Journal of Engineering Technology and Applied Physics

Adaptive Strategies to Mitigate DDoS Attacks in IoT-Devices Through A Moving Target Defense Approach in SDN

Soomal Qureshi[1, *], Hafiz Muhammad Attaullah[2, 3], Ayesha Ashraf[4] and Rabia Laraib[5]
[1]*NEDUET, Department of Telecommunications, Karachi, Pakistan.*
[2]*Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia.*
[3]*Faculty of Computing, Mohammad Ali Jinnah University, Pakistan.*
[4]*Université Libre de Bruxelles, Department of Cyber Security, Brussels, Belgium.*
[5]*NUST, School of Electrical Engineering and Computer Science, Islamabad, Pakistan.*
**Corresponding author*: soomalqureshiofficial@gmail.com, *ORCiD*: 0009-0009-5302-0885
https://doi.org/10.33093/jetap.2025.7.2.13
Manuscript Received: 17 February 2025, Accepted: 13 May 2025, Published: 15 September 2025

***Abstract*—The surge of IoT devices has revolutionized the world, but the inherent complexity and vulnerabilities of these devices pose significant security risks. Among security challenges, distributed denial of service (DDoS) attacks stands out as a major cybersecurity issue aimed at interfering with regular systems. This paper conducts a gap analysis of existing research on DDoS attacks in the context of SDN oriented IoT devices. The research focus is on comparing algorithms and mitigation strategies proposed in different research papers and evaluating their efficiency and cost-effectiveness as previous research efforts have taken a variety of approaches, some focused on inexpensive but ineffective procedures, while others focused on expensive but effective procedures. However, few studies have investigated both cost and performance effectiveness simultaneously. The main objective of this research paper is to evaluate and compare different strategies proposed in the literature to protect Software Defined Network oriented IoT devices from DDoS attacks through an active approach using MTD (Moving Target Defense) technique. The goal of this strategy is to protect the network from attacks while remaining cost-effective through gap analysis to suggest that the Moving Target Defense technique is less complex than previous approaches to provide better security measures and protection against DDoS attacks on networks.**

Keywords*—IoT, MTD, Cyber-attacks, DDoS, Gap analysis.

I. INTRODUCTION

IoT devices have completely transfigured the whole world since they have stepped into the gates of technology to reside in the world of IT. Moreover, an exceptional level of connectivity began to outstretch into this world when IoT devices were integrated within the Software Defined Networking environments [1]. This remarkable integration didn't only lift up the ramping graph of advancements, it also stirred up everything in a much simplified and positive manner. Besides being a beneficial cause, this advancement led to outgrowing network security issues which raised a questionable concern in the field of cybersecurity.

SDN (Software Defined Networks) is a networking methodology that employs software-driven controllers or APIs to interact with the underlying hardware infrastructure, managing and directing network traffic [2]. The enormous number of vulnerabilities in SDN, limited resources, and the necessity for speedy rectification makes identifying criminals increasingly challenging for defenders and law enforcement agencies.

Published by MMU PRESS. URL: https://journals.mmupress.com/index.php/jetap/index

Among all of the network security issues, DDoS attack is one of the most rising issues in the domain of cybersecurity, as shown in Fig. 1 [3]. DDoS attacks are done by the predators for different sort of their gains, they actually overflow the traffic of a particular server so that they can overwhelm that infrastructure with abnormal traffic, disabling the access for the actual users to use that particular website. Before the launch of DDoS attack, an attacker always requires information regarding the point of entries to the targeted network (i.e. IP address of those targets). Just because of the static nature of the entry points in a targeted network, it becomes much easier for the attackers to attack smoothly. To mitigate those attacks, the cybersecurity professionals have lately started to adopt the swapping strategy known as MTD (Moving Target Defense) for the attacker's target points, ramping the effort scale of the attackers [4 - 8]. MTD involves continually changing network configurations so that attackers cannot obtain information about computer networks, making it more difficult to carry out targeted attacks.

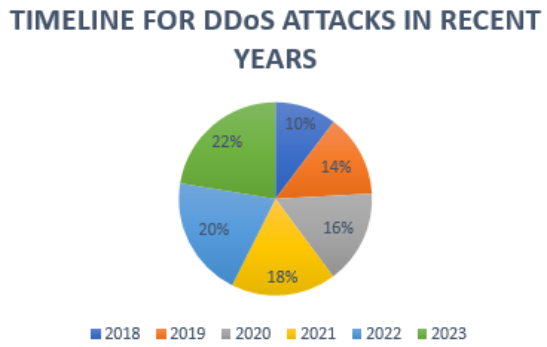
Moving Target Defense allows a proactive approach to swiftly change the attack surface of a system for introducing the unpredictability in the network of a system, hence mitigating the risk rate of attack, and escalating the rate of attacker's efforts [9].

Two years back, Cloud flare, which is such a significant CDN provider encountered a DDoS attack in July 2021 which targeted that network with a peak of 17.2 million requests per second. This attack was one of the most massive on record and was allocated to a botnet that exploited weaknesses in IoT devices of that network [10].

In August of 2021, T-Mobile, a major US cellphone provider, experienced a network disruption "DDoS Attack" that made the users unable to call all for several hours [11, 12]. The company has more than 110 million customers and during that attack, the data of 37 million people was exposed [13]. GitHub is a widely used platform of software development experienced DDoS in 2018 [14 - 16]. This attack caused the service to go down because of the continuous disruptions in the network for a time span of several days. This was one of the major and largest records at the time and this DDoS attack didn't involve any kind of bot because it was a Memcached DDoS attack, leading to a new strategy which was named as "Memcached reflection" [7].

II. GAP ANALYSIS

Different researchers opted for different methods and strategies to mitigate DDoS attacks in a system. Some of them focused on efficient methodologies, some of them opted for cost effective methods & very few of them opted strategical approaches to meet up with all of the expectations of the user. The main objective of this research paper is to shed the light on how to opt for an algorithm for DDoS mitigation by Moving Target Defense approach which will meet all of the expectations of the user by following the phenomenon of gap analysis. Through gap analysis, different algorithms have been assessed and compared from one another and the perfect one among them has been chosen.



Year	Millions
2018	7.09
2019	9.5
2020	10.8
2021	12.1
2022	13.9
2023	15.4

Fig. 1. Number of DDoS attacks in recent years [17].

Algorithms are assessed and compared on the basis of following parameters

- Efficiency
- Cost Effectiveness

Figure 2 demonstrates the flow of selecting algorithm on the basis of efficiency and cost effectiveness and Fig. 3 demonstrates how moving target defense mitigates the DDoS attack following the use cases.

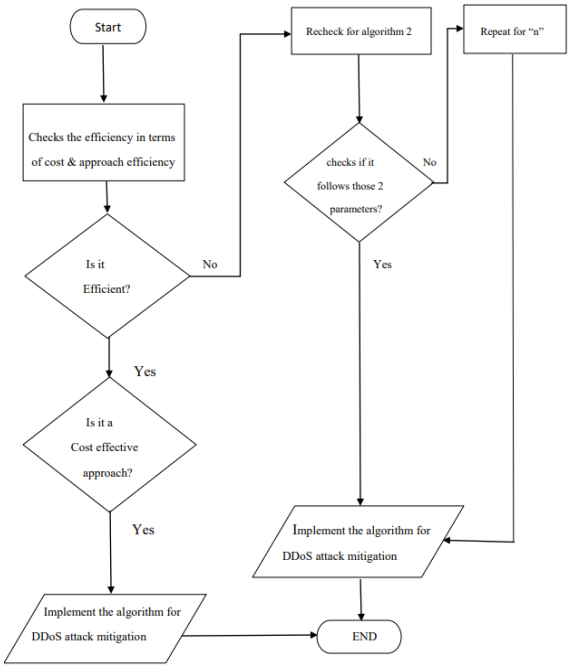


Fig. 2. Flow of selection strategy for best algorithm.

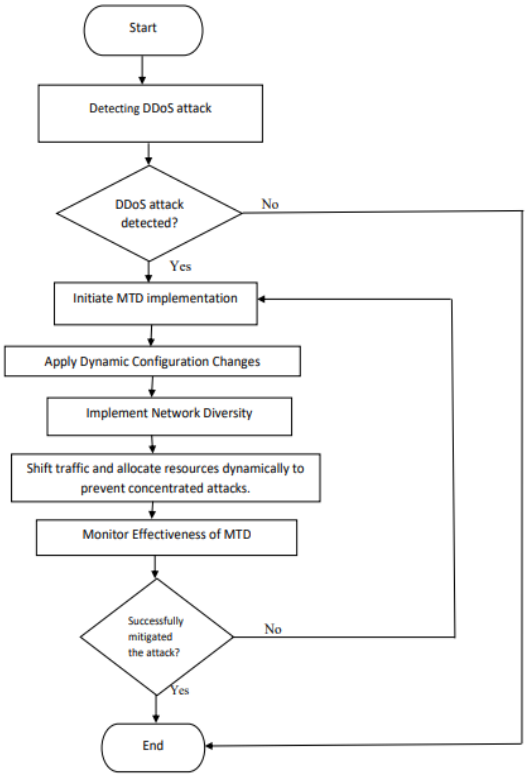


Fig. 3. Flow of Mitigation Strategy by Moving Target Defense Approach.

Algorithm 1: Proposed Detection and Mitigation Algorithm

Input: Training set, threshold h, thresholds θ_1, θ_2, number of devices d
Output: Detection and mitigation actions
$s_0 \leftarrow 0$, $t \leftarrow 0$;
for $n = 1$ **to** N **do**
  Partition training set into X^n_{M1} and X^n_{M2};
  Determine $L^n_{(\alpha)}$;
while $s_t < h$ **do**
  $t \leftarrow t + 1$;
  Get new data $\{X^n_t\}$ and compute $\{D^n_t\}$;
  $s^n_t \leftarrow \max(s^n_t + D^n_t, 0)$;
  $s_t \leftarrow \sum_{n=1}^{N} s^n_t$;
Declare attack at $T = t$;
for $n = 1$ **to** N **do**
  Compute $\bar{s}_n$;
  if $\bar{s}_n \geq \theta_1$ **then**
    for $j = 1$ **to** d **do**
      Compute $\bar{y}_{n,j}$;
      if $\bar{y}_{n,j} \geq \theta_2$ **then**
        Block traffic from device j;

Fig. 4. Detection and mitigation algorithm based on cumulative statistics and threshold analysis.

The algorithm highlighted in Fig. 4 which has been chosen above [17 - 18] follows the approach of traffic monitoring to block the malicious traffic.

III. WORKING STRATEGY

A. Algorithm 1: Proposed Detection and Mitigation Algorithm

1. Initialization

For the sake of initialization, it sets the values as;

- Initial value of s (statistic) to 0
- Time variable t to 0.

2. Training Set Partitioning

Nodes are represented by n, ranging from 1 to N.

- It breaks down the training set into two subsets: X^n_{M1} and X^n_{M2}.
- Finds the value of $\bar{L}^n_{(\alpha)}$ based on the partitioned training sets.

3. Detection and Mitigation Loop

For the purpose of detection and mitigation, the defined loop works for these conditions:

- While the cumulative statistic s_t is less than a predefined threshold h, increment the time variable t by 1.
- Obtain new data $\{ X^n_t \}$ and compute $\{ D^n_t \}$ based on the incoming data data.
- Update the node specific statistics s_n at time t using the formula: $s^n_t = \max \{s^n_t + D^n_t , 0\}$
- Calculate the camultative statistic s_t as a sum of s_n over all the nodes (from 1 to N)

4. Attack Declaration

For the declaration of attacks, it declares an attack at time $T = t$ when the cumulative statistic s_t exceeds the predefined threshold h.

5. Mitigation

As it works for the mitigation for DDoS attacks, it does the below listed works for each node n in the range from 1 to N:

- Calculate the average statistic $\bar{s}_n$ based on the collected data.
- If the average statistic $\bar{s}_n$ is greater than or equal to a predefined threshold θ_1

Now, for each device j in the range from 1 to d, it does the following tasks:

- Compute the average distance $\bar{y}_{n,j}$ for each device based on the collected data.
- If the average distance $\bar{y}_{n,j}$ is greater than or equal to a predefined threshold θ_2 Then block traffic from device j.

Table I. Notation table of algorithm.

NOTATION	DEFINITION
S, R	Sender and Receiver
θ ∈ Θ	Types
m ∈ M, a ∈ A	Messages and Actions
N	Number of servers in the network
ζn(t)	Belief of R for the n-th S at time step t
p(θ)	Prior Probability of Type θ
λS (m|θ)	Strategy of S with Type θ
λR(a|m)	Strategy of R given Message m
U X (θ, m, a)	Payoff Function of Player X, X ∈ {S, R}
gp, cp	Gain and cost of S for sending m = 1
gh, ch	Gain and cost of S for sending m = 0
ga, ca	Gain and cost of R for taking action a = 0

In summary, this algorithm firstly initializes variables, partitions the training set, continuously monitors upcoming data to detect the occurred changes, attack is declared when an attack is witnessed, and starts the mitigation procedure by blocking traffic from certain devices based on predefined thresholds, the details of algo is discussed in Table I.

Algorithm 2: Optimal Selection Algorithm Based on Game Theory

Input: Server types $\theta(t)$, logs $l(t)$, game parameters gp, cp, gh, ch, ga, ca
Output: Decision set $D(t) = \{(d_1, \sigma_1), ..., (d_N, \sigma_N)\}$
Initialize signaling game at $t = 1$;
Formulate the payoff matrix;
for $t = 1$ **to** T **do**
  Compute $\zeta(t)$;
  for $i = 1$ **to** N **do**
    if $0 \leq (1 - \theta_i)l_i \leq Np(\theta_i)$ *and* $\theta_i l_i = 0$ **then**
      $d_i \leftarrow$ IP Randomization; $\sigma_i = 1$;
    else if $|\theta_i l_i - l_i| \geq \epsilon$ *and* $\theta_i = 0$ **then**
      $d_i \leftarrow$ Dynamic Redirection;
      $\sigma_i \leftarrow$ Eq. 11;
    else if $|\theta_i l_i - l_i| \geq \epsilon$ *and* $\theta_i = 1$ **then**
      $d_i \leftarrow$ Response Time Adaptation;
      $\sigma_i \leftarrow$ Eq. 15;
    else
      $D(t) = D(t - 1)$;
return $D(t)$;

Fig. 5. Optimal selection algorithm for defensive actions using signaling game strategy.

B. Algorithm 2: Optimal Selection Algorithm Based on Game Theory

The algorithm in Fig. 5 has been chosen to implement the strategy for optimal selection [17]. Further details are in Table II.

Table II. Payoff matrix for different actions for the two players.

Type & Message	Player	Action	
		a=0	a=1
θ = 0, m = 0	S	−gp	0
	R	ga − ca	−gp
θ = 0, m = 1	S	−gp − cp	gp − cp
	R	ga − gp − ca	−gp
θ = 1, m = 0	S	gh − ch	−ch
	R	−ca	0
θ = 1, m = 1	S	gh	0
	R	−gh − ca	0

The payoff matrix depicts the payoffs (rewards or costs) for all the possible set of actions which are acted by the 2 players in that game. By considering a case [6], the 2 players are the sender (S) and the receiver (R) in a signalling game scenario. The matrix has been divided into 4 rows, each representing a different combination of the sender's type (θ) and message (m), and two columns representing the receiver's possible actions (a = 0 or a = 1). For example, if the sender is of type $\theta = 0$ and sends message $m = 0$, and the receiver takes action a = 0, the payoff for the sender is -gp(a negative value indicates 'a' cost), and the payoff for the receiver is 0 (no reward or cost). On the other hand, if the sender is of type $\theta = 0$ and sends message $m = 1$, and the receiver takes action a = 1, the payoff for the sender is gp-cp (a positive value indicating a reward), and the payoff for the receiver is ga-gp-ca (a negative value indicating 'a' cost). The main objective of this payoff matrix is to let the players decide the best set of actions through the analysis of matrix, maximizing the expected payoffs in the game on the basis of rewards and costs.

This algorithm is "Optimal Strategy Selection Algorithm" which is structured to find the best defensive actions and their related probabilities for each and every server in a signaling game scenario. The simplest breakdown explanation of the algorithm can be explained as:

- The types of servers can be represented by θ in the network at time t:
  Server 1= θ_1, Server 2 = θ_2, ……. Server N = θ_N
  S = {θ_1, θ_2, …….. θ_N}
- Status logs of servers $l(t)$ shows the security status (if it is under attack or not) of each and every server at time t:
  $l(t)$ = {l_1, l_2, …….. l_N}
- The game parameters are assumed for the gains and costs related to various action.
  Parameters = gp, cp, gh, ch, ga, ca

The gamer's set of decision with probability for each and every situation will be $D(t)$ = (d, σ), where d is decision and σ is probability.
$D(t)$ = {(d_1, σ_1), (d_2, σ_2), ...(d_N , σ_N)}

1. Initialization
It initializes the signalling game by giving input at t = 1 & then it sets up the initial Conditions for the signalling game.

2. Formulating the Payoff Matrix
After then, it creates a matrix to depict the payoffs for various combinations of actions taken by the sender and receiver.

3. Iteration Then, iteration begins for each time step within the range of t = [1…….. T}.

4. Calculation of ζ(t)
After the iteration of time steps, it calculates the parameter ζ(t) which is dependent on that current time step & the kind of servers which are present at that moment. After calculating this parameter, this will be involved in further steps of the calculation.

5. Checking the Server's Status
Afterwards, it checks the status of the server i (from 1 to N) and determines the type of server for the determination of suitable action & related probability.

6. Conditions

- If server is honeypot (θ_i = 0) and the server's security is not violated by any attacker yet, perform IP jumbling/ shuffling to baffle the attacker and set probability of attacker as '1', hindering the DDoS attack.
- If server is not honeypot and it is an actual(real) server (θ_i = 1) & the difference between the no. of violated real servers and the number in total of real servers is greater than or equal to threshold ε, then start to redirect the hosts dynamically to relocate attacker's focus to honeypots. After then, do calculation for the associated probability σ_i based on the parameters of the game and ζ(t).
- If the mentioned condition does not occur, then perform Response Time Adaptation so that the access of attacker to the real server gets delayed and do the same calculations for σ_i and ζ(t), as mentioned above.
- If nothing happens as mentioned in the above conditions, then perpetuate the previous decision *D(t) = D(t-1).*

7. Result

Lastly, it returns the sets of decisions with probabilities for each time step in the form of: {(d_1, σ_1), (d_2, σ_2), ... (d_N , σ_N)}. And the details are in Algorithm 3.

C. Algorithm 3: RYU Controller Packet Processing

Algorithm 3: RYU Controller Packet Processing

Input: Network packet *pkt*
if *pkt is DNS request* **then**
  Change $rIP \rightarrow vIP$; Map $\{rIP, vIP\}$;
if *pkt is TCP or UDP* **then**
  if *pkt.dest is rIP connected directly* **then**
    Forward to rIP;
  if *pkt.dest is vIP* **then**
    if *pkt.src is rIP* **then**
      $pkt.srcIP \leftarrow vIP$; Map $\{rIP, vIP\}$;
    Refer IP mapping; Install flows in OF-switches;

Fig. 6. RYU controller-based handling for Moving Target Defense.

The algorithm 3 in Fig. 6 is designed for the facilitation of dynamic handling and manipulation of network traffic within a SOFTWARE DEFINED NETWORK environment by providing flexibility to manage DNS requests, TCP/UDP packets, and IP address mappings through the Ryu controller and OF switches.

Figure 7 demonstrates how the RYU controller manages the communication between hosts by handling DNS requests, IP address mappings, and packet routing, thereby enabling the implementation of the proposed Moving Target Defense technique [15, 18]. This algorithm is for the implementation of the host address mutation technique which can be explained as following:

Step 1
If the checked packet is found to be a DNS request, then the algorithm starts to check the DNS response. DNS response provides the mapping between domain names and IP addresses and then it changes the real (original) IP address *rIP* to a virtual address *vIP*. The mapping is then stored in a table as *Map {rIP, vIP}*.

Step 2
If the network packet is a packet of TCP/UDP protocol, then the algorithm starts to verify if the IP address of the destination is a real IP address *rIP* or not. If it is found to be real then the network packet is sent towards it.

Step 3
If in case, the IP address of that destination is found to be a virtual one *vIP*, then the algorithm verifies the originality of source IP address (if it is real or not). If the source IP address is found to real one then the algorithm varies the IP address of the source to corresponding *vIP* and a kind of mapping is created in between the real and virtual IP Address.

Step 4
The algorithm then checks out the table that shows the mapping between IP addresses. Then it consumes this information to configure the correct paths in OF switches that takes towards the destination.

Step 5
If the network packet is not found to be a DNS request, TCP, or UDP protocol packet then the algorithm starts to drop the packet.

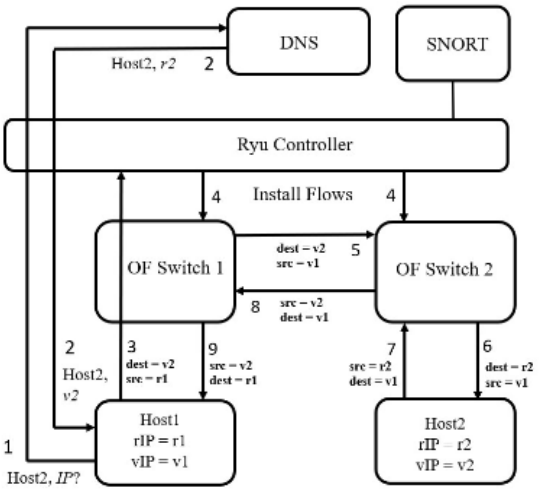


Fig. 7. Flow of the proposed detection and mitigation algorithm.

D. Algorithm 4: Shuffle Degree Calculator

1. Initialization
The algorithm fir SDC module shown in Fig. 8 starts with SDC that initializes a list of *H* zeros to store the n_e (neighbour edges) for each host and a variable to store the sum of all the members in n_e.

2. Calculation of Neighbour Edges

The algorithm then starts to check through each and every host in the network to calculate n_e for each host. Then, it iterates through the connections and increases the neighbour edge count by incrementing it for the host as per the connections it has with the other network.

3. Calculation of Shuffling Degrees
After the calculation of n_e for each host, the algorithm then calculates the shuffling degree for each of it by dividing the neighbour edge count by sum of all of the neighbour edges in the network.

4. Sending Shuffling Degrees to SID Module
The list of shuffling degrees calculated for each and every host is then transferred to the SID module and it uses this information to find the hosts that have to be shuffled in each and every interval of shuffling.

5. Summary of Algorithm
This Algorithm Actually calculates the neighbour edges for each host and ratio of each host with the total connections so that SID Module can be contained further.

Algorithm 4: Shuffle Degree Calculator (SDC Module)

Input: Connection matrix $c_{i,j}$, hosts H, range S
Output: Shuffle degree vector d
$ne \leftarrow$ list of H zeros; $sum \leftarrow 0$;
for $i = 1$ **to** H **do**
  for $j = H + 1$ **to** $H + S + 1$ **do**
    if $c_{i,j} = 1$ **then**
      $ne[i] \leftarrow ne[i] + 1$; $sum \leftarrow sum + 1$;
for $i = 1$ **to** H **do**
  $d[i] \leftarrow ne[i]/sum$;

Fig. 8. Algorithm for SDC module for calculating shuffling probability of hosts.

E. Algorithm 5: Shuffle Interval Detector

The algorithm 5 in Fig. 9 which has been chosen above is for the implementation of shuffle degree calculator module procedure [15]. The above SID Algorithm can be broken down into following steps for better clarity of the working of whole algorithm altogether.

1. Initialization
This SID algorithm starts to initialize an empty list known as *top* to store/keep the $\mu + \rho$ highest degree hosts. The values of μ and ρ are preliminary defined parameters that finds the number of hosts which have to be shuffled in each shuffling interval.

2. Searching for Highest Degree Host
The algorithm then checks through each and every host in the network to find the host which has the highest degree. If the host is not actually in the list of "top" , the algorithm then checks to see if its degree is greater than the present max. degree. And if it is found over there, then the host is added to the "top" list.

3. Shuffling the Set of Host
For each interval of shuffling, the algorithm stores an empty list as λ to store the hosts that have to be shuffled. The algorithm then checks through all the hosts in the network and starts to generate a number randomly between 0 and 1. If the randomly generated number is less than the shuffling degree of the host, then the host is summed up to the λ list.

4. Transfer of Set of Hosts to FEG Module
The "λ" list is then sent to the FEG module to set the linked flow entries. The OpenFlow message that indicates flow entry timeout is as: OFPT_FLOW_REMOVED.

5. Summary of Algorithm
This algorithm does iteration after creating an empty list to store the results. This process is for checking the host with the most highest neighbour edges repeatedly.

Algorithm 5: Shuffle Interval Detector (SID Module)

Input: Degrees $d[i]$, hosts H, params μ, ρ
Output: List λ of hosts to shuffle
$top \leftarrow \emptyset$;
while $|top| < \mu + \rho$ **do**
  $max \leftarrow -1$;
  for $i = 1$ **to** H **do**
    if $i \notin top$ ***and*** $(max = -1$ ***or*** $d[i] > d[max])$ **then**
      $max \leftarrow i$;
  Add max to top;
foreach *shuffling interval* **do**
  $\lambda \leftarrow \emptyset$;
  for $i = 1$ **to** H **do**
    $r \leftarrow$ random number in $[0, 1]$;
    if $r < d[i]$ **then**
      Add i to λ;

Fig. 9. Algorithm for SID module for selecting hosts based on shuffling degrees.

Algorithm 6: Integrated SDC-SID Control Algorithm

Input: Shuffle degrees $d[i]$, hosts H, parameters μ, ρ
Output: Selected host list λ
$top \leftarrow \emptyset$;
while $|top| < \mu + \rho$ **do**
  $max \leftarrow -1$;
  for $i = 1$ **to** H **do**
    if $i \notin top$ ***and*** $(max = -1$ ***or*** $d[i] > d[max])$ **then**
      $max \leftarrow i$;
  Add max to top;
foreach *shuffling interval* **do**
  $\lambda \leftarrow \emptyset$;
  for $i = 1$ **to** H **do**
    $r \leftarrow$ random number in $[0, 1]$;
    if $r < d[i]$ **then**
      Add i to λ;

Fig. 10. Integrated control mechanism combining SDC and SID for adaptive host shuffling in MTD systems.

F. Algorithm 6: Integrated SDC-SID Control Algorithm

Overall, the SID algorithm in Fig. 10 works after integrating it with SDC Module. SDC Module calculates n_e and d for the further computation of SID Module to shuffle the hosts one by one by giving high priority to the hosts with higher number of neighbour edges as .

IV. TIME COMPLEXITY OF ALGORITHM USING BIG O NOTATION

For the sake of comparitive analysis of above 5 algorithms, they can be compared with each other on the basis of their time complexities using big O notation. Afterwards, their cost effectiveness is also assessed on their time complexities.

A. Time Complexity of Proposed Detction and Mitigation Algorithm

The time complexity of algorithm for proposed detection and mitigation can be calculated by using Big O notation as follow:

1. From Line 1 to Line 5
The iteration of 2nd to 5th line runs for N times. Here, N is actually the size of the set. Some of the computations inside the loop are independent on the data size like the calculation of: $\bar{L}^{n}_{(\alpha)}$. Hence, this part's time complexity is O(N).

2. From Line 6 to Line 11
The iteration of while loop is continued until unless the specified condition is met. In this part, the time complexity is not dependent on data but it is actually dependent on the number of iterations repeated in a certain period of time. If T is denoted as the number of iterations then the time complexity will be O(T).

3. From Line 12 to Line 23
In these lines of the final loop, iteration is done for N times.Inside the final loop, the nested loops are from line 16 to line 12 which will be iterated according to data dimension d. Therefore, the time complexity of this part is O(N*d). Overall time complexity is

$$[O(N) + O(T) + O(N * d)]$$

B. Time Complexity of Optimal Selection Algorithm

The time complexity of algorithm for optimal selection can be calculated by using Big O notation as follows.

1. Initialization
Initializing of variables and the formation of pay off matrix are constant time computations, so the time complexity of this part is O(1).

2. Outermost Loop
The outer most loops run for T times. Here, T is the number of total time steps from T=1 to T. So, the time complexity will now be O(T)

3. Innermost Loop
This loop run for N times. Here, N is the number of total servers. Since the inside calculations of this loop take constant time so the time complexity will now be O(N)

4. Overall Time Complexity
By combining the time complexity of overall algorithm, it can be concluded that the overall complexity is O (T * N)

C. Time Complexity of RYU Controller ALgorithm

The time complexity of algorithm for Ryu Controller can be calculated by using Big O notation as follow:

1. DNS Request
The DNS request processing takes constant time. That's why, the time complexity of this part is O (1)

2. TCP and UDP Processing
This part also consumes constant time and hence the time complexity of this part is stated as O (1)

3. Overall Time Complexity
The overall time complexity can be concluded as O (1) since the computations take constant time and none of them depend on the size of data of input packets or the total number of input packets.

D. Time Complexity of SDC and SID Integrated Algorithms

The overall time complexity of Shuffling Degree Calculator algorithm after step by step computations can be concluded as

$$O(H + H * S) \quad (1)$$

The overall time complexity of Shuffling Interval Detector algorithm after step-by-step computations can be concluded as

$$O(\mu + \rho + H) \quad (2)$$

So now, the overall time complexity of the integrated algorithm can be stated by adding the time complexity of algorithm for SDC and SID. By adding Eq. (1) and Eq. (2), time complexity of integrated algorithm is

$$O(H * S + \mu + \rho + H) \quad (3)$$

, where

H = total number of hosts
S = size of 2nd dimension of matrix
μ = size of the top list
ρ = constant value at 5

Hence, the time complexity is affected by the matrix size and the size of the *top* list.

E. Comparitive Analysis Of Algorithms

1. If the time steps (T) and input size of data (N) are in a large amount, algorithm for optimal selection: O (T * N) might have a higher complexity compared to Algorithm for Proposed Detection and Mitigation: O(N) + O(T) + O(N*d). However, the

algorithm for Proposed Detection and Mitigation is still not that much time efficient since it depends on the size of input data.

2. Algorithm for Ryu controller: O (1) has the highest efficiency as far as time complexity is concerned because it is independent on the size of input data.
3. Integrated Algorithm SDC and SID: Eq. (3) is linearly dependent on hosts and matrix, and along with that, its efficiency also depends on the values of μ and ρ.

F. Cost Effectiveness

The cost of the implementation of any algorithm depends on how much time complexity it holds. The higher the time complexity, the more cost it will require and vice versa. So according to this analysis, it can be stated that algorithm 3 is the most cost-effective algorithm.

V. RESULTS

A. Results of Time Complexity using Big O Analysis

The algorithms are assessed on the basis of time complexity analysis by using Big O notation to represent it and then the further observations are carried out from it to prove cost effectiveness of that specified algorithm.

The results display the time complexities of different algorithms by considering both the larger data set of hosts (infinite) and smaller data set of hosts (finite) as in Table III and Table IV. The most preferable algorithm which will be time efficient for future cases is denoted by "☑" and the one which won't be time efficient is denoted by "☒'. For this analysis, "**t_s**" represents less value of time steps in an algorithm.

Table III. Time complexity for low size of input data if H = finite.

Algorithm	Time Complexity
Proposed Detection and Mitigation	☑
Optimal Selection	☒
RYU Controller	☑
SDC & SID Integrated Algorithm	☒

Table IV. Time complexity of large size of input data if H = infinite.

Algorithm	Time Complexity
Proposed Detection and Mitigation	☒
Optimal Selection	☒
RYU Controller	☑
SDC & SID Integrated Algorithm	☒

B. Cost Effectiveness using Predictive Analysis

The results display the cost effectiveness using the predictive analysis of different algorithms by considering both the larger data set of hosts (infinite) and the smaller data set of hosts (finite). The most preferable algorithm, which will be cost-effective for future cases, is denoted by "☑" and the one which won't be cost-effective is denoted by "☒" as in Table V and Table VI.

The predictive analysis for cost effectiveness is basically carried out by having a look at the directly proportional relation of time efficiency with cost effectiveness since more time leads to the usage of more resources, either it is materialistic consumption or any other. For this predictive analysis, t_s represents a lesser value of time steps in an algorithm.

Table V. Cost Effectiveness of low size of input data if H = finite and time steps value = t_s.

Algorithm	Cost effectiveness
Proposed Detection and Mitigation	☒
Optimal Selection	☒
RYU Controller	☑
SDC and SID Integrated Algorithm	☒

Table VI. Cost Effectiveness of large size of input data if H = infinite and time steps value = t_s.

Algorithm	Cost Effectiveness
Proposed Detection and Mitigation	☑
Optimal Selection	☒
RYU Controller	☑
SDC & SID Integrated Algorithm	☒

C. Aligned Results of Efficiency and Cost Effectiveness

Table VII. Cost Effectiveness and Time Efficiency of low size of input data if H = finite.

Algorithm	Time Efficiency	Cost Effectiveness
Proposed Detection and Mitigation	☑	☑
Optimal Selection	☒	☒
RYU Controller	☑	☑
SDC and SID Integrated Algorithm	☒	☒

Table VIII. Cost Effectiveness and Time Efficiency of large size of input data if H = infinite.

Algorithm	Time Efficiency	Cost Effectiveness
Proposed Detection and Mitigation	☒	☒
Optimal Selection	☒	☒
RYU Controller	☑	☑
SDC and SID Integrated Algorithm	☒	☒

The efficiency and cost-effectiveness of the evaluated algorithms are compared based on different input data sizes. When the number of hosts H is relatively small (finite), the results are shown in Table VII, highlighting that the RYU controller performs best in both time and cost dimensions. In contrast, for larger-scale input data (infinite H), the aligned outcomes are summarized in Table VIII, where the

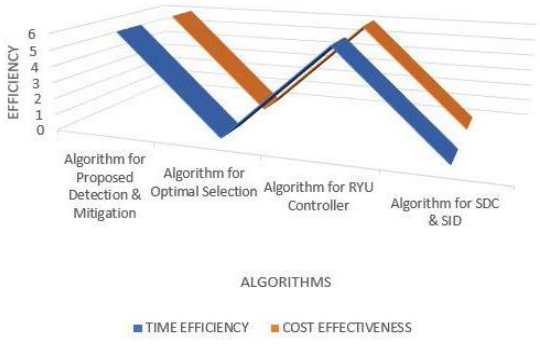


Fig. 11. Graphical comparison of time efficiency and cost effectiveness if set of hosts = finite.

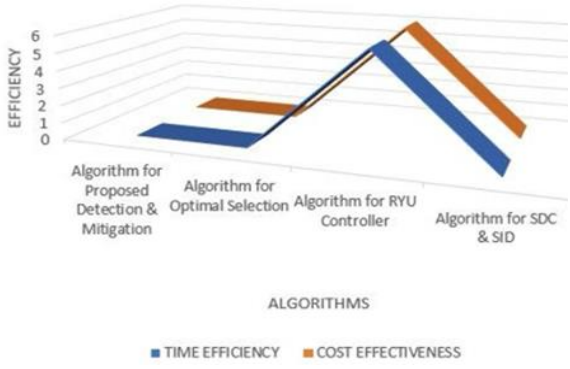


Fig. 12. Graphical comparison of time efficiency and cost effectiveness if set of hosts = infinite.

integrated SDC and SID algorithm demonstrates a more balanced trade-off.

Further based on the graphical representation of the comprehensive analysis in Fig. 11 and Fig. 12, the conclusion drawn is that the RYU Controller algorithm stands out as an optimal choice for both time efficiency and cost effectiveness, even when dealing with an infinite range of input data (hosts). The visual depiction of the results indicates that the RYU controller algorithm consistently outperforms others across various scenarios.

It is essential to acknowledge that the assessment of time complexity and cost effectiveness of any specific algorithm is contingent upon a myriad of conditions and scenarios. The intricacies of real-world applications, varying data sets, and diverse operational environments contribute to the nuanced performance of algorithms. While RYU Controller exhibits notable performance in the context of this study, it's crucial to consider the adaptability of algorithms to specific use cases and the potential impact of different operational parameters. By assuming that Efficient = 6 & non-efficient = 0.

In summary, the choice of the most suitable algorithm extends beyond a one-size-fits-all approach. It necessitates an understanding of the specific requirements, constraints, and conditions within the intended application domain

VI. CONCLUSION

This gap analysis has been done to determine the gap between different research to verify the efficiency of various algorithms for the determination of the most efficient and cost-effective algorithm. All this research work has been done to prove that Moving Target Defense strategy is the most effective and efficient strategy for the mitigation of DDoS attacks as compared to other methods. This study comprises of the comparative analysis of four algorithms of Moving Target Defense strategy and one algorithm of optimal selection to reduce or block the unwanted traffic on the user's system. Each of the algorithms has been assessed on the basis of Time Complexity Analysis by using Big O notation to represent the results. Furthermore, the Big O notations of each and every algorithm are compared altogether to find the most cost-effective algorithm amongst all of them. Overall, the major objective of this research work was to analyze the backend working of DDoS attack's mitigation to choose an algorithm for the implementation of Moving Target Defense strategy successfully by keeping the efficiency and cost-effective criteria in mind.

ACKNOWLEDGEMENT

The authors would like to thank NED UET and MMU for its support of their resources, and the editorial team of JETAP for their valuable feedback during the review process.

REFERENCES

[1] Y. Zhou, G. Cheng and S. Yu, "An Software Defined Network-Enabled Proactive Defense framework for DDOS Mitigation in IoT Networks," *IEEE Trans. Informat. Forens. and Secur.*, vol. 16, pp. 5366–5380, 2021.

[2] L. F. Eliyan and R. Di Pietro, "DoS and DDoS Attacks in Software Defined Networks: A Survey of Existing Solutions and Research Challenges," *Future Generat. Comput. Syst.*, vol. 122, pp. 149-171, 2021.

[3] F. Nabi, X. Zhou, U. Iftikhar and H. M. Attaullah, "A Case Study of Cyber Subversion Attack based Design Flaw in Service Oriented Component Application Logic," *J. Cyber Secur. Technol.*, vol.8, no.3, pp. 204-228, 2024.

[4] H. Galadima, A. Seeam, Amar and V. Ramsurrun, "Cyber Deception Against DDoS Attack using Moving Target Defense Framework in Software Defined Network IoT-Edge Networks,". in *3rd Int. Conf. Next Generat. Comput. Appl.*, Flic-en-Flac, Mauritius, pp. 1-6, 2022.

[5] R. L. Hemanth Kumar, K. P. Bhargava and A. R. Ashok Kumar, "Mitigation and Detection of DDOS Attacks using Software Defined Network (SDN) and Machine Learning," *Int. J. Res. in Appl. Sci. and Eng. Technol.*, vol. 11, no. 4, pp. 4821–4829, 2023.

[6] M. M. Oo, S. Kamolphiwong and T. Kamolphiwong, "The Design of SDN Based Detection for Distributed Denial of Service (DDoS) Attack," in *21st Int. Comput. Sci. and Eng. Conf.*, Bangkok, Thailand, pp. 1-5, 2017.

[7] H. M. Attaullah, S. Memon, O. F. Erkan and R. Khawar, "IoT Based Systems and Services: Recent Security Concerns and Feasible Solutions," *IEEE 1st Karachi Sect. Human. Technol. Conf.*, pp. 1-6, 2024.

[8] H. M. Attaullah, R. A. Khan and S. Mughal, "Cyber Security for Industrial Control System–A Survey," *IKSP J. Emerg. Trends in Basic and Appl. Sci.*, vol. 1, pp. 15–21, 2021.

[9] K. Doshi, Y. Yilmaz and S. Uludag, "Timely Detection and Mitigation of Stealthy DDoS Attacks via IoT Networks," *IEEE*

Trans. Depend. and Secur. Comput., vol. 18, no. 5, pp. 2164-2176, 2021.

[10] M. S. B. Syed, H. M. Attaullah, S. Ali and M. I. Aslam, "Wireless Communications Beyond Antennas: The Role of Reconfigurable Intelligent Surfaces," *Engi. Proc.*, vol. 32, no. 1, pp. 10, 2023.

[11] K. S. Vanitha, S. V. Uma and S. K. Mahidhar, "Distributed Denial of Service: Attack Techniques and Mitigation," in *2017 Int. Conf. Circ., Contr., and Commun.*, Bangalore, India, pp. 226-231, 2017.

[12] A. Javadpour, F. Ja'fari, T. Taleb and M. Shojafar, Mohammad, "A Cost-Effective Moving Target Defense Approach for DDoS Attacks in Software-Defined Networks," in *2022 IEEE Global Commun. Conf.*, Rio de Janeiro, Brazil, pp. 4173-4178, 2022.

[13] O. Yoachimik, "Cloudflare Thwarts 17.2M rps DDoS Attack - The Largest Ever Reported," *The Cloudflare Blog*. [Available online on 19 August 2021] https://blog.cloudflare.com/cloudflare-thwarts-17-2m-rps-ddos-attack-the-largest-ever-reported.

[14] K. S. Kumavat and J. Gomes, "Survey of Detection Techniques for DDoS Attacks," in *3rd Int. Conf. Intellig. Eng. and Manage.*, London, United Kingdom, pp. 657-663, 2022.

[15] C. Gudla and A. H. Sung, "Moving Target Defense Discrete Host Address Mutation and Analysis in SDN," in *Int. Conf. Comput. Sci.and Comput. Intellig.*, Las Vegas, NV, USA, pp. 55–61, 2020.

[16] C. Douligeris and A. Mitrokotsa, "DDoS Attacks and Defense Mechanisms: A Classification," in *Proc. 3rd IEEE Int. Symp. Signal Process. and Informat. Technol.,* Darmstadt, Germany, pp. 190-193, 2003.

[17] S. Venkatesan, M. Albanese, K. Amin, S. Jajodia and M. Wright, "A Moving Target Defense Approach to Mitigate DDoS Attacks Against Proxy-based Architectures," in *IEEE Conf. Commun. and Netw. Secur.*, Philadelphia, PA, USA, pp. 198-206, 2016.

[18] T. Mehmood, H. M. Attaullah, M. Ibrahim and M. B. J. Al Shehry, "Securing AGI-Driven Drone Communications for Climate Change: A Comprehensive Review of Deep Learning-Based IDS," *Artificial General Intelligence-Based Drones for Climate Change*, pp. 97-152, IGI Global Scientific Publishing, 2025.