ISSN: 2366-1313

# DDOS ATTACK DETECTION AND MITIGATION USING MACHINE LEARNING

**[1]Mrs.B.Priyanka,[2]P.Adhithi,[3]S.Murali Krishna,[4]V.Sanjay**

[1]Assistant Professor, Dept. of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad,

priyankabhupathi90@gmail.com

[2, 3, 4, BTech] Student, Dept. of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad

adithipippera@gmail.com,sakethamuraliKrishna9@gmail.com,velugurisanjay@gmail.com

## ABSTRACT:

Distributed Denial of Service (DDoS) assault represents a malevolent effort to disrupt the typical flow of traffic to a targeted server, service, or network by inundating it with a deluge of Internet data. DDoS attacks stand out as one of the most perilous and menacing threats in the realm of networking, capable of inundating networks and impeding access to server resources by bombarding them with an overwhelming volume of packets, thereby depleting network assets and hindering responses to legitimate requests. Particularly in cloud environments, the potency of DDoS attacks tends to amplify. Addressing this challenge necessitates a multifaceted approach, one that melds statistical and machine learning techniques to effectively identify and counteract DDoS onslaughts within Software-Defined Networking (SDN) frameworks. This methodology finds its realization through the integration of the Ryu controller and Mininet network simulator, leveraging the Open Flow SDN protocol. The array of devices involved spans from conventional computers to diverse network resources, including Internet of Things (IoT) devices. Conceptually akin to an unforeseen traffic gridlock, a DDoS incursion erects barriers along the digital highway, impeding the smooth flow of legitimate traffic towards its intended destination. Swift action upon detection of a DDoS strike assumes paramount importance, offering a window of opportunity to circumvent widespread disruption. Prolonged inaction risks server failure, potentially culminating in protracted recovery periods extending over hours. The crux of DDoS mitigation lies in its intricate dance with legitimate traffic, which proves exceedingly challenging to discern and intercept. Malicious actors invest significant effort in camouflaging spoofed traffic to mimic genuine communications, thereby confounding traditional defense mechanisms. For enterprises operating at the precipice, where uninterrupted functionality constitutes an imperative, DDoS mitigation emerges as a linchpin in the defensive architecture. It serves as a bulwark against downtime, ensuring the continuous availability of mission-critical services and functionalities. SDN, heralding a paradigm shift in network orchestration, furnishes the scaffolding for agile network planning, construction, and operation. However, this technological landscape is not impervious to incursions, with DDoS threats looming large, particularly within the purview of SDN networks

**Keywords*:***Distributed Denial of Service (DDoS), Software-Defined Networking (SDN).

## I INTRODUCTION

Cloud computing has emerged as a dominant force in both industry and academia, offering a plethora of benefits compared to traditional networking paradigms. Within this landscape, software-defined networks (SDNs) have surged in popularity, revolutionizing network management and performance. By decoupling the data and control planes of network devices, SDN empowers centralized administration and programmability, facilitated through an SDN controller. Comprising three layers, the SDN architecture includes the infrastructure layer housing networking devices such as switches and hosts. SDN streamlines network monitoring and management, enabling seamless addition, removal, and configuration of network devices from a centralized location. This approach augments flexibility, scalability, and cloud-based administration while enhancing controllability and performance. In the context of cloud computing networks, SDN-based cloud environments have become instrumental in offering networking as a service, bolstering network security and governance.

## II. LITERATURE SURVEY

**1. Machine Learning-Based DDoS Detection in SDN Networks**

This research introduces a model designed to automatically detect and mitigate DDoS attacks within SDN networks using machine learning (ML) techniques. The model periodically gathers traffic flow entries from all switches, extracting native flow features and enhancing them by adding new features. A detection module employs five features to classify each flow as normal or anomalous, swiftly blocking the source of any identified attack.The study evaluates six ML algorithms. Including Logistic Regression (LR), Naive Bayes (NB), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Trees (DT), and Random Forest (RF). Results indicate that RF outperforms other classifiers, demonstrating superior performance in accurately identifying and mitigating attacks while maintaining typical network functionality.

**2. DDoS Detection Using Deep Learning Techniques:**

This approach proposes a method for DDoS attack detection utilizing deep belief network feature extraction and Long Short-Term Memory (LSTM) models. Deep learning techniques are leveraged to extract attributes from IP packets, enabling the construction of an LSTM traffic prediction model. This model is then employed to identify DDoS attacks by detecting irregularities in network traffic patterns. The proposed technology demonstrates efficacy in forecasting typical traffic patterns, detecting deviations indicative of DDoS attacks, and lays the groundwork for future advancements in DDoS detection methodologies.

**3. Correlation Analysis-Based DDoS Detection in Data Centers**:

In this study, the authors present a model that analyses correlation information among flows within data centers to detect DDoS attacks. The model employs a method known as CKNN (K-nearest neighbors traffic categorization with correlation analysis) to provide a robust and reliable means of identifying DDoS attacks. By examining the interplay between different flow patterns, the model enhances its ability to discern anomalous network behavior characteristic of DDoS attacks, thereby bolstering network security and resilience.

**4. SVM-Based DDoS Detection in SDN:**

This research proposes a novel model for DDoS attack detection in SDN environments, utilizing Support Vector

Machine (SVM) algorithms. The model employs a trained SVM approach to extract crucial features from packet-in messages and assesses the distribution of each feature using entropy measures. Experimental findings demonstrate the effectiveness of this technique in real-time DDoS mitigation and security event detection, showcasing its potential as a robust defense mechanism against DDoS attacks within SDN architectures.

## III SYSTEM ANALYSIS

### EXISTING SYSTEM

Compared with other DDoS[5]detection methods, detecting by entropy is proved to have many advantages, such as simpler, higher sensitivity, lowewrate of false positives. Many attackers in different locations continuously send a great deal of packets at the same time, which is out of the target device's processing ability, making the legitimate user out of service. This enables attackers can launch DDos attacks towards SD network from multiple layers. It works well when the attack traffic is very huge.

**Limitations of Existing system**

The sub channels are allocated to optimize throughput and maintain the proportional stream rate ratios. It may cause bottleneck problem in the wireless network. Sometimes it leads to network shut down with whole date loss. ThePerformanceconsequently depends on the scheduling strategy employed

### PROPOSED SYSTEM

The proposed system revolves around the implementation of a DDoS attack detection framework utilizing Random Forest computing within a Software Defined Networking (SDN) environment. The framework is equipped with the Ryu controller and Mininet emulator within the Linux framework.

**Framework Components:**

**Ryu Controller**: The Ryu controller plays a pivotal role in orchestrating operations and facilitating communication between SDN switches and monitoring computations. It acts as the central hub for network management and control.

**Mininet Emulator:** Mininet serves as a simulation platform, enabling the creation of a realistic network environment for testing and evaluation purposes. It allows researchers to emulate various network scenarios and assess the performance of the proposed DDoS detection framework under different conditions.

**Random Forest Computing:** The heart of the proposed system lies in the utilization of Random Forest computing for DDoS attack detection. Random Forest is chosen for its ability to handle large datasets efficiently and provide robust classification performance. By leveraging ensemble learning techniques, Random Forest can effectively capture both normal and anomalous patterns of behavior within network traffic data.

**Proposed system Advantages:**

**Dynamic Adaptation:** The proposed framework harnesses the programmability and centralized control capabilities of SDN to dynamically adapt to evolving deployment conditions and emerging attack strategies. This adaptability enables the system to stay ahead of sophisticated DDoS attacks and ensure robust network security.

**Compound Learning Strategy:** The use of Random Forest algorithm incorporates a compound learning strategy, enhancing the accuracy and versatility of detecting operational information organization anomalies. This approach improves the system's ability to identify subtle deviations indicative of DDoS attacks, minimizing false positives and negatives.

**Isolated Design for Extensibility:** The framework's isolated design facilitates adaptability and extensibility, allowing for the seamless integration of additional location algorithms or mitigation procedures in the future. This modular architecture ensures that the system can evolve alongside emerging threats and security requirements.

## IV IMPLEMENTATION

**Architecture:**

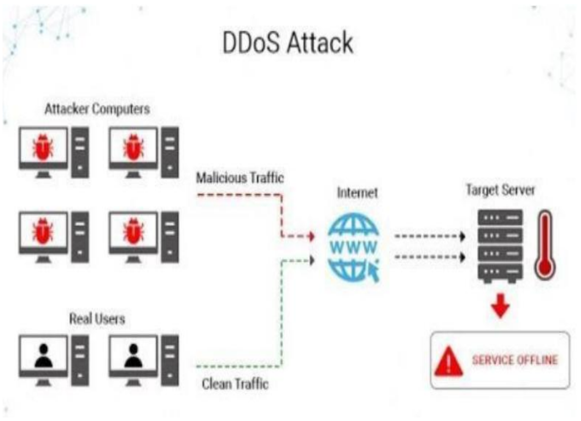


Fig-1. Architectures of the system model

**1. Application Layer:**

The application layer is where network applications are implemented. These applications encompass various functionalities such as network management, traffic planning, security enforcement, and any other network-related tasks. Examples of applications include network monitoring tools, intrusion detection systems, load balancers, and virtual private network (VPN) services. Network administrators and operators interact with applications at this layer to configure network policies, monitor network performance, and troubleshoot issues. The application layer abstracts the underlying network infrastructure, allowing administrators to manage the network using high-level commands and policies without dealing with the complexities of individual network devices.

2. **Control Layer:**

The control layer represents the centralized control plane of the SDN architecture. It hosts the SDN controller software, which acts as the central intelligence of the network. The SDN controller is responsible for making global decisions about network behavior based on input from network applications and policies defined at the application layer. It maintains a global view of the network topology and traffic flows, orchestrates network wide policies, and dynamically adjusts network configurations in response to changing conditions. The SDN controller communicates with network devices in the infrastructure layer to disseminate forwarding instructions and enforce network policies. By decoupling the control plane from the data plane, the control layer enables centralized management and programmability of the network, leading to enhanced flexibility, agility, and scalability. 3. **Infrastructure Layer:**

The infrastructure layer comprises the physical network devices such as switches, routers, and access points that forward network traffic.These devices form the data plane of the SDN architecture and are responsible for packet forwarding based on instructions received from the control layer. Unlike traditional networking architectures where control and data planes are tightly integrated within network devices, SDN separates these planes, allowing for greater flexibility and programmability. Network devices in the infrastructure layer communicate with the SDN controller through standard protocols such as Open Flow, providing visibility into network traffic and allowing for centralized control and management.

**Explanation:**

The SDN architecture provides a flexible and programmable framework for network management, allowing administrators to automate tasks, optimize performance, and enforce security policies more effectively. By centralizing control and decoupling it from the underlying hardware, SDN enables dynamic adaptation to changing network conditions and requirements. The application layer houses network applications that interact with end-users and administrators, while the control layer acts as the brain of the network, making global decisions and enforcing policies. The infrastructure layer comprises the physical network devices that form the backbone of the network, forwarding traffic based on instructions received from the SDN controller.

**MODULES**

**Traffic Management Module:**

To obtain traffic data, the Flow Collector first contacts the controller. When the controller receives a request for traffic data, it uses the Open Flow protocol to collect data from the flow tables 1. Each switch connected to the controller receives a flow statistics request from the controller, which asks the switch for flow statistics. As a result, each switch responds with a flow statistics response message containing all flow entries from all flow tables, each entry containing a description of the flow and associated counters. The controller responds to this component after collecting traffic information from all the switches. The stream collector parses the stream's data after receiving it and discards any unnecessary data. Stream ID and stream counters are important information to record. The source IP address, destination IP address, source MAC address, destination MAC address, TCP/UDP source port, TCP/UDP destination port and transport protocol identifier form a string of seven called the flow ID (protocol). ). It is important to remember that this floor can be modified to meet the network infrastructure requirements. Open Flowflow counters contain only three counters - packet count, byte count and duration.

**Detection module:**

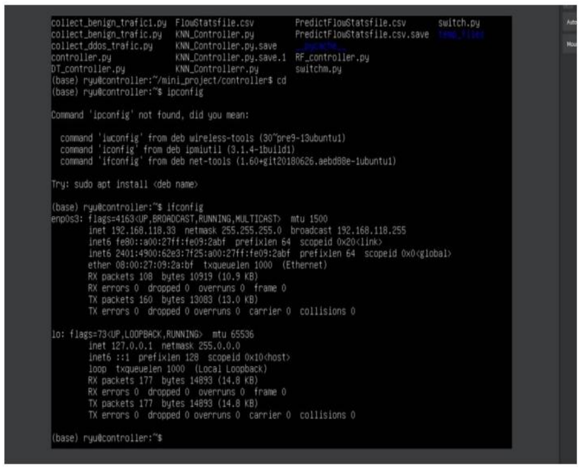
Depending on the source of data examined or the method used to detect abnormal events, several categories can be used to classify intrusion. Detection Depending on the data processing, flow-based or packet-based detection is used, and according to the detection method, signature-based or anomaly-based detection can be used. The flow-based detection was selected as data. Source to be evaluated because it would be more suitable for high-speed networks and more efficient in terms of processing and memory than packet-based detection.\ n Anomaly-based intrusion detection, and more specifically detection with ML, is applied as a detection method.

**Traffic module:**

After the anomaly detection module has inspected the malicious flow, the anomaly mitigation module is responsible for implementing mitigation measures to prevent network disruption or performance degradation. In the frame, the attack is stopped at this source. Preventing IP spoofing attacks does not

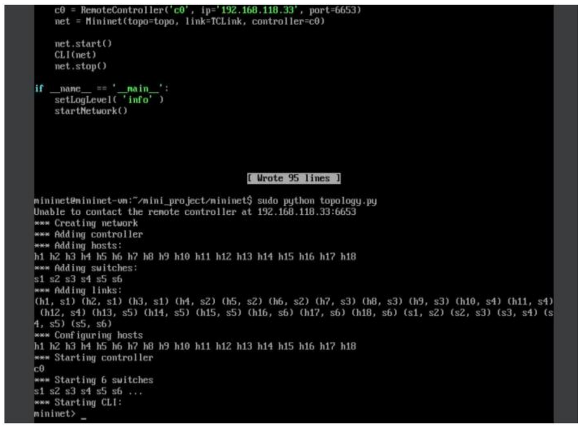
## V RESULT AND DISCUSSION

Ryu controller:



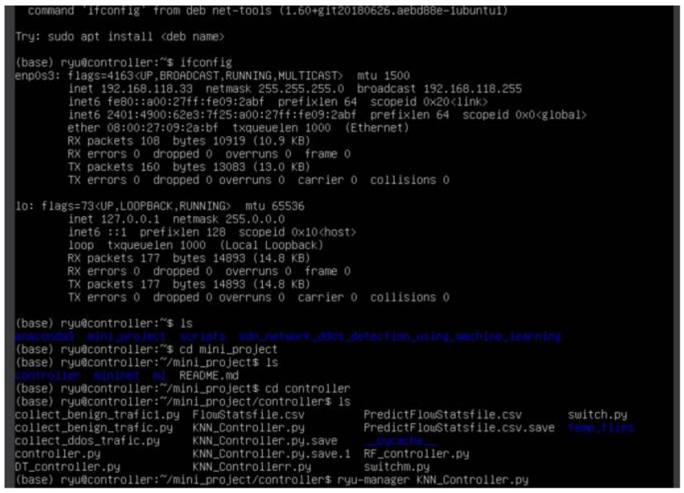
```
collect_benign_trafic1.py  FlowStatsfile.csv         PredictFlowStatsfile.csv       switch.py
collect_benign_trafic.py   KNN_Controller.py         PredictFlowStatsfile.csv.save  temp_files
collect_ddos_trafic.py     KNN_Controller.py.save    __pycache__
controller.py              KNN_Controller.py.save.1  RF_controller.py
DT_controller.py           KNN_Controllerr.py        switchm.py
(base) ryu@controller:~/mini_project/controller$ cd
(base) ryu@controller:~$ ipconfig

Command 'ipconfig' not found, did you mean:

  command 'iwconfig' from deb wireless-tools (30~pre9-13ubuntu1)
  command 'iconfig' from deb ipmiutil (3.1.4-1build1)
  command 'ifconfig' from deb net-tools (1.60+git20180626.aebd88e-1ubuntu1)

Try: sudo apt install <deb name>

(base) ryu@controller:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.118.33  netmask 255.255.255.0  broadcast 192.168.118.255
        inet6 fe80::a00:27ff:fe09:2abf  prefixlen 64  scopeid 0x20<link>
        inet6 2401:4900:62e3:7f25:a00:27ff:fe09:2abf  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:09:2a:bf  txqueuelen 1000  (Ethernet)
        RX packets 108  bytes 10919 (10.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160  bytes 13083 (13.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 177  bytes 14893 (14.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 177  bytes 14893 (14.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

(base) ryu@controller:~$
```

Mininet



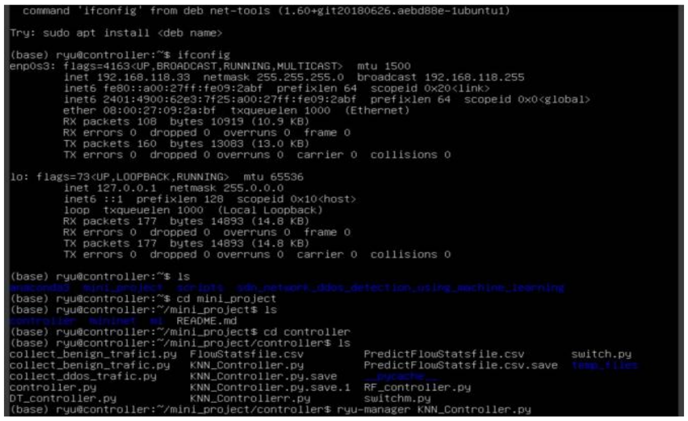
```
    c0 = RemoteController('c0', ip='192.168.118.33', port=6653)
    net = Mininet(topo=topo, link=TCLink, controller=c0)

    net.start()
    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    startNetwork()



                              [ Wrote 95 lines ]

mininet@mininet-vm:~/mini_project/mininet$ sudo python topology.py
Unable to contact the remote controller at 192.168.118.33:6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (h7, s3) (h8, s3) (h9, s3) (h10, s4) (h11, s4)
 (h12, s4) (h13, s5) (h14, s5) (h15, s5) (h16, s6) (h17, s6) (h18, s6) (s1, s2) (s2, s3) (s3, s4) (s
4, s5) (s5, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet> _
```

Activating ryu controller:



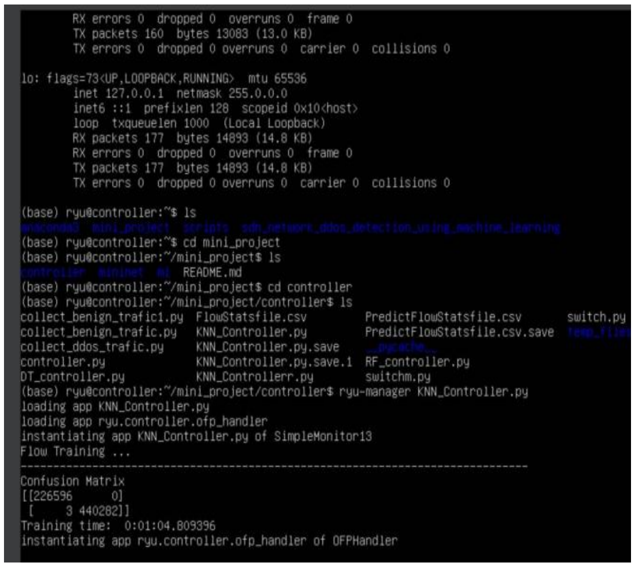
```
  command 'ifconfig' from deb net-tools (1.60+git20180626.aebd88e-1ubuntu1)

Try: sudo apt install <deb name>

(base) ryu@controller:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.118.33  netmask 255.255.255.0  broadcast 192.168.118.255
        inet6 fe80::a00:27ff:fe09:2abf  prefixlen 64  scopeid 0x20<link>
        inet6 2401:4900:62e3:7f25:a00:27ff:fe09:2abf  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:09:2a:bf  txqueuelen 1000  (Ethernet)
        RX packets 108  bytes 10919 (10.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160  bytes 13083 (13.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 177  bytes 14893 (14.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 177  bytes 14893 (14.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

(base) ryu@controller:~$ ls
anaconda3  mini_project  scripts  sdn_network_ddos_detection_using_machine_learning
(base) ryu@controller:~$ cd mini_project
(base) ryu@controller:~/mini_project$ ls
controller  mininet  ml  README.md
(base) ryu@controller:~/mini_project$ cd controller
(base) ryu@controller:~/mini_project/controller$ ls
collect_benign_trafic1.py  FlowStatsfile.csv         PredictFlowStatsfile.csv       switch.py
collect_benign_trafic.py   KNN_Controller.py         PredictFlowStatsfile.csv.save  temp_files
collect_ddos_trafic.py     KNN_Controller.py.save    __pycache__
controller.py              KNN_Controller.py.save.1  RF_controller.py
DT_controller.py           KNN_Controllerr.py        switchm.py
(base) ryu@controller:~/mini_project/controller$ ryu-manager KNN_Controller.py
```

Activating mininet



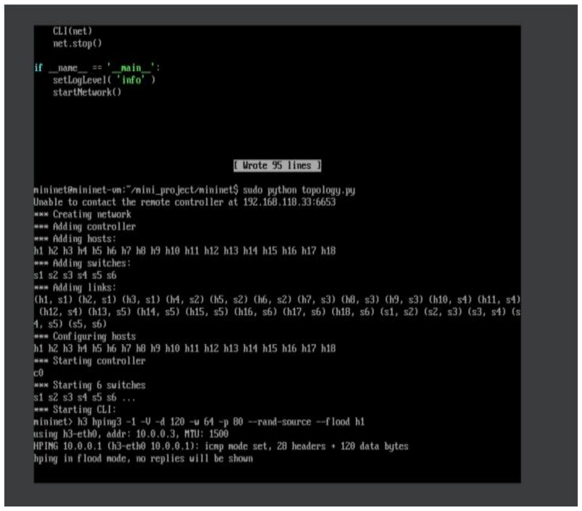
```
  command 'ifconfig' from deb net-tools (1.60+git20180626.aebd88e-1ubuntu1)

Try: sudo apt install <deb name>

(base) ryu@controller:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.118.33  netmask 255.255.255.0  broadcast 192.168.118.255
        inet6 fe80::a00:27ff:fe09:2abf  prefixlen 64  scopeid 0x20<link>
        inet6 2401:4900:62e3:7f25:a00:27ff:fe09:2abf  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:09:2a:bf  txqueuelen 1000  (Ethernet)
        RX packets 108  bytes 10919 (10.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160  bytes 13083 (13.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 177  bytes 14893 (14.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 177  bytes 14893 (14.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

(base) ryu@controller:~$ ls
anaconda3  mini_project  scripts  sdn_network_ddos_detection_using_machine_learning
(base) ryu@controller:~$ cd mini_project
(base) ryu@controller:~/mini_project$ ls
controller  mininet  ml  README.md
(base) ryu@controller:~/mini_project$ cd controller
(base) ryu@controller:~/mini_project/controller$ ls
collect_benign_trafic1.py  FlowStatsfile.csv         PredictFlowStatsfile.csv       switch.py
collect_benign_trafic.py   KNN_Controller.py         PredictFlowStatsfile.csv.save  temp_files
collect_ddos_trafic.py     KNN_Controller.py.save    __pycache__
controller.py              KNN_Controller.py.save.1  RF_controller.py
DT_controller.py           KNN_Controllerr.py        switchm.py
(base) ryu@controller:~/mini_project/controller$ ryu-manager KNN_Controller.py
```

Dataset training



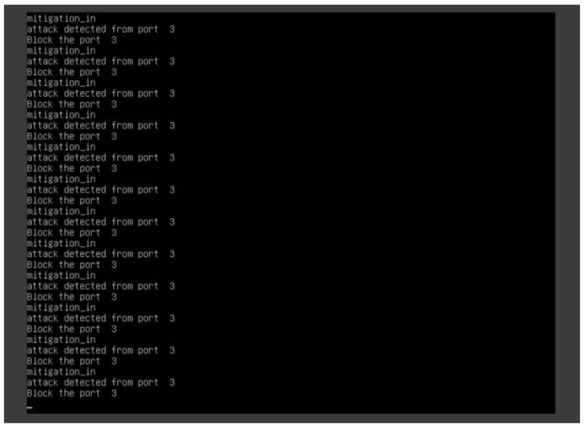
```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160  bytes 13083 (13.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 177  bytes 14893 (14.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 177  bytes 14893 (14.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

(base) ryu@controller:~$ ls
anaconda3  mini_project  scripts  sdn_network_ddos_detection_using_machine_learning
(base) ryu@controller:~$ cd mini_project
(base) ryu@controller:~/mini_project$ ls
controller  mininet  ml  README.md
(base) ryu@controller:~/mini_project$ cd controller
(base) ryu@controller:~/mini_project/controller$ ls
collect_benign_trafic1.py  FlowStatsfile.csv         PredictFlowStatsfile.csv      switch.py
collect_benign_trafic.py   KNN_Controller.py         PredictFlowStatsfile.csv.save temp_files
collect_ddos_trafic.py     KNN_Controller.py.save    __pycache__
controller.py              KNN_Controller.py.save.1  RF_controller.py
DT_controller.py           KNN_Controllerr.py        switchm.py
(base) ryu@controller:~/mini_project/controller$ ryu-manager KNN_Controller.py
loading app KNN_Controller.py
loading app ryu.controller.ofp_handler
instantiating app KNN_Controller.py of SimpleMonitor13
Flow Training ...
------------------------------------------------------------------------------
Confusion Matrix
[[226596      0]
 [     3 440282]]
Training time:  0:01:04.809396
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Creating floods



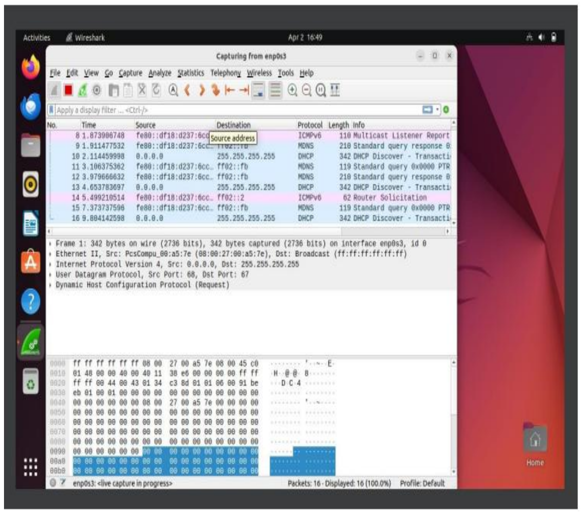
```
    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    startNetwork()



                                  [ Wrote 95 lines ]

mininet@mininet-vm:~/mini_project/mininet$ sudo python topology.py
Unable to contact the remote controller at 192.168.118.33:6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (h7, s3) (h8, s3) (h9, s3) (h10, s4) (h11, s4)
 (h12, s4) (h13, s5) (h14, s5) (h15, s5) (h16, s6) (h17, s6) (h18, s6) (s1, s2) (s2, s3) (s3, s4) (s
4, s5) (s5, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet> h3 hping3 -1 -V -d 120 -w 64 -p 80 --rand-source --flood h1
using h3-eth0, addr: 10.0.0.3, MTU: 1500
HPING 10.0.0.1 (h3-eth0 10.0.0.1): icmp mode set, 28 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

Blocking floods



```
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
mitigation_in
attack detected from port  3
Block the port  3
_
```

Blocked floods



Activities
Wireshark
Apr 2 16:49
Capturing from enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Source address
Home

```
No.   Time          Source                 Destination       Protocol  Length Info
    8 1.873906748   fe80::df18:d237:6cc…   ff02::16          ICMPv6    110 Multicast Listener Report
    9 1.911477532   fe80::df18:d237:6cc…   ff02::fb          MDNS      210 Standard query response 0
   10 2.114459998   0.0.0.0                255.255.255.255   DHCP      342 DHCP Discover - Transacti
   11 3.106375362   fe80::df18:d237:6cc…   ff02::fb          MDNS      119 Standard query 0x0000 PTR
   12 3.979666632   fe80::df18:d237:6cc…   ff02::fb          MDNS      210 Standard query response 0
   13 4.653783697   0.0.0.0                255.255.255.255   DHCP      342 DHCP Discover - Transacti
   14 5.499210514   fe80::df18:d237:6cc…   ff02::2           ICMPv6     62 Router Solicitation
   15 7.373737596   fe80::df18:d237:6cc…   ff02::fb          MDNS      119 Standard query 0x0000 PTR
   16 9.804142598   0.0.0.0                255.255.255.255   DHCP      342 DHCP Discover - Transacti

› Frame 1: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface enp0s3, id 0
› Ethernet II, Src: PcsCompu_00:a5:7e (08:00:27:00:a5:7e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
› Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
› User Datagram Protocol, Src Port: 68, Dst Port: 67
› Dynamic Host Configuration Protocol (Request)

0000  ff ff ff ff ff ff 08 00  27 00 a5 7e 08 00 45 c0   ········ '··~··E·
0010  01 48 00 00 40 00 40 11  38 e6 00 00 00 00 ff ff   ·H··@·@· 8·······
0020  ff ff 00 44 00 43 01 34  c3 8d 01 01 06 00 91 be   ···D·C·4 ········
0030  eb 01 00 01 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 08 00  27 00 a5 7e 00 00 00 00   ········ '··~····
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········

enp0s3: <live capture in progress>          Packets: 16 · Displayed: 16 (100.0%)    Profile: Default
```

## VI CONCLUSION

The conclusion of this study underscores the efficacy of employing a Random Forest (RF) machine learning algorithm for the detection and mitigation of Distributed Denial of Service (DDoS) attacks within Software-Defined Networking (SDN) environments. Through rigorous experimentation and analysis, it became evident that RF outperformed other evaluated algorithms such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) in accurately classifying network traffic and discerning between normal and anomalous flows. One of the key strengths of the implemented methodology lies in its ability to swiftly identify and mitigate DDoS attacks without disrupting regular network operations. By continuously collecting and analyzing traffic flow entries, the system demonstrated proactive threat detection capabilities, effectively preventing the source of detected attacks and minimizing potential service disruptions. This rapid response mechanism enhances network resilience and ensures uninterrupted service delivery. The validation of the methodology under diverse network conditions and attack scenarios further substantiates its effectiveness in real-world deployment scenarios. The comprehensive evaluation process underscored the robustness and reliability of the RF algorithm in combatting evolving DDoS threats, instilling confidence in its potential for widespread adoption within SDN security frameworks. Looking ahead, future research endeavors could focus on refining the feature extraction process, optimizing machine learning algorithms, and incorporating advanced threat intelligence mechanisms to enhance the system's capabilities. Additionally, exploring scalability and extensibility to accommodate larger network infrastructures would be beneficial. Collaboration with industry stakeholders and cyber security experts could provide valuable insights, fostering the development of comprehensive SDN security solutions to address emerging threats effectively.

## FUTURE ENHANCEMENT

Detecting and mitigating distributed denial of service (DDoS) attacks is an ongoing challenge in cyber security. Using machine learning algorithms such as Random Forest can improve detection and mitigation capabilities. To improve the random forest to detect and mitigate DDoS attacks, do the following.

**1.Feature Design:** Refine feature selection by considering various network traffic attributes such as packet size, packet rate, protocol distribution, source and destination IP addresses, etc. Include functions that can effectively distinguish between normal and attack traffic.

**2. Collection Methods:** Implement collection methods such as bagging or boosting with Random Forest to improve its performance. Ensemble methods can combine multiple weak models to create a stronger predictive model that increases the accuracy of DDoS attack detection.

**3. Dynamic Update:** Enable a dynamic update mechanism that continuously updates the model based on changes in DDoS attacks and network traffic patterns. This ensures that the model remains effective in detecting new types of attacks.

**4. Anomaly detection:** A Random Forest can be trained to detect anomalies in network traffic that may indicate potential

DDoS attacks. Include anomaly detection techniques to detect abnormal network behavior.

**5. Real-Time Analytics:** Develop a real-time analytics framework that can process network traffic data in real-time and make timely decisions about DDoS attack detection and mitigation. This requires algorithm optimization for low latency and high throughput.

**6. Integration with network infrastructure:** Integrate the Random Forest model with network infrastructure components such as firewalls, routers and switches to automate the mitigation process. This allows the network to automatically respond to detected DDoS attacks by blocking or redirecting malicious traffic.

## VII REFERENCES

[1]v. jyothi, s . k. addepalli, and R. karri, " deep packet field extraction engine DPFEE A pre-processor for network intrusion detection and Denial-of-service detection systems," IEEE ICCD,PP.287-293,2015.

[2]Imperva, "Q2 2015 Global DDoS threat landscape: assaults resemble advanced persistent threats", https://www.incapsula.com/blog/ddos-globalthreat-landscape-report-q2-2015.html.[3]X. Wangand R. karri ," unchecked/; detecting kernel control-flow modifying rootlets by using hardware performance counters,"IEEE DAC,PP.1-7,2013.

[4]K.Kato and v.klyuev, "An intelligent DDoSattacks detection system using packet analysis and support vector machine,"IJICR,PP.478-485,2014.

[5]K. Devi, G. Preetha, G. Selvaram, and S.M. Shalinie, "An impact analysis: Real time DDoS attacks detection and mitigation using machine learning,"ICRTIT,PP.-7,2014.

[6]a. Ramamoorthi, T. subbulakshmi, and S. M Shaline, " Realtime detection and classification of ddos attacks using enhanced svm with string kernels,"ICRTIT,PP 91-96,2011.

[7] Prasadu Peddi (2018), Data sharing Privacy in Mobile cloud using AES, ISSN 2319-1953, volume 7, issue 4.

[8]A.Li.X. yang,S. Kandula,and M.Zhang, "cloudscape: comaring public cloud providers," in proc.ACM SIGCOMN Internet meas. Conf, 200,pp.1-4.

[9]A Bessani, M. cotteia, B. Quaresma, F.Andr_e, and p. souse, “depsky. Dependable and secure stoage in a cloud-of-clouds,” in proc 6thconf. comut system, 2013, pp. 31-46.

[10] Prasadu Peddi (2015) "A review of the academic achievement of students utilisinglarge-scale data analysis", ISSN: 2057-5688, Vol 7, Issue 1, pp: 28-35.

## AUTHORS

**Mrs.B.Priyanka,AssistantProfessor** Dept. of CSE, Teegala Krishna Reddy Engineering College Meerpet, Hyderabad.

Email: priyankabhupathi90@gmail.com

**Mr. P.Adhithi**, Dept. of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad.

Email: adithipippera@gmail.com

**Mr. S.Murali Krishna**, Dept. of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad.

Email: sakethamuraliKrishna9@gmail.com

**Mr. V.Sanjay,** Dept. of CSE, Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad.

Email: velugurisanjay@gmail.com