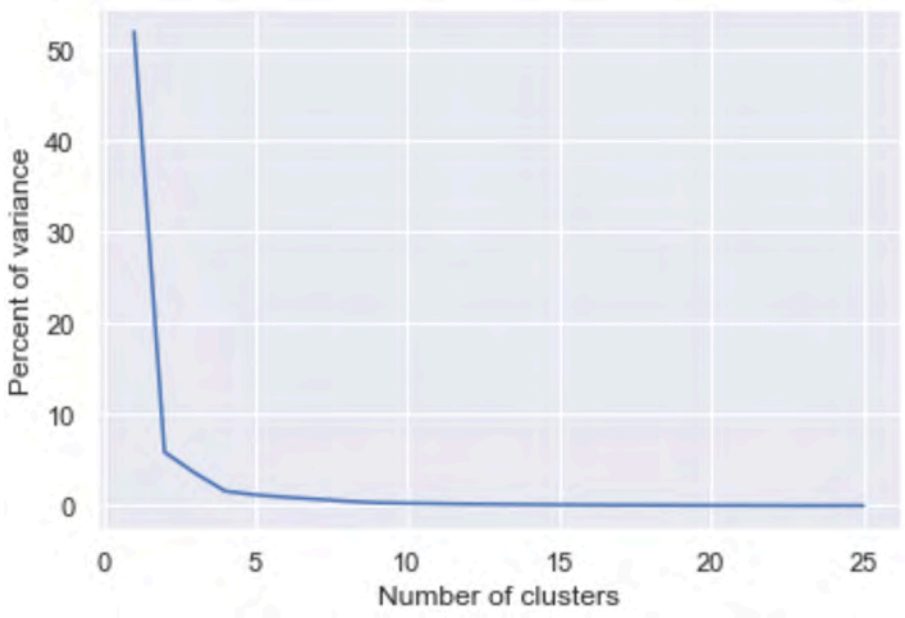


Figure 3.1: Elbow method to find optimal cluster.

Using Elbow method, the following diagram is produced. In this diagram, x-axis represents the number of clusters of k in a training set and y-axis represents the variance explained.
Here, we use an improvement of k-means which is named as k-means++ algorithm. An arbitrarily initialization step is replaced by randomized, simple seeding technique. *k*-means++ always tries to search and find the centroids as far away from each other. If E(x) is the shortest distance from a data point $x \in X$ to the closest centroid we have already chosen. Then we follow *k*-means++ algorithm [29].

3.4.3 Support Vector Machine

Support Vector Machine (SVM) [30] is a learning model with associated learning algorithms defined by a separating hyperplane, involves plotting of data as points in an n-dimensional space, where n denotes the number of features. Support Vector Machines are a set of associated supervised learning methods that inspect, analyse data and identify patterns. It is used for machine learning classification and regression analysis. Identifying the right hyperplane which differentiates the two classes is the key to design an efficient system. For defining the hyperplane, there are cases called as the Support Vectors. SVM tries to maximize the margins by finding an appropriate hyperplane so that it clearly separates the cases into two non-overlapping classes. Figure 3.2 shows the Support Vector Machine.
SVM model and support vector regression can be used to avoid the difficulties of using linear functions in the high-dimensional feature space and the goal is to predict which category a particular subject or individual belongs to, based on training set examples.

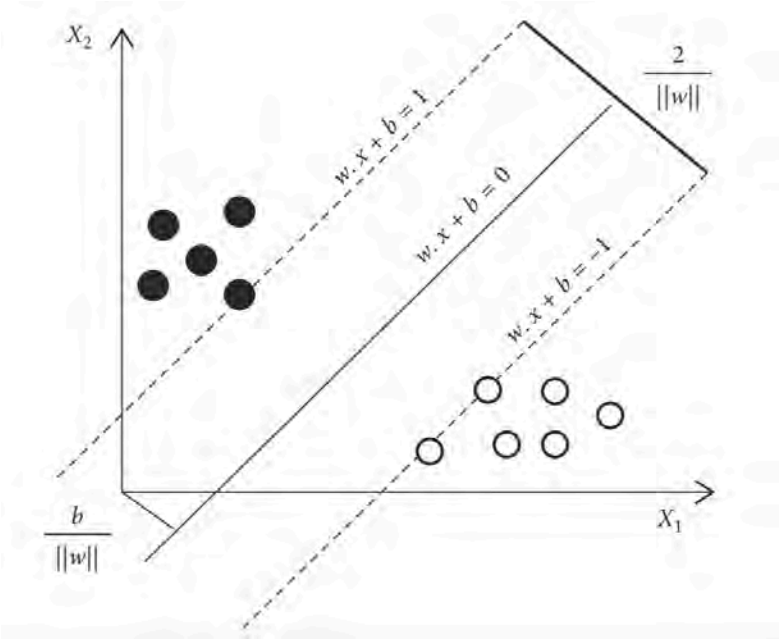


Figure 3.2: Support Vector Machine.

SVM tries to classify training examples into two classes and classification is based on the labels of a training set. Consider training set distance from different point $x^1, y^1, ..., x^n, y^n$, where $x^i \in S^d$ is the training set example and $y^i \in -1, +1$ is the label for x^i.

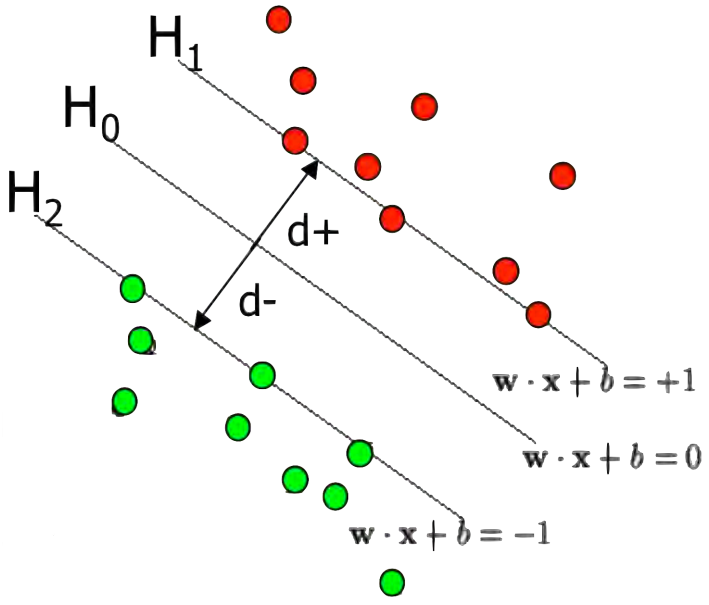


Figure 3.3: Linear Separation using Support Vector Machine(SVM).

To generate the optimization algorithm in such a way that only the support vectors determine the weights and thus the boundary. Input vectors that just touch the boundary of the margin are tips of the vectors. The decision surface separating the classes is a hyperplane of the form of equation:

$w^T x + b$

where, w is a weight vector, x is input vector, b is bias.

Margin of Separation d is called the separation between the hyperplane and the nearest data point for a given weight vector w and bias b. Optimal Hyperplane also named as maximal margin is called the particular hyperplane for which the margin of separation d is maximized.

The points on the planes H_1 and H_2 are the tips of the Support Vectors. H_1 and H_2 are the planes form of equation:

$w^T x + b \geq 0 for d_i = +1$

$w^T x + b < 0 for d_i = -1$

where,

d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin of a separating hyperplane is $(d+) + (d-)$

The plane H_0 is the median in between H_1 and H_2, the equation:

$w^T x + b = 0$, Figure 3.2 and Figure 3.3 shows the H_0, H_1 and H_2 with equation and details of svm.

we are using SVM classifier to differentiate or separate the attack traffic data from the normal traffic, as big a margin as possible using a line or a curve.

3.4.4 Data Structures

We know that we use three table structures: Content Store(CS), Forwarding Information Base(FIB), Pending Interest Table (PIT) and a new file that run ML based Detection script, which is maintained by every router for learning and detecting the attack.

Offline Phase for ML Based Model:

In offline phase, creating Data Collection and learning from the traffic flows are discussed. Learning behavior from each traffic at the router can form the basis of analysis in this thesis. By analyzing traffic flow of a router, we can detect if there is any suspicious behavior in the network traffic, and thus router mainly plays as an important role as a point of analyzer. An Internet traffic data will be captured from network on a router. Once we have a flow information, we can apply learning techniques iteratively on each flow to gain deeper knowledge about normal behavior of the traffic flow.

Network Data Collection Module

Normal traffic data as well as attack traffic data are collected on different time frames using the the ns3-based ndnSIM simulator [31] as shown in Figure 3.7. Generated Traffic is collected in the file after different time interval of simulation time.

There are different network parameters that routers can monitor to determine whether they are attacked by IFA or not, such as PIT Usage, Dropped Interests, ISR (Interest Satisfaction Ratio), PIT Size, Incoming Interests etc. We select the most prominent parameters in the feature selection module.

Content	In Interest Count	Out Interest Count	Served Interests	Dropped Interests	ISR	PIT Usage
A	10000	10000	10000	0	100	200
B	10000	10000	5630	4370	56.3	331
C	10000	10000	5887	4113	58.87	3321
D	1000	1000	789	211	78.87	194
E	1000	1000	763	237	76.30	203
F	1000	1000	1000	0	100	20
G	500	500	444	56	88.87	65
H	500	500	500	0	100	10
I	500	500	432	68	86.3	74

Table 3.1: Reduced sample data used for the classification.

Feature Extraction for Clustering

For learning the network traffic behavior we use k-means clustering and we have used standardized vectors as input for k-means on reduced sample data Table 3.1.
Elbow method has been used to find the optimal number of clusters in our sample training set. This method checks the portion of the variance of the number of clusters explained by a function. The extracted features will be used for training a learning algorithm. A router will constantly keep learning from the traffic and updating the learned parameters and save it on file.

In Interest Count	PIT Usages	Dropped Interests	ISR	Cluster
10000	200	0	100	0
10000	331	4370	56.3	2
10000	3321	4113	58.87	3
1000	194	211	78.87	1
1000	203	237	76.3	2
1000	20	0	100	0
500	65	56	88.87	1
500	10	0	100	0
500	74	68	86.3	3

Table 3.2: Labeled Training Set (with cluster number).

We have used Scikit-learn libraries, for elbow and k-means clustering. Scikit-learn is the most famous, open-source, rich machine learning software library for Python programming language.

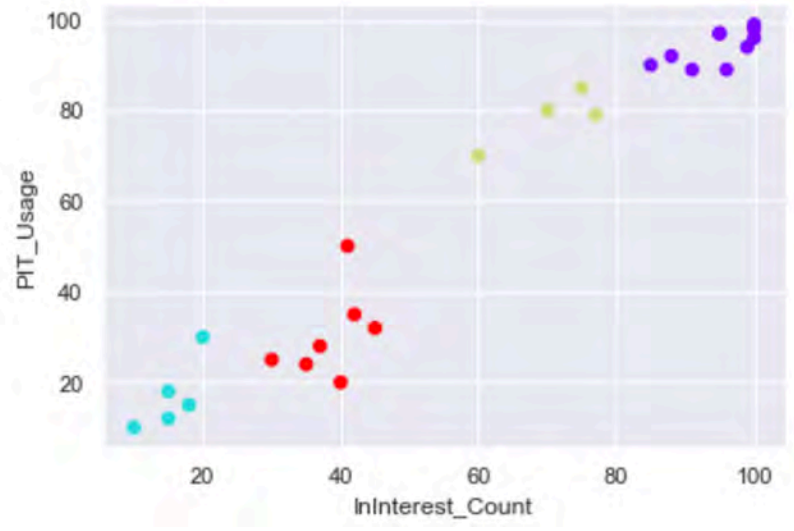


Figure 3.4: Cluster Formation using k-means

3.5 SVM Training and ML Construction

SVM learns using cluster parameters and it can be used to form hyper planes in a high dimensional space to classify incoming traffic as normal or malicious. On the file system, the interest packets captured during a normal traffic and attack traffic were saved. SVM model is trained by using this characteristic from the file system. Legitimate interests flow information is used for training purpose and attack traffic is used for attack detection purpose. Both files (attack and normal) are used to generate test features for SVM [32].

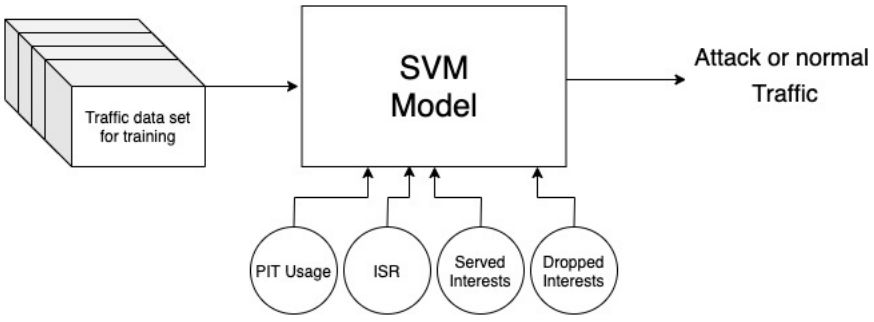
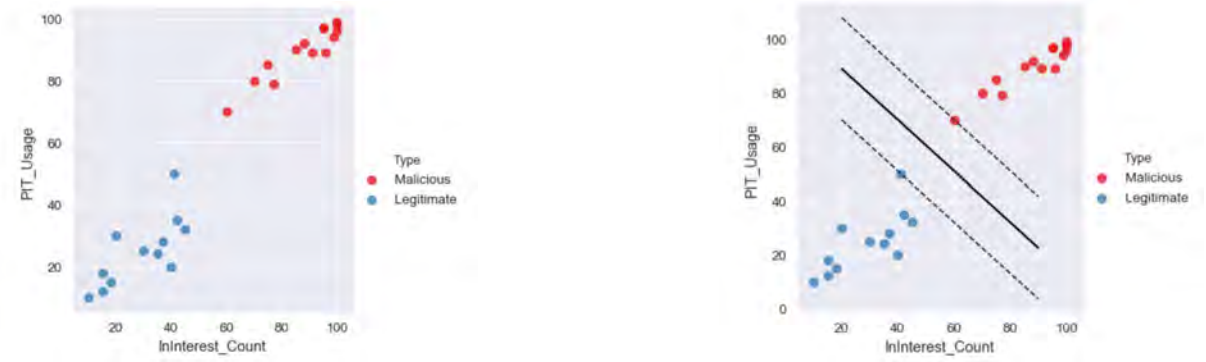


Figure 3.5: Traffic data classification using SVM trained model.

Our algorithms were trained using multiple training data sets and after that a trained model was created. That model was stored on file system and refresh it every time after new information was learned along with previous learned information.
We have used here Scikit-learn libraries, for our DDoS ML based attack detection classification and model construction program.



(a) SVM Classification (b) SVM Model

3.5.1 Online Phase for Traffic Classification

This section provides the details of how detection and mitigation steps are integrated into an overall system, while maintaining routes for normal traffic and blocking attack traffic in a network.

In online phase, ML based Attack Detector is added to every router that classifies the network traffic operation. Figure 3.7 shows learning phase and detection phase. For detection purpose we need to test traffic data into SVM model.

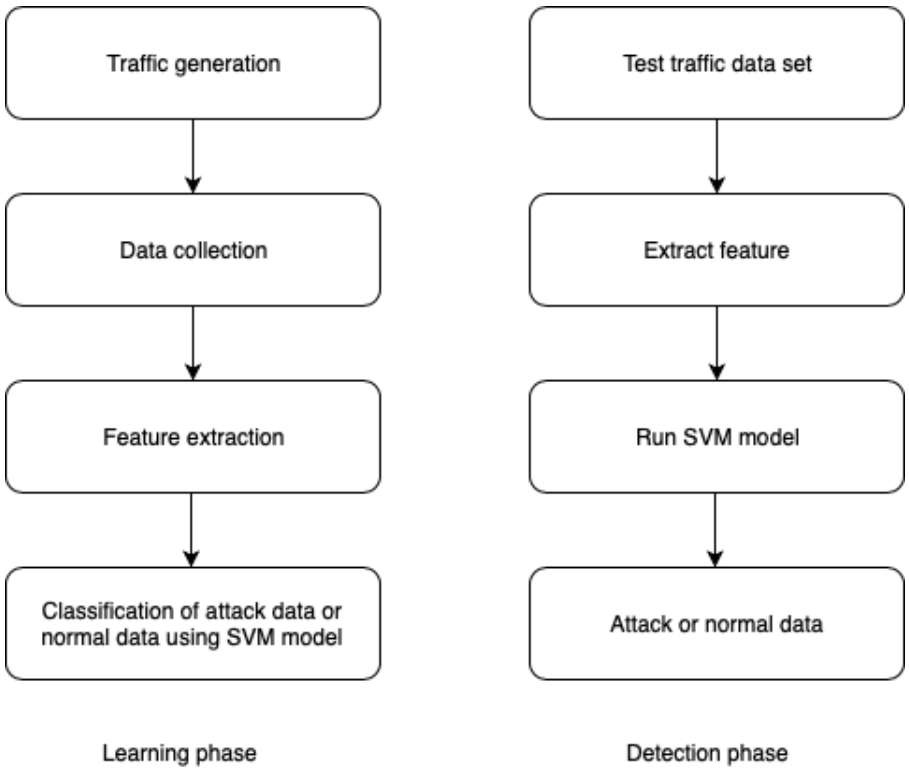


Figure 3.7: Traffic data learning and detection phase.

Data collection (Learning Phase) related to the behaviour of malicious or legitimate user within period of time and then applies machine learning algorithm to the collected data to

determine whether that request is legitimate or not. The advantage of using this technique is that we can easily update SVM model on file system without modifying the existing rules.

Our ML based Attack Detector script runs on every router on the network for attack detection. After detecting an attack a set of rules can be generated. Set of rules plays the major role for further action, that can block all the packets or communicate with destination for sharing the nature of the attacker or store temporarily for further checking and etc. Due to continuous learning, growing and in-depth inspection of traffic flow, if the packets are found to be false positives, they should be routed back to their respective destinations.

3.6 Steps in Details

- We deeply analyse the packet contents, captured Internet traffic at the router during a fixed time interval.
- We maintain a file system for legitimate and malicious traffic data and create training sets.
- We update file system with the new flow traffic data.
- In the case of a DDoS attack, traffic flagged as malicious by SVM is dropped if the packet does not match with the cluster label it was labeled during training.
- Before using reduced data to a learning algorithm, first we had to feature scaling (also known as Standardizing). For feature scaling, we need subtracting the mean and scaling the feature to a unit variance value. It is important because different features at different scales can cause one feature to dominate others in the algorithms output result. Mean and standard deviation can be calculated, to standardize the vector. Then subtracting the mean from each feature vector and dividing that feature vector by its standard deviation creates a new vector.
- We use k-means to extract the features by using that vector at an interval of every sampling period (340 seconds) at network during a fixed time window. Then we pass the features already to trained SVM classifier where the classifier classifies the features set as normal behaviour or malicious behaviour from the network.

3.6.1 Algorithms in Details

We provide the flowchart.

Algorithm 1 Basic k-means algorithm

1: Select any random k points as the initial centroids, where k ∈ M
2: **repeat**
3: To all the centroids of the training set calculate distance for each element.
4: From k clusters, assign centroid which is closest to it.
5: Recalculate the new centroid of each cluster by taking the mean. If $m^{j1}, m^{j2}....m^{jn}$ are the elements of the cluster j, then for cluster j new centroid will be $\mu_j = \frac{1}{n}[m^{j1} + m^{j2} + ...m^{jn}]$, where n is the number of points assigned to cluster j. Recompute this for all the clusters.
6: **until** The centroids do not change

Algorithm 2 k-means++ algorithm

1: Select a centroid μ_1, choose uniformly at random from X points as first centroid.
2: Choose a new centroid $\mu \in X$, such that the probability $\frac{E(\mu_i)^2}{\sum_{x \in X} E(x)^2}$ is highest.
3: Repeat Step 2 unless all k centroids are taken.
4: Continue with the Lloyds k-means algorithm, skipping random initialization stage.

Algorithm 3 ML based Attack Detector for NDN network attacks

Require: ML based Attack Detector
Ensure: Legitimate or malicious traffic detection
1: Chose the Machine Learning Algorithm for Clustering
2: Learned Traffic behaviour from network
3: Chose the Machine Learning Algorithm for Classification
4: Train the ML algorithm using the Learned Information
5: **if** The trained model predicts an attack on a host router **then**
6: Update the NDN set rules to block that types of traffic
7: **else**
8: Allow the interest data to access the resources
9: **end if**

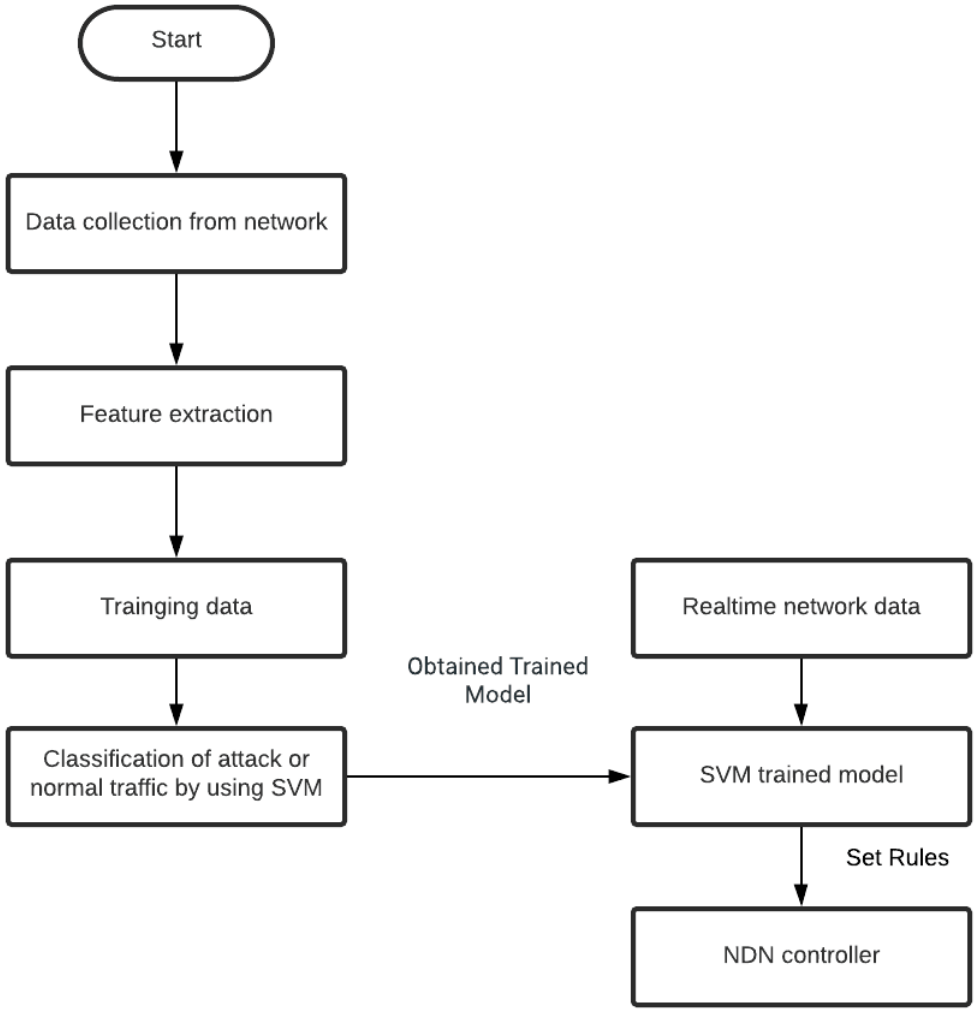


Figure 3.8: Machine Learning based architecture for defining security rules on NDN

3.7 Cluster Evaluation

To ensure the accuracy of our system, cluster parameters are needed to be tested. Evaluation of cluster can be measured using the confusion matrix. Where,

- *TP* - Network traffic attack type and classified as an attack.
- *FP* - Network traffic attack type and classified as an not an attack.
- *FN*- Network traffic not attack type and classified as an attack.
- *TN*- Network traffic not attack type and classified as not an attack.

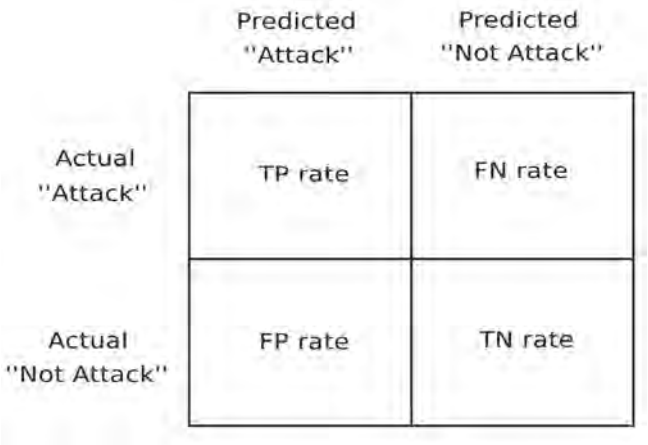


Figure 3.9: Confusion Matrix.

We report our results using confusion matrix shown in Figure 3.9.

Clustering information reveals the normal or attack traffic flow. Malicious Interest or Legitimate Interest should be found on same cluster as was found during the training set. If an Interest found on multiple training sets, then maximum count of a specific type of training set will be considered on that case. If an interest is found on three attack training sets and two non attack training sets then it will be an attack type of data.

Rand Index(RI) [33] was used for measuring this similarity. How well does the test clustering matches with the trained model can be determine Using RI:

$RI = \frac{TP+TN}{TP+FP+FN+TN}$

In Interest Count	Cluster	RI
10000	0	1
10000	2	0.75
10000	3	0.75
1000	1	0.25
1000	2	0.50
1000	0	1
500	1	0.25
500	0	1
500	3	0.75

Table 3.3: Learned information after clustering.

3.8 SVM Model Evaluation

Using different Data set we can evaluate our Trained Svm Model. SVM classifier generates one sub model for every sample data set using SVM classification and produce results as given in different tables in coming sections.

3.8.1 Data Set

Without any duplicate instances in any collected traffic data, the data set should be separated by a random selection of instances into three groups to make a good data set for evaluation:

- *Training set*
- *Cross-validation set*
- *Testing set*

Dataset	Original	Training	Cross	Test
Split %	100%	70%	15%	15%
Ex. Split of data	A,B,C,D,E,F,G,H,I	A,B,E,G,H	C,F,H	D,B

Table 3.4: Breaking up of collected data set into three.

Confusion matrix is shown in Table 3.5 which is the main basis for checking credibility, accuracy and efficiency of the proposed ML Based Detection model.

Dataset	Attack Data(d)	Normal(Data(n)	Outcome Cluster
Training	5000	22	M
Training	38	1000	L
Cross validation	500	11	M
Cross validation	3	500	L
Testing	1000	11	M
Testing	2	500	L

Table 3.5: Confusion Matrix for SVM Model.

Table 3.6 represents results of SVM classifier model for training, cross-validation and testing data set in terms of Data type, Mean, Standard deviation, TP rate, and FP rate for all selected attributes.

Dataset	Data Type	Mean	Standard dev.	TP rate	FP rate
Training	M	0.12	26.32	77.31	0.94
Training	L	8.75	753.77	78.21	1.48
Cross	M	0.47	10.94	59.42	1.35
Cross	L	0.17	5.31	67.91	1.86
Testing	M	1535.28	257.70	82.37	2.46
Testing	L	0.45	3.18	84.03	2.12

Table 3.6: Result of attack detection by ML Based Detection.

Chapter 4

Results and Analysis

In this chapter, we provide the experimental results of our proposed model. We assess DDoS attack's vulnerability and provide the results received from simulation study. We use ndnSIM simulator [31] to evaluate the performance of our proposed algorithm.

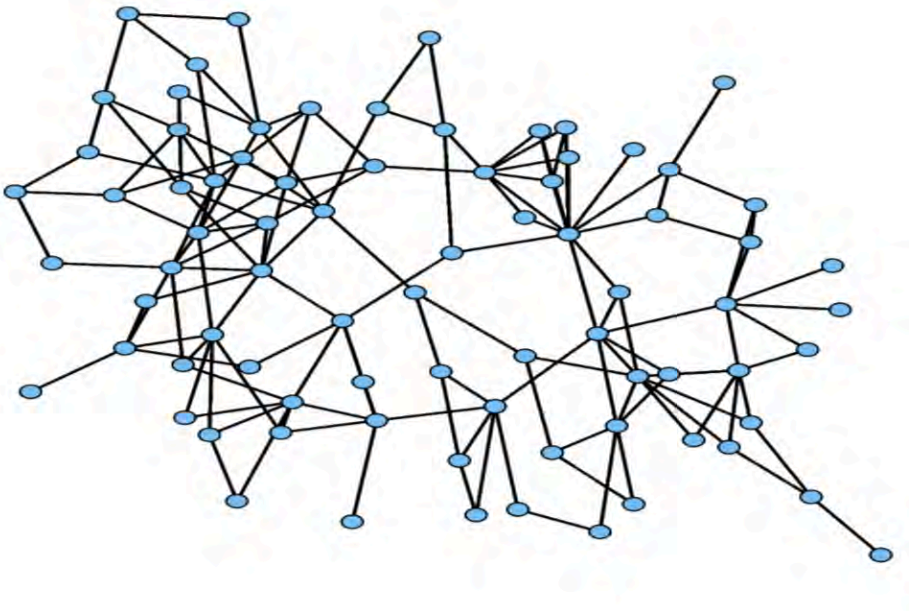
4.1 Testbed

This section represents experimental setup in which we have implemented our algorithm and run simulation. We compared the performance with popular relevant models *Satisfaction-based Interest Acceptance(SBA)* by Atanasyev et al. [21], *Satisfaction-Based Pushback (SBP)* algorithm by Salah et al. [34], *Coordination Monitoring Router (CNMR)* Model.

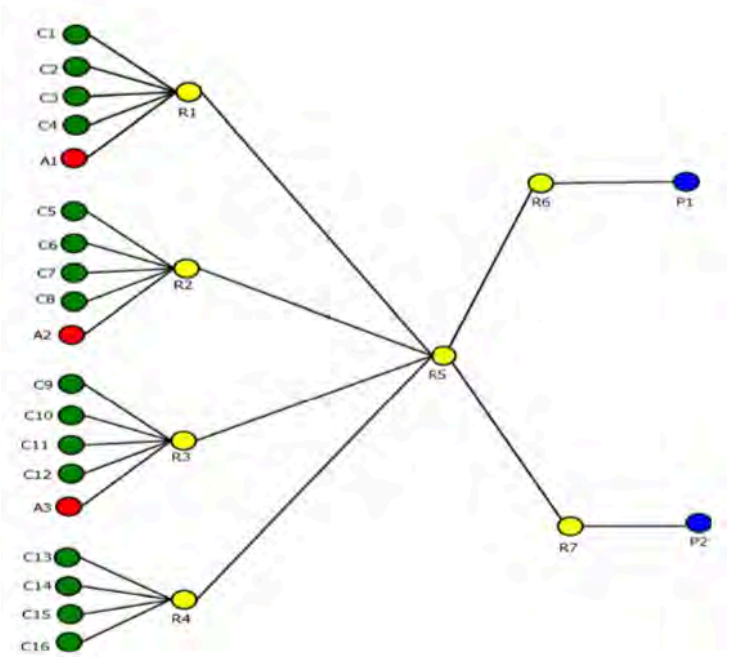
We used a standard network topology which is similar to the topology in [34]. It consists of 79 nodes and 147 bidirectional edges which is shown in Figure 4.1a (AS3967). Then we used another topology *Tree* topology consisting of 22 bidirectional edges and 28 nodes which is shown in Figure 4.1b for simulations. For normal data flow scenario, we have examined using the requests from every legitimate clients or consumers for existing contents with a rate of 100 *Interest Packets per Second* (*IPPS*). Each router has a uniform PIT capacity of 4000 entries. We assume that, every request packet is valid, and there is no attack ongoing. Thus, the average PIT utilization is very low and does not create any PIT overflow. For attack detection, each attacker sends requests at higher rates for non-existent contents. We have varied the number of interests and simulated with three different attack rate: 500 IPPS, 1000 IPPS, 10000 IPPS. The simulation time is set to 380s. Each simulation run time lasts for 6 minutes: the attack starts at the beginning of minute 1 (second 60) and stops at end of minute 6 (second 340). We have used uniform Interest packet's size of 1100 bytes.

The lines are labeled SBA, SBP, CNMR [21], [34]. We also compared our detection mechanism with PIT size for different PIT entries and the probability of dropped packets of legitimate

users [2].



(a) AS3967 Topology [34].



(b) Tree topology [35].

Figure 4.1: Different topologies.

Here, Table A.17 provides simulation parameters.

Table 4.1: Simulation Parameters

Parameter	Value
PIT Size	1200 KB
Interest expire time	4s
CS Size	4000
Packet Size	1100 Bytes

Three relevant metrics are used to measure the performance and effectiveness of our algorithm against Interest Flooding Attack. These are *Interest Satisfaction Ratio (ISR)*, *Pending Interest Table (PIT) Usage*, *Dropped Interest Ratio*.

- ISR: It indicates the quality of the satisfaction ratio of legitimate Interest packets during the attack period. It gives an indication of the quality of service perceived by consumer while the network attack is ongoing.
- PIT Usage: It indicates the protection of PIT usage measured as a ratio of the overall PIT space which is the direct target of IFAs. PIT Usage shows the process of legitimate traffic using available capacity on routers during an attack over a time window.
- Drop Interest Ratio: In terms of legitimate traffic and malicious traffic drop ratio, we can easily evaluate the effectiveness of detection under IFA. This shows the percentage of Legitimate Interest packets and Malicious Interest Packets over total received at each router, respectively.

We simulate by varying the number of Interests per second, and we look into the impact on ISR, PIT Usage, Packet Drop probability, PIT size under different attack rates. We have checked eight routers (R1-R8) during the attack time and evaluated the impact on *Dropped Interest Ratio* of *Malicious Interest*s and *Legitimate Interest*s. We collect the output from simulator and present into graphical format (stated in the following section).

4.2 Impact on ISR

We have checked the impact on Interest Satisfaction Ratio by changing the number of interests per second and checking the ISR during attack time. We estimate the Legitimate Interest packets satisfaction ratio by running our defence mechanism and other state of the art methods. We compare these results to a system without defence, a system including satisfaction-based acceptance (SBA), a system including satisfaction-based push back (SBP) and a system including Coordination Monitoring Routers (CNMR).

4.2.1 ISR under Attack-500 IPPS

To measure ISR, we maintain the fixed number of interests 200 per second, packet size to 1100 bytes and we keep the attack rate 500 IPPS. Attacks have been generated at 61s and ended at 340s. Figure 4.2 and Figure 4.3 show the graphical representation of ISR under the attack rate of 500 IPPS.

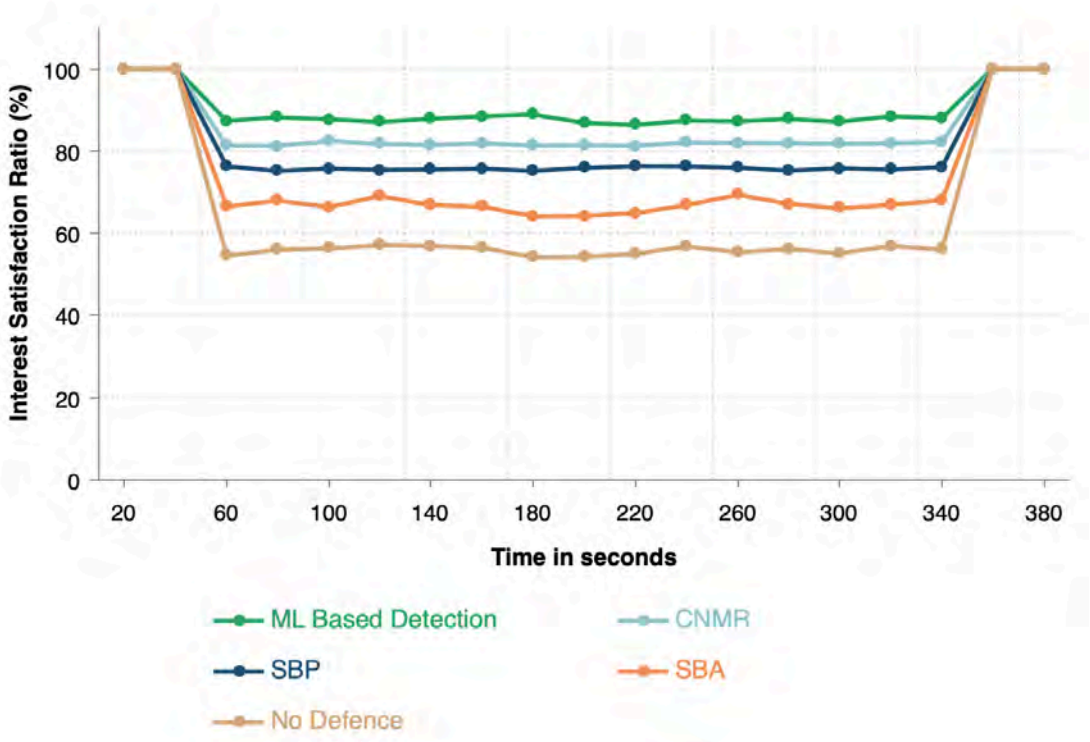


Figure 4.2: ISR under attack rate of 500 IPPS (attack period: sec. 61 sec. 340) (AS 3967 topology).

We compared the performance of our proposed method under 500 IPPS with all other algorithms. We find that our method improve the attack scenario compares with no defence. Here, we saw that SBA is a very simple algorithm and of very light weight, it doesn't perform well but SBP is more effective and it increases ISR value from 58% to 78%. CNMR is also very effective and it increases ISR value from 58% to 81%. When enabling our defence mechanism, it gives optimal solution comparing with other algorithms and it increases the ISR value from 58% to 85% using AS 3967 topology, which is shown in Figure 4.2 and 58% to 92% using tree topology, which is shown in Figure 4.3. Both topologies satisfies our original intention.

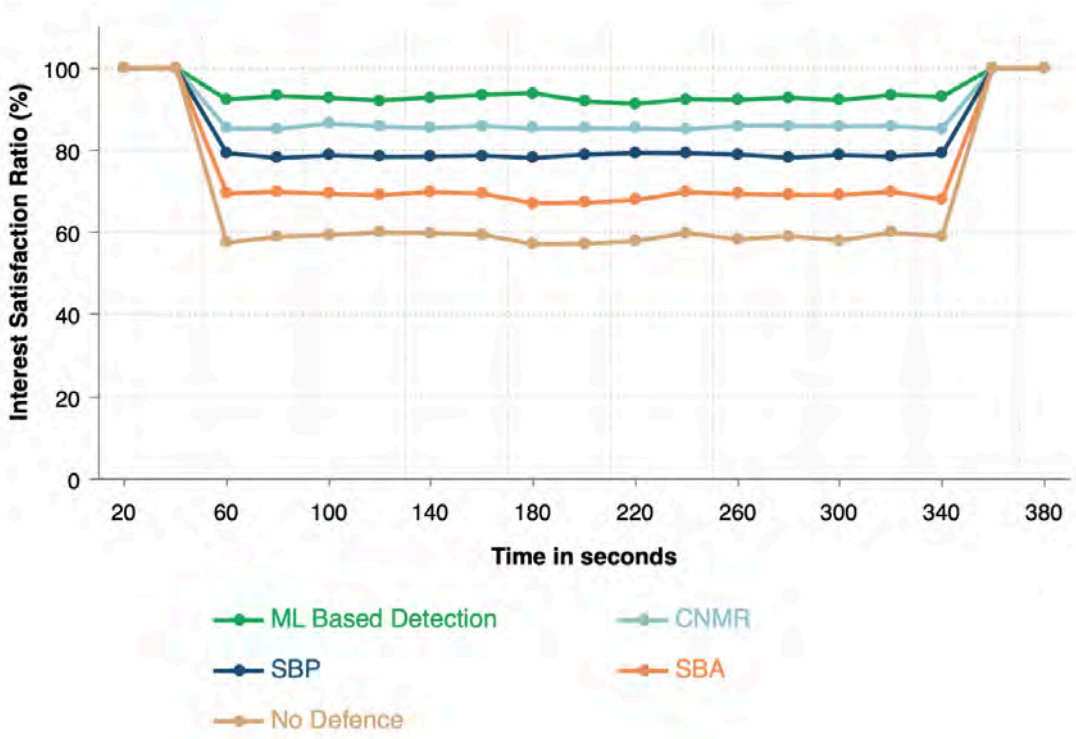


Figure 4.3: ISR under attack rate of 500 IPPS (attack period: sec. 61 sec. 340) (Tree topology).

4.2.2 ISR under Attack-1000 IPPS

To estimate the ISR value, we maintained the uniform packet size of 1100 byte and we kept the attack rate to 1000 IPPS. We simulated the program in total 380s. We recorded the attack, generated at 61s and end at 340s. Figure 4.4 and Figure 4.5 shows the graphical representation of ISR under attack rate of 1000 IPPS.

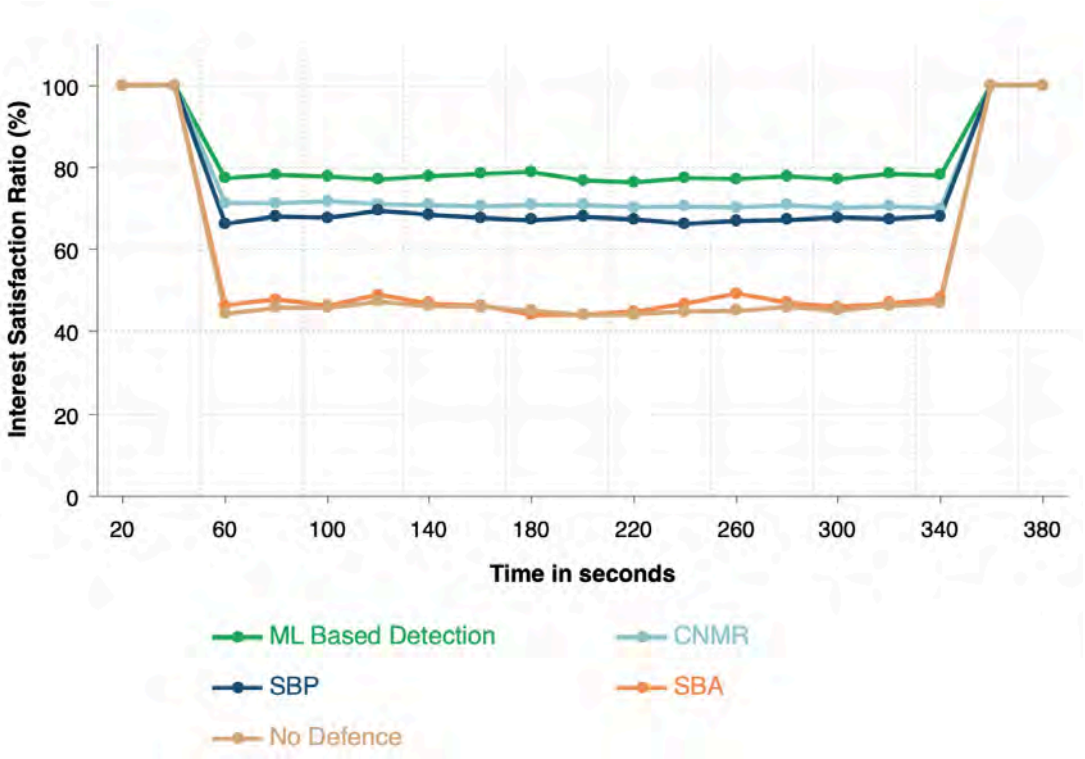


Figure 4.4: ISR under attack rate of 1000 ipps (attack period: sec. 61 sec. 340) (AS 3967 topology).

With the increase of attack rate, the number of Interest satisfaction ratio is decreased. SBA could not handle the situation and worked like that there is no defence mechanism. SBP slightly improved the situation but CNMR performed better. ML based detection method increases satisfaction ratio better than previous any other state of the art methods and it increases the ISR value from 42% to 80% using AS 3967 topology, which is shown in Figure 4.4 and 42% to 84% using tree topology, which is shown in Figure 4.5.

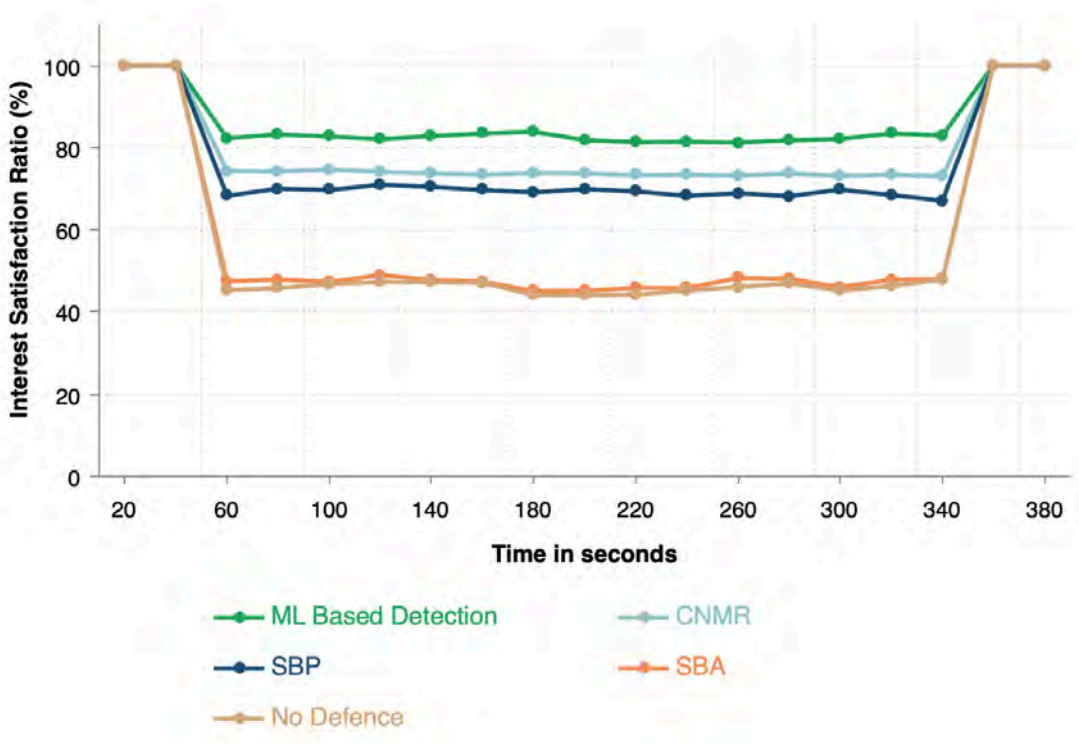


Figure 4.5: ISR under attack rate of 1000 ipps (attack period: sec. 61 sec. 340) (Tree topology).

4.2.3 ISR under Attack-10000 IPPS

To measure ISR, we now maintain the fixed number of Interest packets 10000 per second, uniform packet size 1100 byte and we keep the attack rate same during the whole attack duration. Attacks have been generated at 61s and end at 340s. Figure 4.6 shows the graphical representation of ISR under attack rate 10000 IPPS. We compare the performance of our proposed method with SBA, SBP, CNMR.

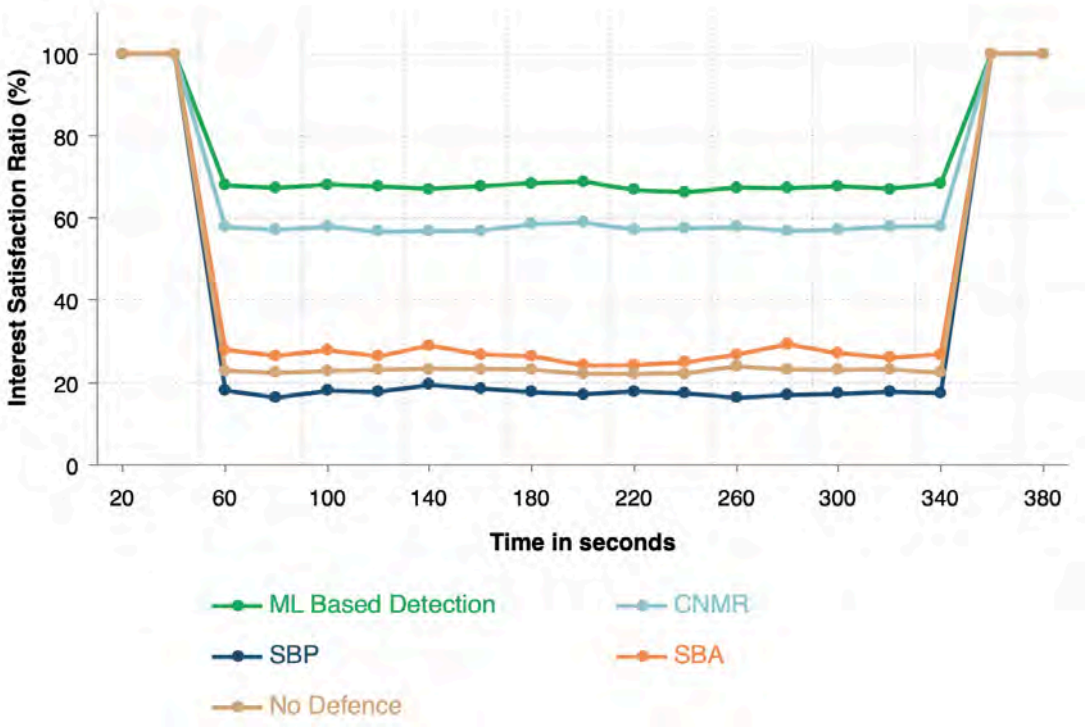


Figure 4.6: ISR under attack rate of 10000 IPPS (attack period: sec. 61 sec. 340) (AS 3967 topology)

With the increase of the number of Interests per second, the number of Interest Satisfaction Ratio is decreased which is normal because for load is being generated in the scenario. The utility of SBA almost disappears under high attack rates and behaved almost similar to no defence. SBP also did not perform well in this scenario but CNMR has improved the scenario slightly. We can see that the ISR during the attack period improves significantly with our mechanism, it does not cause packet drops and enables full satisfaction in both topologies, which are shown in Figure 4.6 and Figure 4.7.

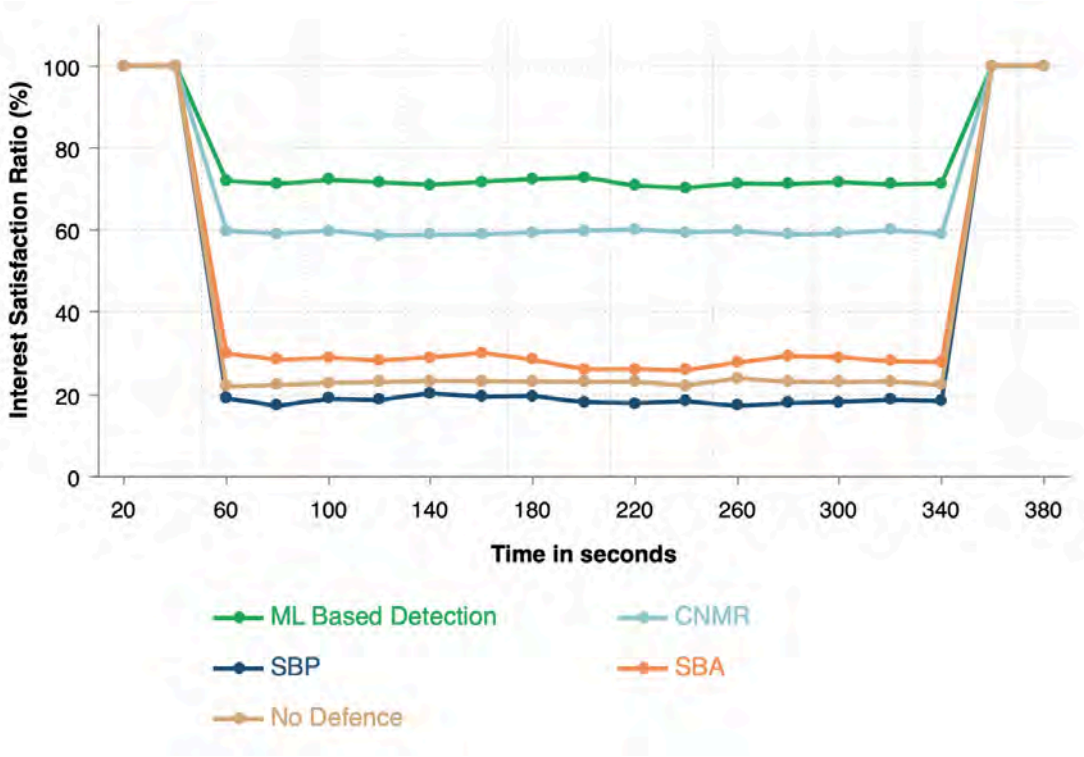


Figure 4.7: ISR under attack rate of 10000 IPPS (attack period: sec. 61 sec. 340) (Tree topology)

4.3 Impact on PIT Usage

We've checked the impact on PIT Usage by changing the number of interests per second and evaluating the PIT occupied ratio during attack time. PIT usage is a great indicator of router load.

4.3.1 PIT Usage During Attack without Defence

To measure PIT Usage we maintained the same uniform packet size of 1100 byte and we vary the attack rate to 500 IPPS, to 1000 IPPS and to 10000 IPPS. Attack generated at 61s and end at 340s. Figure 4.8 and Figure 4.9 shows the graphical representation of PIT Usage under attack.

We have compared the performance of our proposed method with a method without any defence mechanism.

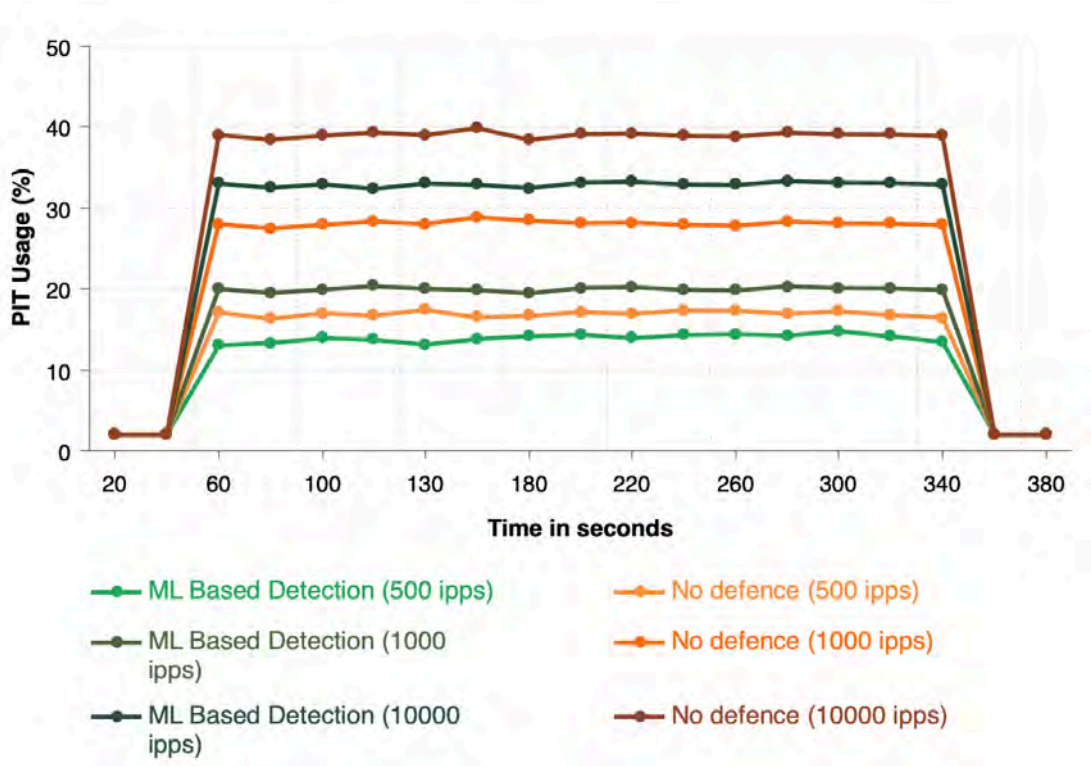


Figure 4.8: PIT usage under three attack rates (attack period: sec. 61 sec. 340)

The efficiency of ML based attack detection against IFAs is also checked by PIT Usage. We compare our results with a method without defence by varying the number of attack rate. ML based detection showed the lower PIT usage during the attack period. If defence is not used, PIT usage is increased, proportionally with the attack rate.

PIT usage is reduced from 18% to 12% under attack rate of 500 IPPS, from 28% to 20% under attack rate of 1000, and from 40% to 34% under attack rate of 10000 IPPS, which is shown in Figure 4.8 using AS 3967 topology. It should be noted that this reduction is measured over all PITs.

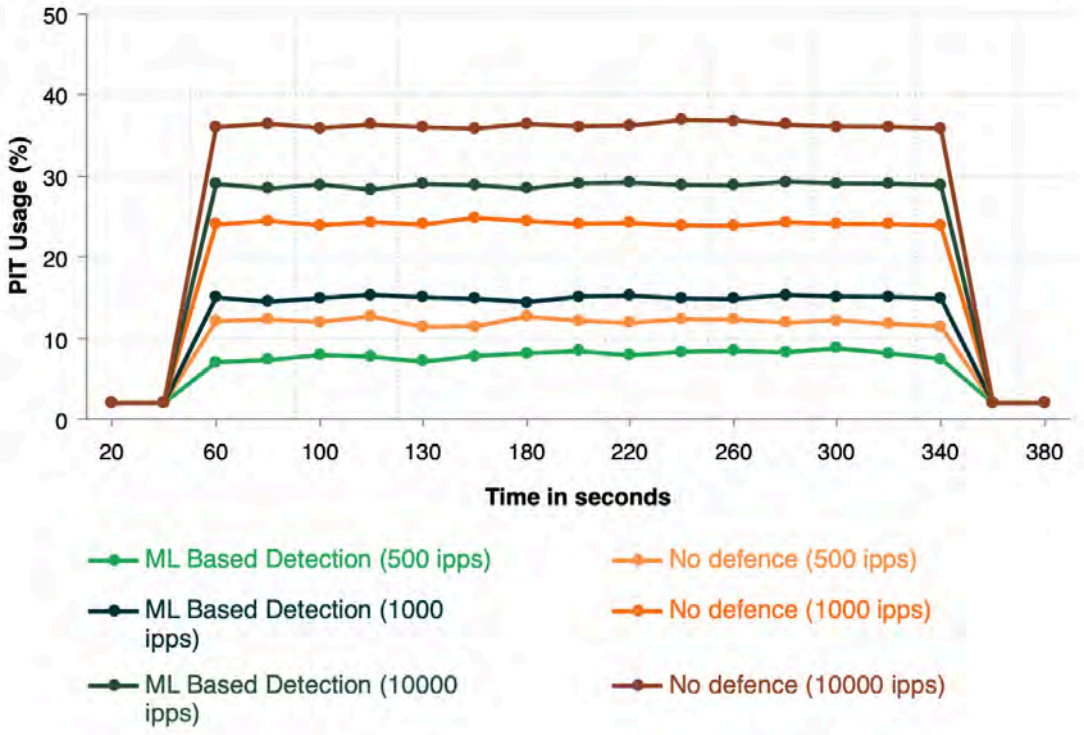


Figure 4.9: PIT usage under three attack rates (attack period: sec. 61 sec. 340) (Tree topology)

PIT usage is reduced from 18% to 8% under attack rate of 500 IPPS, from 28% to 15% under attack rate of 1000, and from 38% to 30% under attack rate of 10000 IPPS, which is shown in Figure 4.9 using tree topology.

4.3.2 PIT Usage During Attack with Defence Mechanism

We kept the number of Interest per second 500 and packet size 1100 byte. Attack generated at 61s and end at 340s. Figure 4.10 and Figure 4.11 shows the graphical representation of PIT Usage under attack. We compared the performance of our proposed method with three other mitigation strategies named SBA, SBP, CNMR and evaluated PIT Usage during this period.

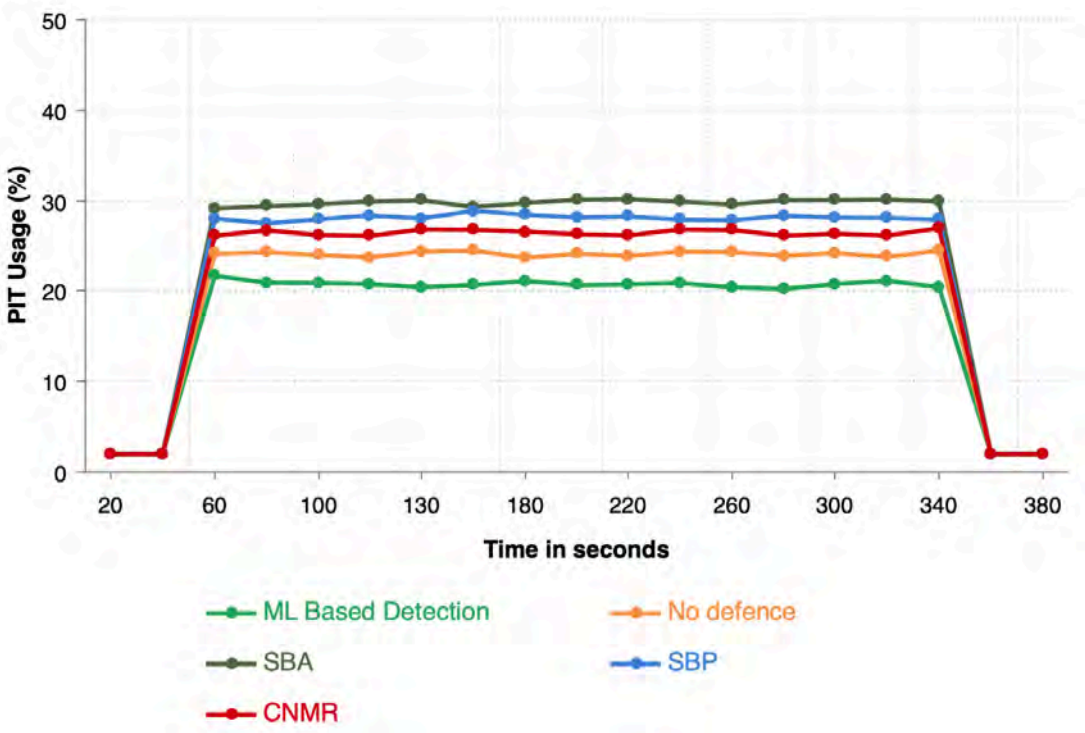


Figure 4.10: PIT usage under attack (attack period: sec. 61 sec. 340) compare (AS3967 topology) with other state of the art.

The reduction value per router differs by the amount of Malicious Interest packets each router receives. PIT usage is reduced from under attack rate of 500 IPPS compare with no defence about 30% to about 21% using ML Based Attack Detection(AS3967 topology), which is shown in Figure 4.10, 30% to about 17% using ML Based Attack Detection(Tree topology), which is shown in Figure 4.11, 30% to about 39% using SBA, 30% to about 25% using SBP, 30% to about 23% using CNMR. The effectiveness of our mechanism is confirmed by the results of the PIT Usage.

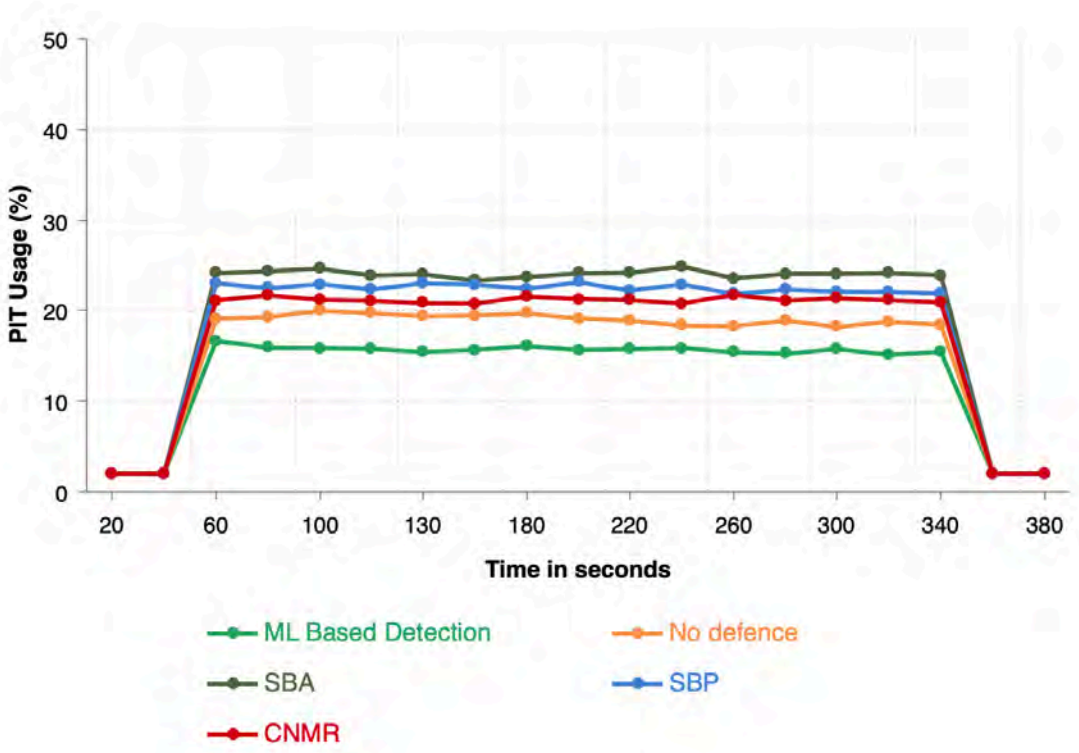


Figure 4.11: PIT usage under attack (attack period: sec. 61 sec. 340) compare (Tree topology) with other state of the art.

4.4 Dropped Interest Ratio During Attack

In terms of legitimate traffic and malicious traffic drop, Figure 4.12 reports effectiveness of ML Based Attack Detection under IFA. Dropped Interest Ratio is the percentage of total Legitimate Interests and Malicious Interests dropped over total received at each router.

4.4.1 Impact on Dropped Interests

We kept the number of Interest packets per second as 1000 and packet size to 1100 bytes. We observed eight routers R1 to R8 during the attack period.

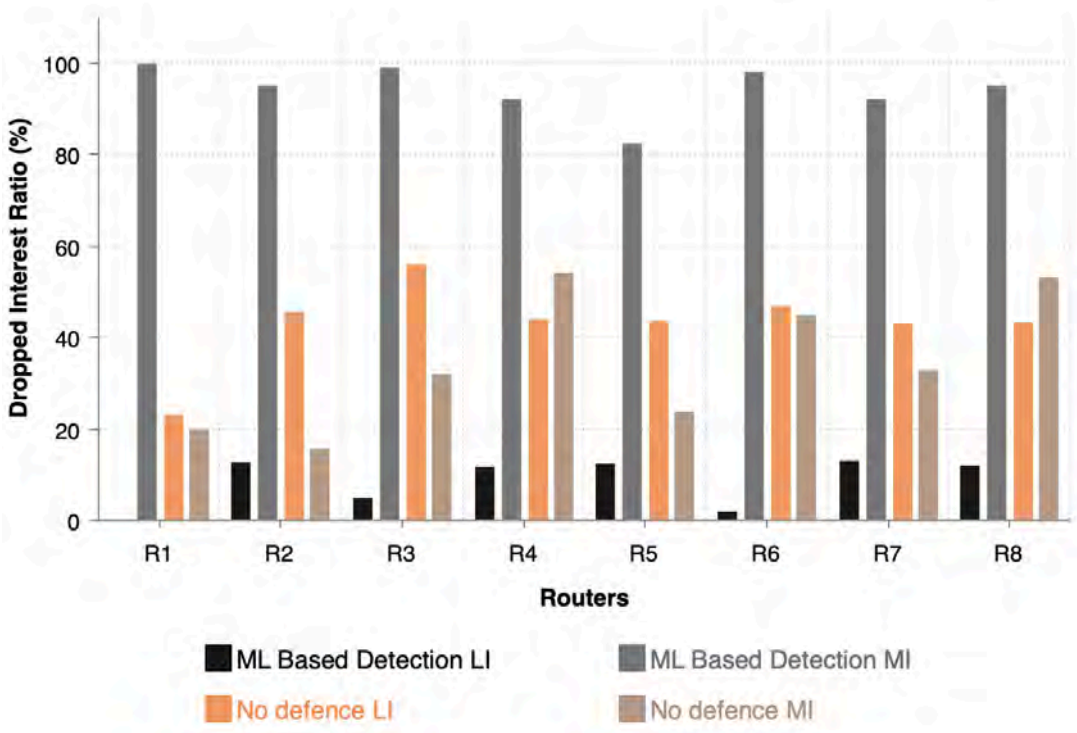


Figure 4.12: MI drop and LI drop without defence(AS3967 topology).

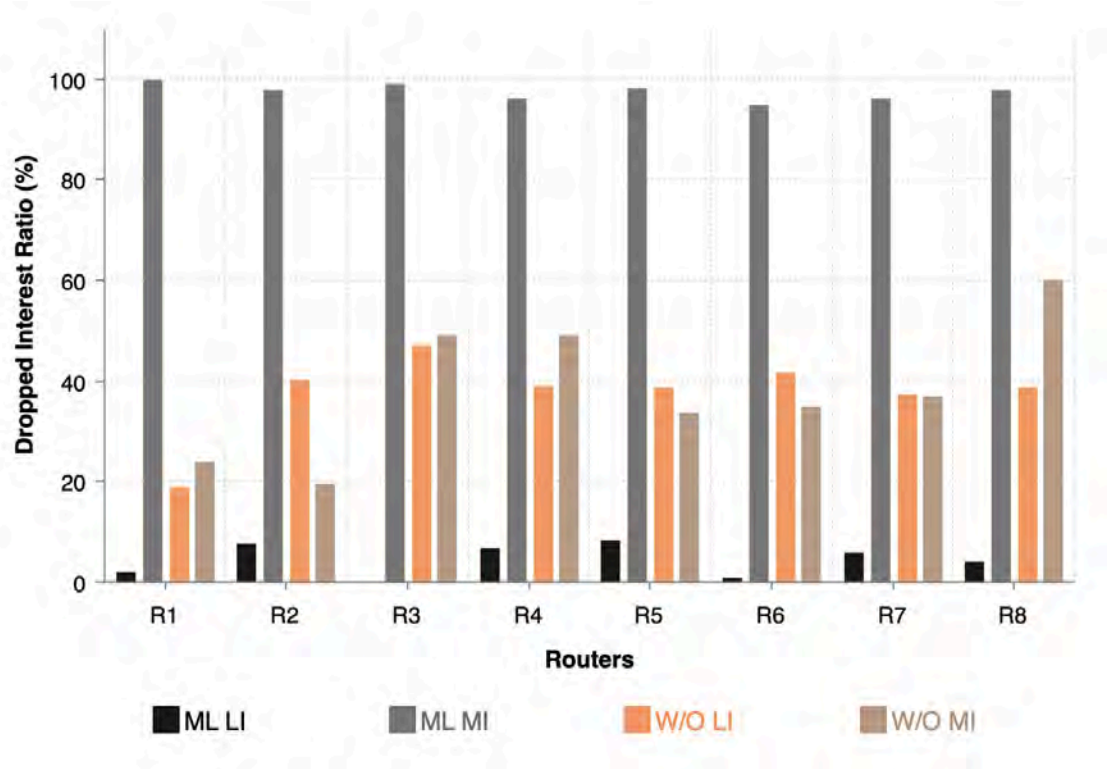


Figure 4.13: MI drop and LI drop without defence(Tree topology).

Without defence scenario, PIT is filled up with *Malicious Interests* very quickly. In this graph, we have shown that 91% of Malicious Interests are dropped during attack but with the use

of ML based detection only 5% of *Legitimate Interests* are dropped. This is what is needed to mitigate DDoS attack.

4.4.2 Comparison with SBA for Dropped Interests

We kept the number of Interest packet per second as 1000 and packet size to 1100 bytes. We observed eight routers R1 to R8 during the attack period. We observed Dropped Ratio using our defence mechanism and using SBA.

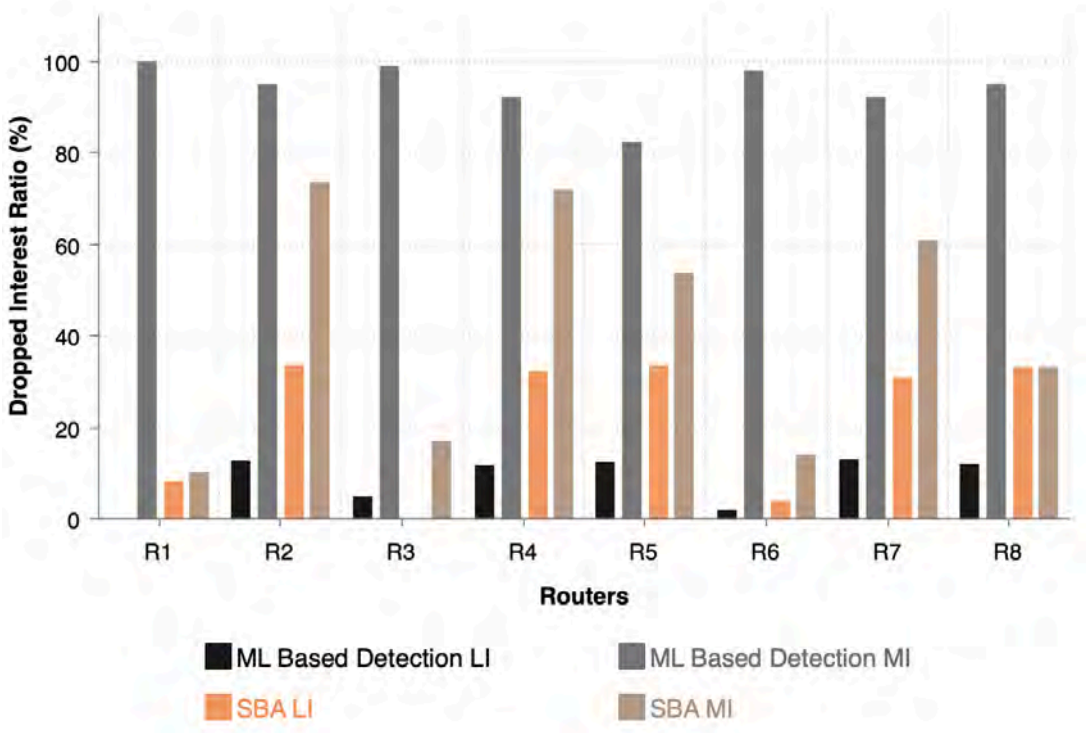


Figure 4.14: MI drop and LI drop compared with SBA (AS3967 topology).

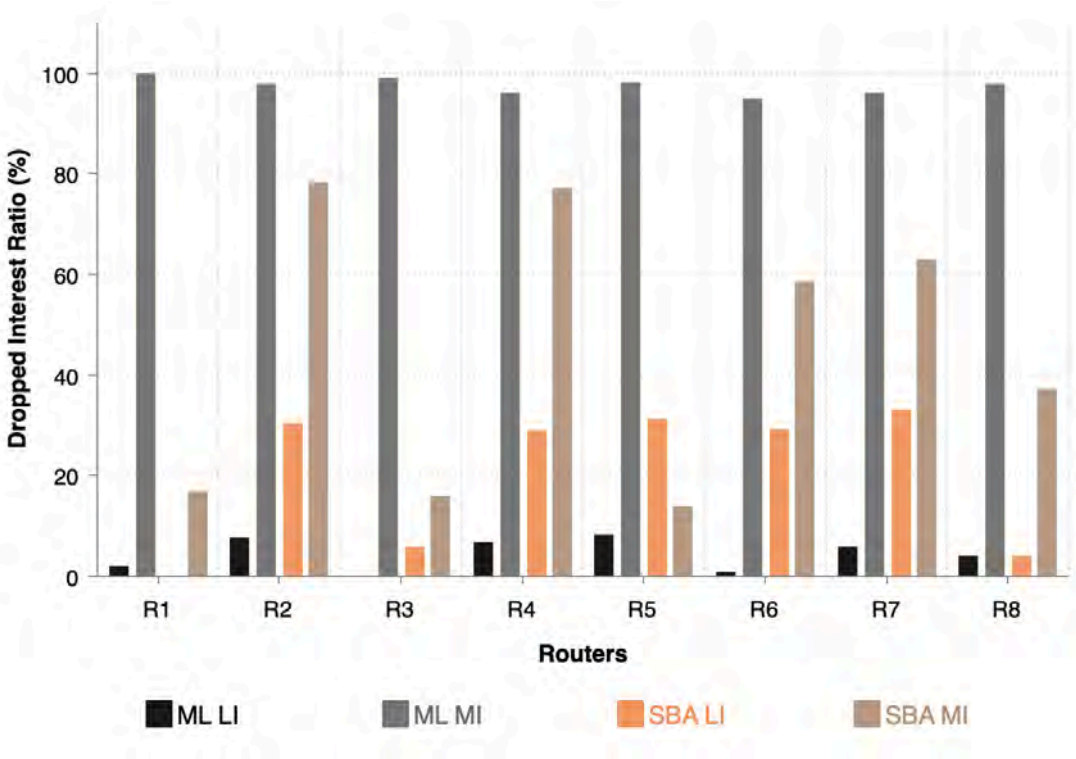


Figure 4.15: MI drop and LI drop compared with SBA (Tree topology).

The number of dropped interest ratio is increased with the increase of the number of Interests per second which is natural because the load becomes higher. We compared the performance of our proposed method with SBA. It shows that some routers using SBA performs better but overall ML based detection works better.

4.4.3 Comparison with SBP for Dropped Interests

We kept the number of Interest packet per second as 1000 and packet size as 1100 bytes. We observed eight routers R1 to R8 during the attack period. We report dropped ratio using our defence mechanism and using SBP.

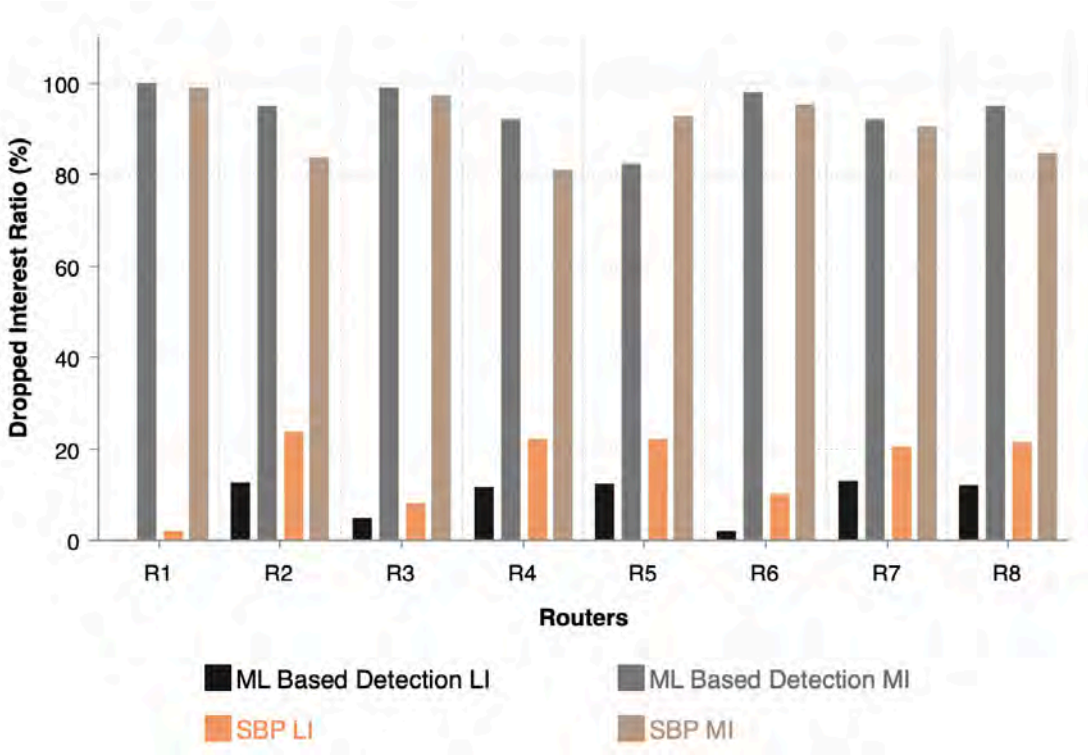


Figure 4.16: ML and SBP(AS 3967 topology).

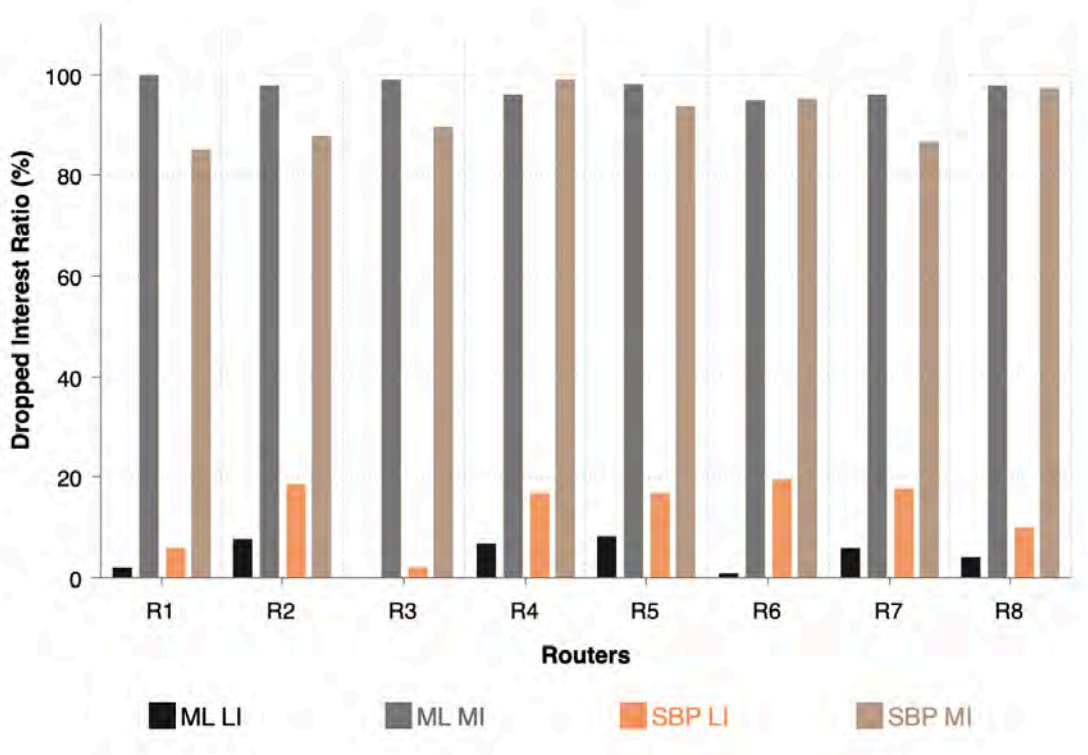


Figure 4.17: ML and SBP(Tree topology).

The number of legitimate packet drop is increased under the attack scenario. We compared the performance of our proposed method with SBP. Simulation result shows that our method

shows better result in dropping the less legitimate packets under attack scenario of NDN, which fulfils our original intention of designing this strategy.

4.4.4 Comparison with CNMR for Dropped Interest

We kept the number of Interest packet per second as 1000 and packet size 1100 bytes. We observed the output of eight routers from R1 to R8 during the attack period. We computed interest dropped ratio using our method and compare with CNMR.

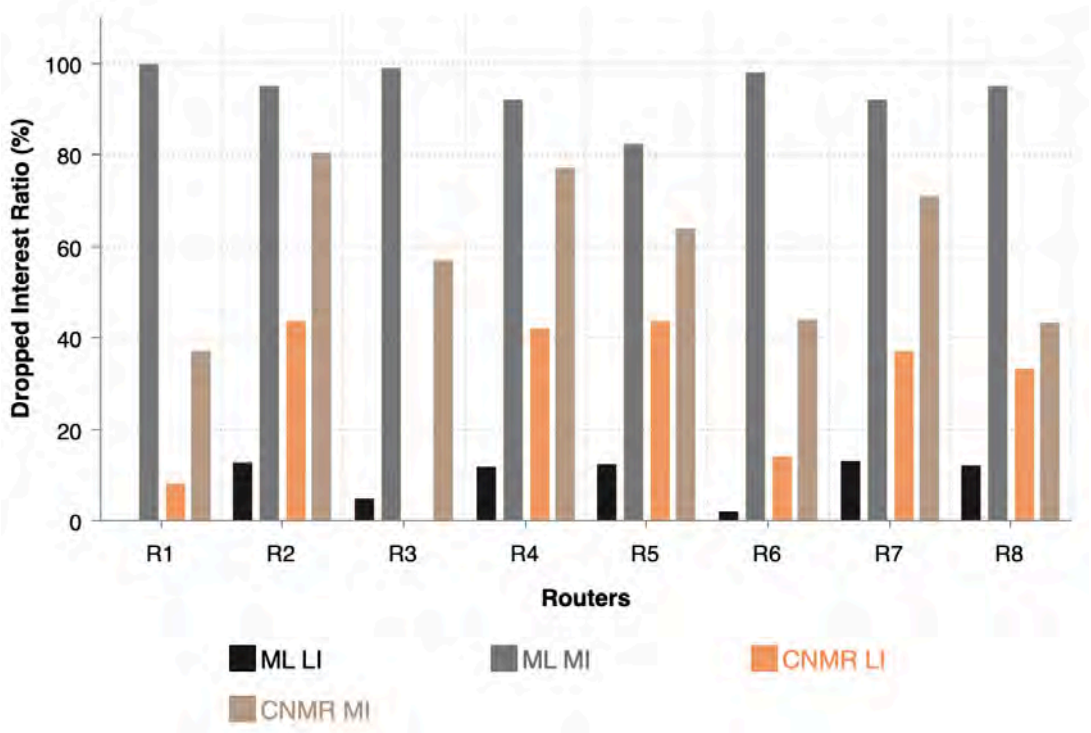


Figure 4.18: MI drop and LI drop compare with CNMR(AS 3967 topology).

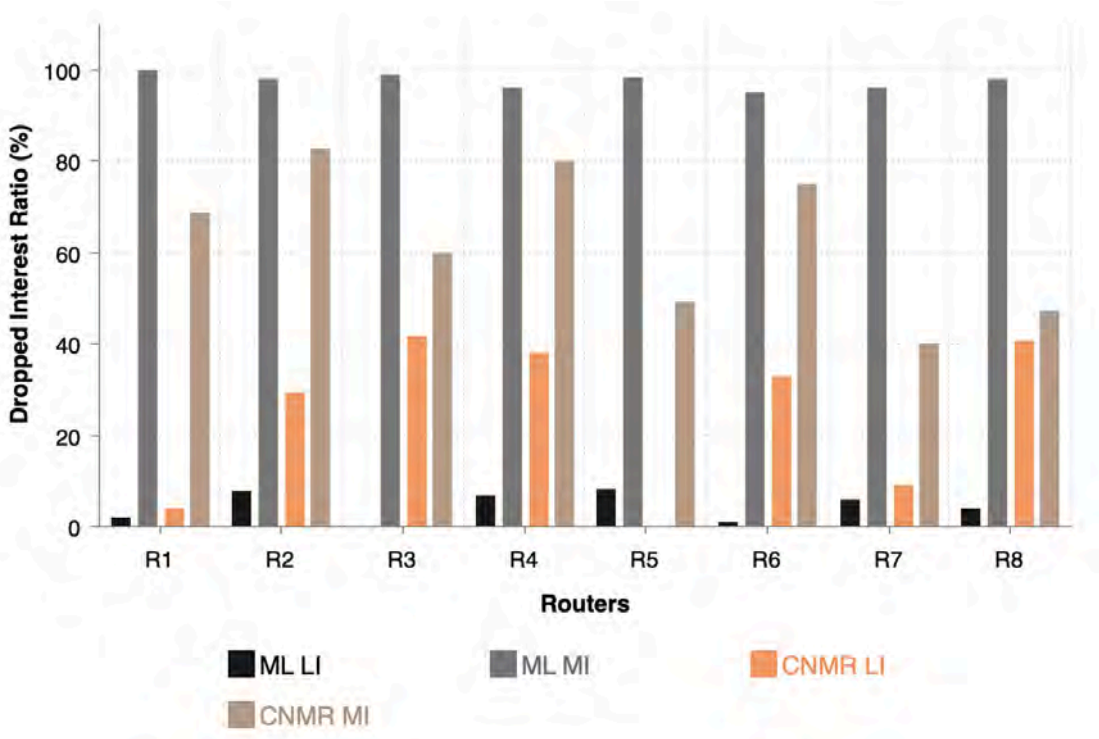


Figure 4.19: MI drop and LI drop compare with CNMR(Tree topology)

A consequence of the poor detection has been achieved by CNMR. We found more space across monitoring routers' PITs. Higher PIT values corresponds to more computational and memory resources used in routers. CNMR mitigates only specific detection as infected prefix names. Compared with the performance of our proposed method, it gives optimal solution where PIT value does not increase as sharply as for CNMR.

4.5 PIT Size under Attack

Under attack, large amount of routers' storage can be occupied by attacker nodes. Figure 4.20 shows that how easily PIT storage can be filled up during attack.

We kept the PIT entry expiration time as 500ms and the number of Interest packets per second is 1000 and packet size is 1100bytes. Total simulation run time duration 60s. The attack starts at $t = 4$s and lasts until $t = 42$s. The results are shown for three different PIT entry expiration times without defence and with our defence mechanism.

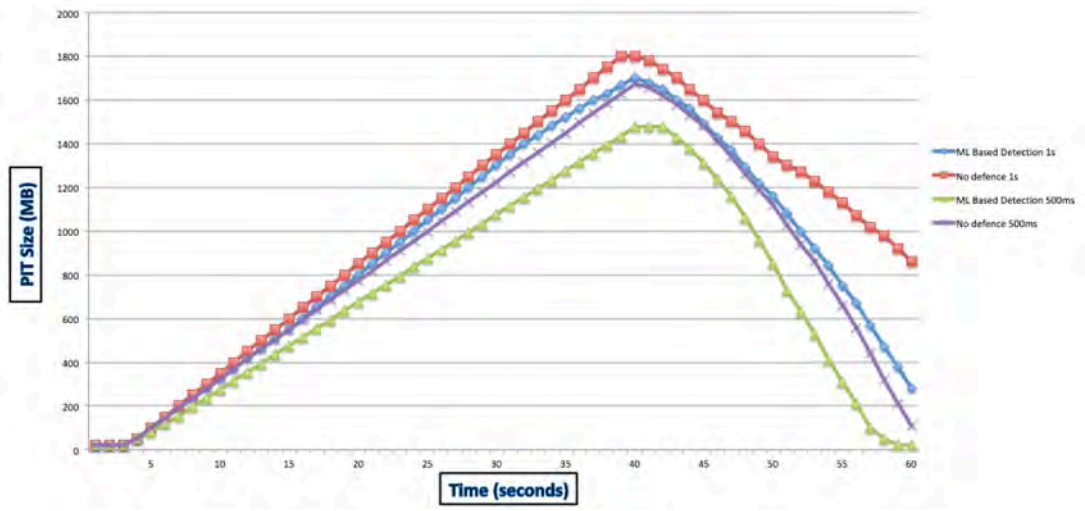


Figure 4.20: PIT size for different PIT entry expiration times.

Tough the corresponding PIT entry of each request will expire with such small expiration time of 500ms before the content arrives, a large number of legitimate requests will not be satisfied. Before attack, PIT size is usually very small but during attack period PIT size increases rapidly. During this scenario, our ML based detection minimizes PIT size to relatively small and satisfies our main intention.

4.6 Probability of Packet Drop

Due to PIT overflow legitimate users packet dropping probability is the negative impact of IFA. Figure 4.21 shows the packet dropping probability under attack.

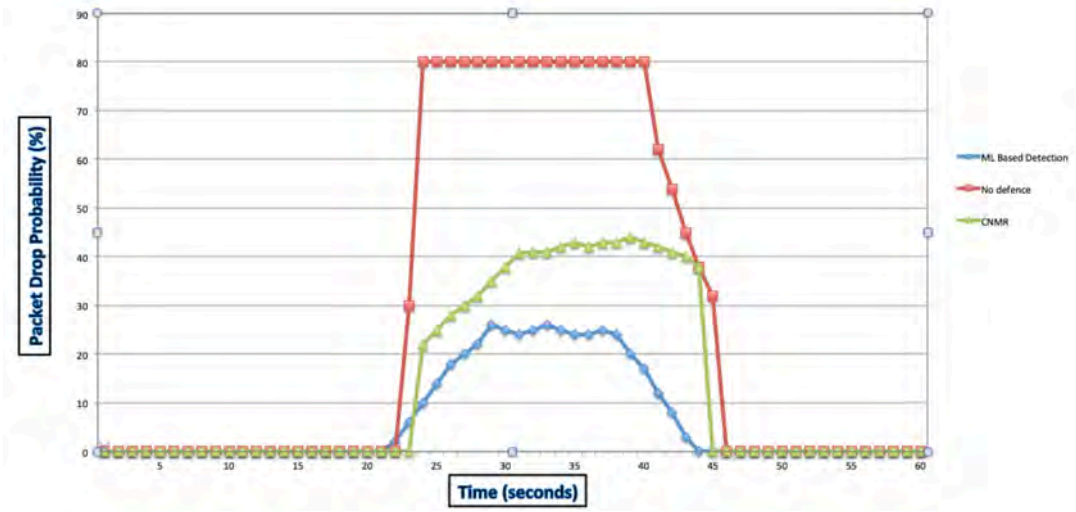


Figure 4.21: Packet dropping probability of legitimate users

We kept the PIT entry expiration time as 500ms, and 1s respectively and the number of

Interest packets per second is 1000 and packet size of 1100 bytes. Total simulation run time duration is 50s. We evaluated and compared our ML based attack detection with packet dropping probability of legitimate users for the basic NDN and two threshold-based mitigation schemes and without defence. When no defence is used, then 80% of the Interest packets of legitimate users are dropped between 22s to 42s. CNMR improved the situation 42% during under attack. Using our methodology, packet drop is reduce to 28% between 23s to 44s. Thus, we can see our model improves the scenario under attack.

Chapter 5

Conclusion

In this chapter, we conclude our thesis by presenting the contributions. Machine Learning has gained a lot of popularity and momentum in the recent years. Different types of learning and classification algorithms have shown tremendous promise of efficiency and speed in this domain. We have collected different types of features that get affected due to Interest Flooding Attack and selected main features by applying k-means clustering to get major required parameters.

In terms of accuracy, precision, sensitivity or recall, and specificity, our results show that machine learning approaches are better than the statistical approaches for handling Interest Flooding Attacks in NDN. We implement our algorithm on AS3967 topology, tree topology using ndnSIM. We compare the output with other popular methods. We have found better performance of our method.

Our proposed framework is capable to mitigate DDoS attack for NDN.

- In detection phase, learning continuously network traffic data and updating SVM by training gives us a better system to detect DDoS attack.
- In mitigation phase, a set of rules are followed by NDN Controller for further action.
- Our method has used standard topology.

5.1 Future Work

- We would like to implement our method with benchmark topology for large scale network.
- We will explore other feature selection methods for including wider range of attacks where differences in these parameters are subtle.

References

[1] "Named data networking." Last Accessed on February 02, 2019. Available: http://named-data.net/.

[2] V. Vassilakis, "Mitigating distributed denial-of-service attacks in named data networking," 06 2015.

[3] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 181–192, 10 2007.

[4] C. Dannewitz, "Netinf: An information-centric design for the future internet," 01 2009.

[5] Visala, Kari, Visala@hiit, Kari, Fi, Dmitrij, Lagutin, Tarkoma, and Sasu, "Lanes: An inter-domain data-oriented routing architecture," December 2009.

[6] G. Tselentis, A. Galis, A. Gavras, S. Krco, V. Lotz, E. Simperl, B. Stiller, and T. Zahariadis, *Towards the Future Internet - Emerging Trends from European Research.* 04 2010.

[7] N. Fotiou, P. Nikander, D. Trossen, and G. Polyzos, "Developing information networking further: from psirp to pursuit," vol. 66, 10 2010.

[8] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[9] K. Wang, H. Zhou, Y. Qin, J. Chen, and H. Zhang, "Decoupling malicious interests from pending interest table to mitigate interest flooding attacks," in *2013 IEEE Globecom Workshops (GC Wkshps)*, pp. 963–968, Dec 2013.

[10] "Ndn project team. ndn technical memo: Naming conventions," *Technical report, NDN*, July 2014.

[11] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "Dos and ddos in named data networking," in *22nd International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–7, IEEE, 2013.

[12] M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in named data networking," *Comput. Netw.*, vol. 57, pp. 3178–3191, Nov. 2013.

[13] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, and C. Papadopoulos, "Named data networking (ndn) project," *Xerox Palo Alto Research Center-PARC*, 2010.

[14] N. Chhetry and H. K. Kalita, "Interest flooding attack in named data networking: A survey," 03 2016.

[15] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[16] H. Dai, Y. Wang, J. Fan, and B. Liu, "Mitigate ddos attacks in ndn by interest traceback," in *Conference on Computer Communications Workshops.(INFOCOM WKSHPS)*, pp. 381–386, IEEE, 2013.

[17] A. Karami and M. Guerrero-Zapata, "A hybrid multiobjective rbfpso method for mitigating dos attacks in named data networking," in *Neurocomputing*, vol. 151, pp. 1262–1282, 2015.

[18] T. Nguyen, X. Marchal, G. Doyen, T. Cholez, and R. Cogranne, "Content poisoning in named data networking: Comprehensive characterization of real deployment," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 72–80, May 2017.

[19] M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in named data networking," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 57, pp. 3178–3191, 11 2013.

[20] D. Saxena, "Named data networking: A survey," *Elsevier Computer Science Review*, 01 2016.

[21] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," in *IFIP Networking Conference*, pp. 1–9, IEEE, 2013.

[22] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating interest flooding ddos attacks in named data networking," in *IFIP Networking Conference*, pp. 630–638, IEEE, 2013.

[23] G. P. Compagno A, Conti M, "Ndn interest flooding attacks and countermeasures," in *Annual Computer Security Applications Conference*, 2012.

[24] S. Choi, K. Kim, S. Kim, and B. Roh, "Threat of dos by interest flooding attack in content-centric networking," in *International Conference on Information Networking (ICOIN)*, pp. 315–319, IEEE, 2013.

[25] A. R. Gawande, "Ddos detection and mitigation using machine learning," tech. rep., Rutgers, The State University of New Jersey, 2018.

[26] T. Mitchell, *Machine Learning. McGraw-Hill international editions - Computer Science Series*. McGraw-Hill Education, 1997.

[27] A. R. Jeffrey D. Ullman Jure Leskovec, "Mining of massive datasets, chapter clutering," Stanford University, 2014.

[28] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 881–892, July 2002.

[29] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pp. 1027–1035, 2007.

[30] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Sept. 1995.

[31] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim: ndn simulator for ns-3," 01 2012.

[32] W. Isaac and S. Iyer, "Software-defined security," tech. rep., 04 2018.

[33] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[34] H. Salah, J. Wulfheide, and T. Strufe, "Coordination supports security: A new defence mechanism against interest flooding in ndn," 10 2015.

[35] N. Kumar, A. K. Singh, and S. Srivastava, "Feature selection for interest flooding attack in named data networking," 2019.

Appendix A

Raw Data

A.1 ISR vs Time under attack rate of 500 ipps

Table A.1: Running Time vs ISR, under attack rate of 500 ipps

Running Time(s)	ISR Under SBA (%)	ISR Under SBP (%)	ISR Under CNMR (%)	ISR Under ML Based Detection (%)
20	100	100	100	100
40	100	100	100	100
60	66.46	76.28	81.24	87.3
80	66.46	75.12	81.2	88.2
100	66.33	66.33	82.4	87.7
120	69.02	75.38	81.7	87.08
140	66.86	75.47	81.41	87.8
160	66.43	75.67	81.87	88.45
180	64.12	75.13	81.26	88.87
200	64.21	75.9	81.3	86.8
220	64.9	76.34	81.22	86.3
240	66.83	76.29	82.06	87.4
260	69.31	75.92	81.93	87.23
280	67.11	75.2	81.89	87.76
300	66.07	75.78	81.76	87.12
320	66.89	75.45	81.85	88.44
340	68.02	76.1	82.2	88
360	100	100	100	100
380	100	100	100	100

Table A.2: Running Time vs ISR, under attack rate of 500 ipps using tree topology

Running Time(s)	ISR Under SBA (%)	ISR Under SBP (%)	ISR Under CNMR (%)	ISR Under ML Based Detection (%)
20	100	100	100	100
40	100	100	100	100
60	92.3	92.3	79.28	79.28
80	93.2	85.2	78.12	69.93
100	92.7	86.4	78.71	69.33
120	92.08	85.7	78.38	69.02
140	92.8	92.8	78.47	69.86
160	93.45	93.45	78.67	69.43
180	93.87	78.13	78.13	78.13
200	91.9	85.3	78.9	78.9
220	91.3	85.22	85.22	67.9
240	92.4	92.4	79.29	79.29
260	92.23	85.93	78.92	69.31
280	69.31	69.31	78.2	69.11
300	92.12	85.76	78.78	78.78
320	93.44	85.85	78.45	69.89
340	93	85.2	79.1	68.02
360	100	100	100	100
380	100	100	100	100

A.2 ISR vs Time under attack rate of 1000 ipps

Table A.3: Running Time vs ISR, under attack rate of 1000 ipps

Running Time(s)	ISR Under SBA (%)	ISR Under SBP (%)	ISR Under CNMR (%)	ISR Under ML Based Detection (%)
20	100	100	100	100
40	100	100	100	100
60	46.46	46.46	71.3	71.3
80	47.93	68	71.2	78.2
100	46.33	67.71	71.7	77.7
120	49.02	69.38	71.08	77.08
140	77.08	77.08	77.08	77.08
160	46.43	67.67	70.45	78.45
180	44.12	67.13	70.87	78.87
200	44.21	67.9	70.8	76.8
220	44.9	67.34	70.3	76.3
240	44.83	66.29	70.4	77.4
260	49.31	66.92	70.23	77.23
280	47.11	67.2	70.76	77.76
300	46.07	67.78	70.12	77.12
320	46.89	67.45	70.44	78.44
340	48.02	68.1	70.01	78
360	100	100	100	100
380	100	100	100	100

Table A.4: Running Time vs ISR, under attack rate of 1000 ipps using tree topology

Running Time(s)	ISR Under SBA (%)	ISR Under SBP (%)	ISR Under CNMR (%)	ISR Under ML Based Detection (%)
20	100	100	100	100
40	100	100	100	100
60	47.46	68.28	74.3	82.3
80	47.93	70	74.2	83.2
100	47.33	69.71	74.7	82.7
120	49.02	70.98	74.08	82.08
140	47.86	70.47	73.8	82.8
160	47.43	69.67	73.45	83.45
180	45.12	69.13	73.87	83.87
200	45.21	69.9	73.8	81.8
220	45.9	69.34	73.3	81.3
240	45.83	68.29	73.4	81.4
260	48.31	68.92	73.23	81.23
280	48.11	68.2	73.76	81.76
300	46.07	69.78	73.12	82.12
320	47.89	68.45	73.44	83.44
340	48.02	67.1	73.01	83
360	100	100	100	100
380	100	100	100	100

A.3 ISR vs Time under attack rate of 10000 ipps

Table A.5: Running Time vs ISR, under attack rate of 10000 ipps

Running Time(s)	ISR Under SBA (%)	ISR Under SBP (%)	ISR Under CNMR (%)	ISR Under ML Based Detection (%)
20	100	100	100	100
40	100	100	100	100
60	28.02	18.1	57.8	68
80	26.46	16.28	16.28	67.3
100	27.93	18	57.86	68.2
120	26.33	17.71	56.7	67.7
140	29.02	19.38	56.88	67.08
160	26.82	18.47	56.98	67.8
180	26.43	17.67	58.45	68.45
200	24.12	67.9	70.8	76.8
220	24.21	17.9	57.18	66.8
240	24.9	17.34	57.43	66.3
260	26.83	16.29	57.74	67.4
280	29.31	16.92	56.93	67.23
300	27.11	17.2	57.17	67.76
320	26.07	17.78	57.92	67.12
340	26.89	17.45	58.06	68.44
360	100	100	100	100
380	100	100	100	100

Table A.6: Running Time vs ISR, under attack rate of 10000 ipps using tree topology

Running Time(s)	ISR Under SBA (%)	ISR Under SBP (%)	ISR Under CNMR (%)	ISR Under ML Based Detection (%)
20	100	100	100	100
40	100	100	100	100
60	30.02	19.1	59.8	72
80	28.46	17.28	59.13	71.3
100	28.93	19	59.86	72.2
120	28.33	18.71	58.7	71.7
140	29.02	20.38	58.88	71.08
160	30.2	19.47	58.98	71.8
180	28.43	19.67	59.45	72.45
200	26.12	18.13	59.87	72.87
220	26.21	17.9	60.18	70.8
240	25.9	18.34	59.43	70.3
260	27.83	17.29	59.74	71.4
280	29.31	17.92	58.93	71.23
300	29.11	18.2	59.17	71.76
320	28.07	18.78	59.92	71.12
340	27.9	18.45	59.06	71.44
360	100	100	100	100
380	100	100	100	100

A.4 Dropped Interest ratio

Table A.7: ML Based Detection vs No Defence

Router	ML Based Detection LI drop (%)	ML Based Detection MI drop (%)	No Defence LI drop(%)	No Defence MI drop (%)
R1	0	100	8	10
R2	12.7	94.87	33.54	73.54
R3	5	99	0	17
R4	11.8	92.11	32.07	72.07
R5	12.3	82.3	33.67	53.67
R6	2	98	4	14
R7	12.92	92.12	30.98	60.98
R8	12.2	95.02	33.14	33.14

Table A.8: ML Based Detection vs No Defence using tree topology

Router	ML Based Detection LI drop (%)	ML Based Detection MI drop (%)	No Defence LI drop(%)	No Defence MI drop (%)
R1	2	100	19	24
R2	7.7	97.87	40.14	19.54
R3	0	99	47	49
R4	6.8	96.11	39.07	49.14
R5	8.3	98.3	38.67	33.67
R6	1	95	41.76	35
R7	5.92	96.12	37.36	36.98
R8	4	98.02	38.64	60

Table A.9: ML Based Detection vs SBA

Router	ML Based Detection LI drop (%)	ML Based Detection MI drop (%)	SBA LI drop(%)	SBA MI drop (%)
R1	0	100	8	10
R2	12.7	94.87	33.54	73.54
R3	5	99	0	17
R4	11.8	92.11	32.07	72.07
R5	12.3	82.3	33.67	53.67
R6	2	98	4	14
R7	12.92	92.12	30.98	60.98
R8	12.2	95.02	33.14	33.14

Table A.10: ML Based Detection vs SBA using tree topology

Router	ML Based Detection LI drop (%)	ML Based Detection MI drop (%)	SBA LI drop(%)	SBA MI drop (%)
R1	2	100	0	17
R2	7.7	97.87	30.54	78.34
R3	0	99	6	16
R4	6.8	96.11	29.07	77.07
R5	8.3	98.3	31.23	14
R6	1	95	29.14	58.67
R7	5.92	96.12	33.14	62.98
R8	4	98.02	4	37.14

Table A.11: ML Based Detection vs SBP

Router	ML Based Detection LI drop (%)	ML Based Detection MI drop (%)	SBP LI drop(%)	SBP MI drop (%)
R1	0	100	2	99
R2	12.7	94.87	23.72	83.72
R3	5	99	8	97.4
R4	11.8	92.11	22	81.07
R5	12.3	82.3	22.29	92.83
R6	2	98	10	95.32
R7	12.92	92.12	20.62	90.62
R8	12.2	95.02	21.53	84.53

Table A.12: ML Based Detection vs SBP using tree topology

Router	ML Based Detection LI drop (%)	ML Based Detection MI drop (%)	SBP LI drop(%)	SBP MI drop (%)
R1	2	100	6	85.07
R2	7.7	97.87	18.72	87.72
R3	0	99	2	89.62
R4	6.8	96.11	16.93	99
R5	8.3	98.3	17	93.83
R6	1	95	19.53	95.32
R7	5.92	96.12	17.62	86.53
R8	4	98.02	10	97.4

Table A.13: ML Based Detection vs CNMR

Router	ML Based Detection LI drop (%)	ML Based Detection MI drop (%)	CNMR LI drop(%)	CNMR MI drop (%)
R1	0	100	8	37
R2	12.7	94.87	43.54	80.54
R3	5	99	0	57
R4	11.8	92.11	42.07	77.07
R5	12.3	82.3	43.67	63.67
R6	2	98	14	44
R7	12.92	92.12	36.98	70.98
R8	12.2	95.02	33.14	43.14

Table A.14: ML Based Detection vs CNMR using tree topology

Router	ML Based Detection LI drop (%)	ML Based Detection MI drop (%)	CNMR LI drop(%)	CNMR MI drop (%)
R1	2	100	4	68.67
R2	7.7	97.87	29.14	82.54
R3	0	99	41.67	60
R4	6.8	96.11	38.07	80.07
R5	8.3	98.3	0	49
R6	1	95	32.98	74.98
R7	5.92	96.12	9	40
R8	4	98.02	40.54	47.14

A.5 PIT Usage vs Time

Table A.15: PIT Usage vs Time under PIT expire time 1s

Running Time (s)	PIT Size under ML Based Detection(%)	PIT Size under No Defence(%)
5	100	100
10	325	350
15	550	600
20	800	850
25	1050	1100
30	1300	1350
35	1520	1600
40	1700	1800
45	1490	1600
50	1160	1340
55	750	1130
60	280	860

Table A.16: PIT Usage vs Time under PIT expire time 500ms

Running Time (s)	PIT Size under ML Based Detection(%)	PIT Size under No Defence(%)
5	80	100
10	275	325
15	475	550
20	675	775
25	875	1000
30	1075	1225
35	1275	1450
40	1475	1675
45	1310	1480
50	850	1120
55	310	660
60	10	110

A.6 Packet Drop probability

Table A.17: Packet Drop Probability under Attack

Running Time (s)	Packet Drop Probability of ML Based Detection(%)	Packet Drop Probability of No Defence(%)	Packet Drop Probability of CNMR(%)
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0
25	14	80	25
30	25	80	38
35	24	80	43
40	17	80	43
45	0	32	0
50	0	0	0
55	0	0	0
60	0	0	0

Appendix B

Codes

B.1 *k*-mean Cluster

We use this code to find out cluster parameters.

```
1
2 import random
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7 sns.set()
8 from sklearn.cluster import KMeans
9
10
11 original_data=pd.read_csv("cluster_interest_data.csv")
12 column_original_data=original_data.copy()
13 keep_col = ['InInterest_Count','PIT_Usage']
14 data = original_data[keep_col]
15 data
16
17 keep_type_col = ['Type']
18 column_data = column_original_data[keep_type_col]
19
20 plt.scatter(data['InInterest_Count'],data['PIT_Usage'])
21 plt.xlabel('InInterest_Count')
22 plt.ylabel('PIT_Usage')
23 plt.show()
```

```
24
25 x=data.copy()
26
27 kmeans=KMeans(3)
28 kmeans.fit(x)
29
30 clusters=x.copy()
31 clusters['cluster_pred']=kmeans.fit_predict(x)
32 plt.scatter(clusters['InInterest_Count'],clusters['PIT_Usage'],c=clusters
33 plt.xlabel('InInterest_Count')
34 plt.ylabel('PIT_Usage')
35 plt.show()
36
37 from sklearn import preprocessing
38 x_scaled=preprocessing.scale(x)
39 x_scaled
40
41 wcss=[]
42
43 for i in range(1,26):
44     kmeans=KMeans(i)
45     kmeans.fit(x_scaled)
46     wcss.append(kmeans.inertia_)
47 wcss
48
49
50 plt.plot(range(1,26),wcss)
51 plt.xlabel('Number␣of␣clusters')
52 plt.ylabel('Percent␣of␣variance')
53 plt.show()
54
55
56 kmeans_new=KMeans(4)
57 kmeans.fit(x_scaled)
58 cluster_new=x.copy()
59 cluster_new['cluster_pred']=kmeans_new.fit_predict(x_scaled)
60 cluster_new
61
```

```
62
63 plt.scatter(cluster_new['InInterest_Count'],cluster_new['PIT_Usage'],c=cl
64 plt.xlabel('InInterest_Count')
65 plt.ylabel('PIT_Usage')
66 plt.show()
67
68
69 m_type = column_data
70 cluster_new.insert(0, "Type", m_type, False)
71 cluster_new.to_csv('kmeans_output.csv', encoding='utf-8', index=False)
```

B.2 SVM Model

We use this code to create SVM Model for attack detection.

```
1 from IPython import get_ipython
2 import random
3 # **Classifying Legitimate and Malicious with SVM**
4 # __Step 1:__ Import Packages
5 # Packages for analysis
6 import pandas as pd
7 import numpy as np
8 from sklearn import svm
9
10 # Packages for visuals
11 import matplotlib.pyplot as plt
12 import seaborn as sns; sns.set(font_scale=1.2)
13
14 # Allows charts to appear in the notebook
15 get_ipython().run_line_magic('matplotlib', 'inline')
16
17 # Pickle package
18 import pickle
19
20 # __Step 2:__ Import Data
21 # Read in Legitimate and Malicious data
22 dataset = pd.read_csv('../kmean/kmeans_output.csv')
23 dataset
```

```
24
25 # __Step 3:__ Prepare the Data
26
27 # Plot two dataItem
28 sns.lmplot('InInterest_Count', 'PIT_Usage', data=dataset, hue='Type',
29            palette='Set1', fit_reg=False, scatter_kws={"s": 70});
30
31
32 # Specify inputs for the model
33 # dataItem = dataset[['Similar_Prefix_Name_Count', 'Interface', 'PIT_Usag
34 dataItem = dataset[['InInterest_Count','PIT_Usage']].as_matrix()
35 type_label = np.where(dataset['Type']=='Legitimate', 0, 1)
36
37 # Feature names
38 data_features = dataset.columns.values[1:].tolist()
39 data_features
40
41 # __Step 4:__ Fit the Model
42 # Fit the SVM model
43 model = svm.SVC(kernel='linear')
44 model.fit(dataItem, type_label)
45
46 # __Step 5:__ Visualize Results
47
48 # Get the separating hyperplane
49 w = model.coef_[0]
50 a = -w[0] / w[1]
51 xx = np.linspace(20, 90)
52 yy = a * xx - (model.intercept_[0]) / w[1]
53
54 # Plot the parallels to the separating hyperplane that pass through the s
55 b = model.support_vectors_[0]
56 yy_down = a * xx + (b[1] - a * b[0])
57 b = model.support_vectors_[-1]
58 yy_up = a * xx + (b[1] - a * b[0])
59
60
61 # Plot the hyperplane
```

```
62 sns.lmplot('InInterest_Count', 'PIT_Usage', data=dataset, hue='Type', pal
63 plt.plot(xx, yy, linewidth=2, color='black');
64
65
66 # Look at the margins and support vectors
67 sns.lmplot('InInterest_Count', 'PIT_Usage', data=dataset, hue='Type', pal
68 plt.plot(xx, yy, linewidth=2, color='black')
69 plt.plot(xx, yy_down, 'k--')
70 plt.plot(xx, yy_up, 'k--')
71 plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],
72             s=80, facecolors='none');
73
74 # __Step 6:__ Predict New Case
75
76 # Create a function to guess when a recipe is a Legitimate or a Malicious
77 def legitimate_or_malicious(InInterest_Count, PIT_Usage):
78     if(model.predict([[InInterest_Count, PIT_Usage]]))==0:
79         print('You\'re␣looking␣at␣a␣Legitimate␣traffic')
80     else:
81         print('You\'re␣looking␣at␣a␣Malicious␣traffic')
82
83
84 # Predict if 50 parts Similar_Prefix_Name_Count and 20 parts PIT_Usage
85 legitimate_or_malicious(100, 98)
86
87
88 # Plot the point to visually see where the point lies
89 sns.lmplot('InInterest_Count', 'PIT_Usage', data=dataset, hue='Type', pal
90 plt.plot(xx, yy, linewidth=2, color='black')
91 plt.plot(50, 20, 'yo', markersize='9');
92
93
94 # Predict if 40 parts Similar_Prefix_Name_Count and 20 parts PIT_Usage
95 legitimate_or_malicious(100,98)
96
97
98 attack_data_dict = {'legitimate_malicious_model': model, 'legitimate_mali
99
```

```
100
101 attack_data_dict
102
103
104 # Pickle
105 pickle.dump(attack_data_dict, open("attack_data_dict.p", "wb"))
106
107
108 # S = String
109 pickle.dumps(attack_data_dict)
```

B.3 DdosAttack-and-Mitigation

We use this scenerio to run NDN ns-3 Simulator to mitigate DDoS Attack.

```
1
2
3 #include "ns3/core-module.h"
4 #include "ns3/network-module.h"
5 #include "ns3/ndnSIM-module.h"
6
7 #include <boost/lexical_cast.hpp>
8
9 using namespace ns3;
10 using namespace ns3::ndn;
11 using namespace std;
12
13 #include "calculate-max-capacity.h"
14
15 uint32_t Run = 1;
16
17 void PrintTime (Time next, const string name)
18 {
19   cerr << "␣===␣" << name << "␣" << Simulator::Now ().ToDouble (Time::S)
20   Simulator::Schedule (next, PrintTime, next, name);
21 }
22
23 int main (int argc, char**argv)
```

```
24 {
25   string folder = "tmp";
26   string topology = "small-tree";
27   string prefix = "";
28   string producerLocation = "gw";
29   Time defaultRtt = Seconds (0.25);
30   Time evilGap = Time::FromDouble (0.02, Time::MS);
31   uint32_t badCount = 1;
32   uint32_t goodCount = 1;
33
34
35   CommandLine cmd;
36   cmd.AddValue ("topology", "Topology", topology);
37   cmd.AddValue ("run", "Run", Run);
38   cmd.AddValue ("defaultRtt", "Default␣RTT␣for␣BDP␣limits", defaultRtt);
39   cmd.AddValue ("algorithm", "DDoS␣mitigation␣algorithm", prefix);
40   cmd.AddValue ("producer", "Producer␣location:␣gw␣or␣bb", producerLocati
41   cmd.AddValue ("badCount", "Number␣of␣bad␣guys", badCount);
42   cmd.AddValue ("goodCount", "Number␣of␣good␣guys", goodCount);
43   cmd.AddValue ("folder", "Folder␣where␣results␣will␣be␣placed", folder);
44   cmd.Parse (argc, argv);
45
46   Config::SetGlobal ("RngRun", IntegerValue (Run));
47   StackHelper helper;
48
49   AppHelper evilAppHelper ("DdosApp");
50   evilAppHelper.SetAttribute ("Malicious", BooleanValue (true));
51   evilAppHelper.SetAttribute ("LifeTime", StringValue ("1s"));
52   evilAppHelper.SetAttribute ("DataBasedLimits", BooleanValue (true));
53
54   AppHelper goodAppHelper ("DdosApp");
55   goodAppHelper.SetAttribute ("LifeTime",  StringValue ("1s"));
56   goodAppHelper.SetAttribute ("DataBasedLimits", BooleanValue (true));
57
58   AppHelper ph ("ns3::ndn::Producer");
59   ph.SetPrefix ("/good");
60   ph.SetAttribute ("PayloadSize", StringValue("1100"));
61
```

```
62  string name = prefix;
63  name += "-topo-" + topology;
64  name += "-evil-" + boost::lexical_cast<string> (badCount);
65  name += "-good-" + boost::lexical_cast<string> (goodCount);
66  name += "-producer-" + producerLocation;
67  name += "-run-"  + boost::lexical_cast<string> (Run);
68
69
70  string graph_dot_file    = "results/" + folder + "/" + name + ".dot";
71  string graph_pdf_file    = "results/" + folder + "/" + name + ".pdf";
72  string results_file = "results/" + folder + "/" + name + ".txt";
73  string meta_file    = "results/" + folder + "/" + name + ".meta";
74
75  if (prefix == "simple-limits")
76    {
77      helper.EnableLimits (true, defaultRtt);
78      helper.SetForwardingStrategy ("ns3::ndn::fw::BestRoute::PerOutFaceL
79                                    "Limit", "ns3::ndn::Limits::Window");
80    }
81  else if (prefix == "fairness")
82    {
83      helper.EnableLimits (true, defaultRtt);
84      helper.SetForwardingStrategy ("ns3::ndn::fw::BestRoute::TokenBucket
85                                    "Limit", "ns3::ndn::Limits::Window");
86    }
87  else if (prefix == "satisfaction-accept")
88    {
89      helper.EnableLimits (true, defaultRtt);
90      helper.SetForwardingStrategy ("ns3::ndn::fw::BestRoute::Stats::Sati
91                                    "GraceThreshold", "0.05");
92    }
93  else if (prefix == "satisfaction-pushback")
94    {
95      helper.EnableLimits (true, defaultRtt);
96      helper.SetForwardingStrategy ("ns3::ndn::fw::BestRoute::Stats::Sati
97                                    "GraceThreshold", "0.01");
98    }
99  else
```

```
100     {
101       cerr << "Invalid␣scenario␣prefix" << endl;
102       return -1;
103     }
104
105   AnnotatedTopologyReader topologyReader ("", 1.0);
106   topologyReader.SetFileName ("topologies/" + topology + ".txt");
107   topologyReader.Read ();
108
109   helper.Install (topologyReader.GetNodes ());
110
111   topologyReader.ApplyOspfMetric ();
112
113   GlobalRoutingHelper grouter;
114   grouter.Install (topologyReader.GetNodes ());
115
116   NodeContainer leaves;
117   NodeContainer gw;
118   NodeContainer bb;
119   for_each (NodeList::Begin (), NodeList::End (), [&] (Ptr<Node> node) {
120       if (Names::FindName (node).compare (0, 5, "leaf-")==0)
121         {
122           leaves.Add (node);
123         }
124       else if (Names::FindName (node).compare (0, 3, "gw-")==0)
125         {
126           gw.Add (node);
127         }
128       else if (Names::FindName (node).compare (0, 3, "bb-")==0)
129         {
130           bb.Add (node);
131         }
132     });
133
134   system (("mkdir␣-p␣\"results/" + folder + "\"").c_str ());
135   ofstream os (meta_file.c_str(), ios_base::out | ios_base::trunc);
136
137   os << "Total_numbef_of_nodes␣␣␣␣␣␣" << NodeList::GetNNodes () << endl;
```

```
138   os << "Total number of leaf nodes " << leaves.GetN () << endl;
139   os << "Total number of gw nodes   " << gw.GetN () << endl;
140   os << "Total number of bb nodes   " << bb.GetN () << endl;
141
142   NodeContainer producerNodes;
143
144   NodeContainer evilNodes;
145   NodeContainer goodNodes;
146
147   set< Ptr<Node> > producers;
148   set< Ptr<Node> > evils;
149   set< Ptr<Node> > angels;
150
151   if (goodCount == 0)
152     {
153       goodCount = leaves.GetN () - badCount;
154     }
155
156   if (goodCount < 1)
157     {
158       NS_FATAL_ERROR ("Number of good guys should be at least 1");
159       exit (1);
160     }
161
162   if (leaves.GetN () < goodCount+badCount)
163     {
164       NS_FATAL_ERROR ("Number of good and bad guys ("<< (goodCount+badCou
165       exit (1);
166     }
167
168   if (producerLocation == "gw")
169     {
170       if (gw.GetN () < 1)
171         {
172           NS_FATAL_ERROR ("Topology does not have gateway nodes that can
173           exit (1);
174         }
175     }
```

```
176   else if (producerLocation == "bb")
177     {
178       if (bb.GetN () < 1)
179         {
180           NS_FATAL_ERROR ("Topology␣does␣not␣have␣backbone␣nodes␣that␣can
181           exit (1);
182         }
183     }
184   else
185     {
186       NS_FATAL_ERROR ("--producer␣can␣be␣either␣'gw'␣or␣'bb'");
187       exit (1);
188     }
189
190   os << "Number␣of␣Malicious␣nodes:␣" << badCount << endl;
191   while (evils.size () < badCount)
192     {
193       UniformVariable randVar (0, leaves.GetN ());
194       Ptr<Node> node = leaves.Get (randVar.GetValue ());
195
196       if (evils.find (node) != evils.end ())
197         continue;
198       evils.insert (node);
199
200       string name = Names::FindName (node);
201       Names::Rename (name, "malicious-"+name);
202     }
203
204   while (angels.size () < goodCount)
205     {
206       UniformVariable randVar (0, leaves.GetN ());
207       Ptr<Node> node = leaves.Get (randVar.GetValue ());
208       if (angels.find (node) != angels.end () ||
209           evils.find (node) != evils.end ())
210         continue;
211
212       angels.insert (node);
213       string name = Names::FindName (node);
```

```
214       Names::Rename (name, "good-"+name);
215     }
216
217   while (producers.size () < 1)
218     {
219       Ptr<Node> node = 0;
220       if (producerLocation == "gw")
221         {
222           UniformVariable randVar (0, gw.GetN ());
223           node = gw.Get (randVar.GetValue ());
224         }
225       else if (producerLocation == "bb")
226         {
227           UniformVariable randVar (0, bb.GetN ());
228           node = bb.Get (randVar.GetValue ());
229         }
230
231       producers.insert (node);
232       string name = Names::FindName (node);
233       Names::Rename (name, "producer-"+name);
234     }
235
236   auto assignNodes = [&os](NodeContainer &aset, const string &str) {
237     return [&os, &aset, &str] (Ptr<Node> node)
238     {
239       string name = Names::FindName (node);
240       os << name << "␣";
241       aset.Add (node);
242     };
243   };
244   os << endl;
245
246   // a little bit of C++11 flavor, compile with -std=c++11 flag
247   os << "Malicious:␣";
248   std::for_each (evils.begin (), evils.end (), assignNodes (evilNodes, "M
249   os << "\nGood:␣";
250   std::for_each (angels.begin (), angels.end (), assignNodes (goodNodes,
251   os << "\nProducers:␣";
```

```
252  std::for_each (producers.begin (), producers.end (), assignNodes (produ
253  os << "\n";
254
255  grouter.AddOrigins ("/", producerNodes);
256  grouter.CalculateRoutes ();
257
258  // verify topology RTT
259  for (NodeList::Iterator node = NodeList::Begin (); node != NodeList::En
260    {
261      Ptr<Fib> fib = (*node)->GetObject<Fib> ();
262      if (fib == 0 || fib->Begin () == 0) continue;
263
264      if (2* fib->Begin ()->m_faces.begin ()->GetRealDelay ().ToDouble (T
265        {
266          cout << "DefaultRTT␣is␣smaller␣that␣real␣RTT␣in␣the␣topology:␣"
267          os << "DefaultRTT␣is␣smaller␣that␣real␣RTT␣in␣the␣topology:␣" <
268        }
269    }
270
271  double maxNonCongestionShare = 0.8 * calculateNonCongestionFlows (goodN
272  os << "maxNonCongestionShare␣␣␣" << maxNonCongestionShare << endl;
273
274  saveActualGraph (graph_dot_file, NodeContainer (goodNodes, evilNodes));
275  system (("twopi␣-Tpdf␣\"" + graph_dot_file + "\"␣>␣\"" + graph_pdf_file
276  cout << "Write␣effective␣topology␣graph␣to:␣" << graph_pdf_file << endl
277  cout << "Max␣non-congestion␣share:␣␣␣" << maxNonCongestionShare << endl
278
279  // exit (1);
280
281  for (NodeContainer::Iterator node = goodNodes.Begin (); node != goodNod
282    {
283      ApplicationContainer goodApp;
284      goodAppHelper.SetPrefix ("/good/"+Names::FindName (*node));
285      goodAppHelper.SetAttribute ("AvgGap", TimeValue (Seconds (1.100 / m
286
287      goodApp.Add (goodAppHelper.Install (*node));
288
289      UniformVariable rand (0, 1);
```

```
290       goodApp.Start (Seconds (0.0) + Time::FromDouble (rand.GetValue (),
291     }
292
293   for (NodeContainer::Iterator node = evilNodes.Begin (); node != evilNod
294     {
295       ApplicationContainer evilApp;
296       evilAppHelper.SetPrefix ("/malicious/"+Names::FindName (*node));
297       evilApp.Add (evilAppHelper.Install (*node));
298
299       UniformVariable rand (0, 1);
300       evilApp.Start (Seconds (300.0) + Time::FromDouble (rand.GetValue ()
301       evilApp.Stop (Seconds (600.0) + Time::FromDouble (rand.GetValue (),
302     }
303
304   ph.Install (producerNodes);
305
306   L3RateTracer::InstallAll (results_file, Seconds (1.0));
307
308   Simulator::Schedule (Seconds (10.0), PrintTime, Seconds (10.0), name);
309
310   Simulator::Stop (Seconds (900.0));
311   Simulator::Run ();
312   Simulator::Destroy ();
313
314   L3RateTracer::Destroy ();
315
316   cerr << "Archiving␣to:␣" << results_file << ".bz2" << endl;
317   system (("rm␣-f␣\"" + results_file + ".bz2" + "\"").c_str() );
318   system (("bzip2␣\"" + results_file + "\"").c_str() );
319
320   return 0;
321 }
```

Generated using Postgraduate Thesis LaTeX Template, Version 1.01. Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

This thesis was generated on Monday 17th February, 2020 at 2:07pm.