Simulation of Low-Rate Targeted DoS Attacks and ML based Prevention Techniques

Course Project: Communication Networks (EE 706)

Mohit Kumar Meena
Department of EE
IIT Bombay
Mumbai, India
213070021@iitb.ac.in

Harsh Diwakar
Department of EE
IIT Bombay
Mumbai, India
213070018@iitb.ac.in

Sunaina Saxena
Department of EE
IIT Bombay
Mumbai, India
213070001@iitb.ac.in

***Abstract*—Denial of Service attacks can prevent or degrade the service of users by consuming resources in networks, server clusters, or end hosts. Many different varieties of DoS attacks exist, such as TCP SYN attacks or DNS flood attacks, but one commonality is that these attacks generally require high-rate transmission of packets. Here, we simulate a 'Low-Rate TCP targetted DoS attack' which is harder to detect as compared to the traditional DoS attacks. These DoS attacks make use of the TCP congestion control's retransmission timeout (RTO) functionality to stop communication between a sender and a receiver. In later section, we explore different prevention techniques and found based on previous researches that the existing techniques does not perform well or have some negative side effects. For these stated reasons we also provide ML based prevention techniques that gives upto 98.23%. Such ML based models not only prevent from LDOS attack but can also be used to prevent from different types of attack on network.**

I. Introduction

In networking, a denial-of-service attack (DoS attack) is a cyber-attack in which the intruder seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to a network. Denial of service is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled [1].

TCP congestion control relies on the implicit assumption of end-system cooperation. In this particular TCP mechanism the a target is the Retransmission Time Out (RTO) mechanism: in times of severe congestion, flows reduce their windows to a single packet and wait for a single RTO, after which that packet is resent; further loss causes the RTO to double. With "square wave" shrew attacks that alternate between outages (bursts of packets that exceed the bottleneck link capacity) and sending no packets, we exploit repeated timeouts to disrupt throughput. For a single TCP flow, the attacker can force the flow's throughput to 0 if they send an outage timed with the flow's RTO.

A. TCP Congestion Control

The motive is to limit send rate when network is congested and to perceive the congestion the implicit end to end feedback i.e ACK (Acknowledgement) is used. The sending rate is formulated as

$$rate = \frac{cwnd(CongestionWindow)}{roundTripTime}$$

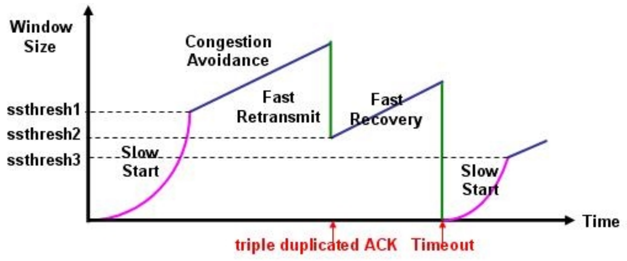

Figure 1: TCP Congestion control and it's modes [2]

Now the oracle solution is to find the rate just below the congestion level, which is hard to find so we used TCP's timeout mechanism for bandwidth probing as discussed in section IV-A. TCP Congestion control can be summarized using the state diagram as shown in figure 2. It contains three

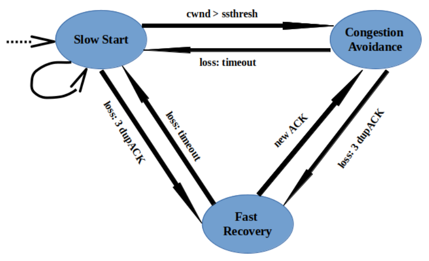

Figure 2: State diagram of TCP congestion control

main blocks:

- **Slow start:** Slow start begins with a initial value of cwnd, the transmission rate will be increased by the slow-start algorithm until either a packet loss is detected or slow start threshold (ssthresh) is reached. When $cwnd > ssthresh$ it goes into congestion avoidance mode.
- **Congestion Avoidance:** When there's a timeout then it goes into slow start mode. Congestion avoidance mode is very conservative so when it is detected increase the cwnd by 1 MSS (Maximum segment size) per RTT. Therefore, for each ACK received

$$cwnd+ = MSS/cwnd$$

- **Fast Recovery:** This mode lies in between agressive slow start mode and conservative congestion avoidance mode, i.e if loss of three duplicate ACK (dupACK) is received then the network from either slow start or congestion avoidance goes into fast recovery mode and if loss of new ACK is received then it goes to congestion avoidance mode as shown in figure 2.

The graphical illustration of TCP congestion window variation and AIMD (Additive increase and multiplicative decrease) can be clearly seen in figure 10.
In section II, prior work related to low rate TCP congestion control and novelty of our project is discussed. Contribution made by each team member is specified in the Section III. Section IV focuses on methodologies that we are going to implement in our project for simulating the LDOS attack, Section V focuses on the different prevention techniques from LDOS attack, Section VI explains the different models that we implemented for prevention from LDOS techniques, Section VII carries the implementation details of low-rate targetted DoS attacks and ML based prevention methods. In Section III, the contribution of each group member is listed. Finally, the conclusion is drafted in Section VIII.

II. Prior work and our novelty

A. Prior work

In paper "Low-Rate TCP-Targeted Denial of Service Attacks" Kuzmanovic and Knightly present a class of low-rate, or shrew, attacks that can severely disrupt TCP flows while evading detection. Using modeling and simulation in ns2, the paper demonstrates the effectiveness of these maliciously chosen traffic patterns.
In paper "Low-Rate DoS Attacks, Detection, Defense, and Challenges: A Survey" Zhijun Wu et al [3] theoretically carried out a survey of detection and prevention methods of LDoS attacks. They firstly discussed about LDoS attacks and provide a classification of LDoS attacks, then they categorize the characteristics of each kind of LDoS attacks. Secondly, they explored the security vulnerabilities exploited by LDoS attacks, the organization of LDoS attacks, different scenarios LDoS attacks occur in, and the assessment of LDoS attacks effect. And lastly they present a discussion of detection and defense of LDoS attacks where they talk about about mainly two types of defence mechanism- 1) Router- Assisted Mechanisms which they have experimentally proved can only mitigate, but not eliminate the effectiveness of attack and 2) End-point minRTO Randomization where they have explained about the very complex randomization procedure for defending the system by sacrificing the system performance in the absence of such attack.

B. Our Novelty

Our novelty in this project is that we first simulated the results described in the paper by Kuzmanovic and Knightly on ns3 and visualized the animation using NetAnim pro interface. We are also planning to explore, design and implement a defence mechanism to prevent such types of DoS attacks based on the end- point randomization methods because the router assisted methods are proved not to be affective [4].

There were few prevention techniques mentioned in the paper by Kuzmanovic and Knightly but they were not much effective or had some side effects. For this reason and also that we wanted to integrate ML and network security in our project we implemented ML based models for prevention from low rate DOS attack.

III. Contribution

A. Project Idea and Literature Review
Sunaina: TCP Timeout Mechanism
Harsh: Low Rate DoS attack
Mohit: TCP Congestion Control

B. Implementation in ns3
Sunaina & Harsh: Low Rate DoS attack
Mohit: TCP Congestion Control

C. Prevention based on ML models
Sunaina: RNN based model
Harsh: GRU based model
Mohit: LSTM based model

IV. Low rate denial of service attack

In this section we will explain how a low rate denial of service attack is possible exploiting the TCP's time out mechanism. But before we explain the TCP timeout mechanism.

A. TCP's Timeout Mechanism

TCP timeout mechanism is used for loss recovery in the presence of severe congestion. When losses occur, TCP Reno detect loss either by timeout from non-receipt of ACKs or by receipt of a three duplicate ACKs. If the sender has not received an ACK packet for a certain period of time since the packet was sent, it times out. TCP interprets the timeout as indication of severe congestion. Selection of timeout value is crucial because if it is too low then spurious retransmissions will occur as packets are incorrectly assumed to be lost if ACKs are just delayed. Similarly, if set too high, packets will have to wait unnecessarily long to recover from congestion. To address this issue, it is experimentally found that that

TCP achieves near-maximal throughput if there exists a lower bound for RTO of 1 second [4]. To address this factor, a TCP sender maintains two state variables, SRTT (smoothed round-trip time) and RTTVAR (round trip time variation).
For the first RTT measurement ($R^{'}$),
$SRTT = R^{'}$ and $RTTVAR = \frac{R^{'}}{2}$
$RTO = SRTT + \max(G, 4 \times RTTVAR)$

where, G denotes the clock granularity (typically 100 ms) [4]. When a subsequent RTT measurement $R^{'}$ is made, a host sets the following:

$RTTVAR = (1 - \beta)RTTVAR + \beta|SRTT - R^{'}|$
$SRTT = (1 - \alpha)SRTT + \alpha R^{'}$

where, α and β are implementation dependent.
Combining these, the value of RTO becomes:

$$RTO = \max\{\min RTO, SRTT + \max(4 \times RTTVAR, G)\}$$

Figure 3 illustrates the behavior of the TCP retransmission timer. At reference time t = 0, sender sends a packet and retransmission timer of 1 second is initiated. If the packet is lost, the flow “times out” when the timer expires at t = 1 sec. The sender enters the exponential backoff phase. It reduces the congestion window to one, doubles the RTO value to 2 seconds.Now the un-ACKed packet is send again and retransmission timer is reset with this new RTO value. If the packet is lost again, this process continues. Alternatively, if packet is successfully retransmitted at time t = 1 sec as shown in Figure 3, its ACK will arrive to the sender at time t=1+RTT and sender exits the exponential backoff phase and enters slow start, doubles its window size, transmitting 2 new packets and reseting the retransmission timer with the current RTO value of 2 sec. If the two packets are not lost, they are acknowledged at time t= 1+2*RTT, and SRTT, RTTVAR and RTO are recomputed [4].

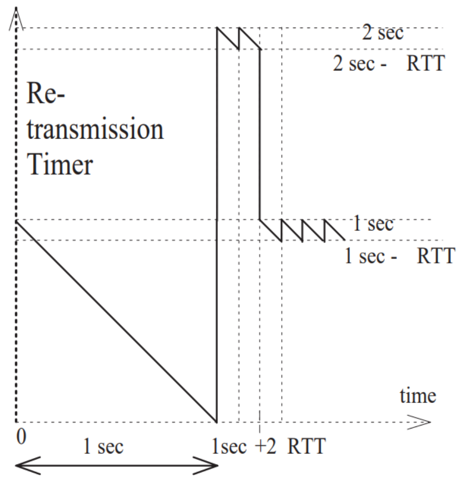

Figure 3: Illustration of TCP retransmission timer [4]

B. Low-Rate TCP-Targeted Denial of Service Attacks

Low-rate TCP-Targeted DoS attack[4] (also named as Shrew attack) is a low-rate DoS attack that attempts to deny bandwidth to TCP flows while sending sufficiently low rate to not get detected by the counter-DoS methods. The attacker essentially sends short bursts of packet to a bottleneck link in the network periodically. The sending rate during the burst is high enough to fill the bottleneck quickly. When the queue is full, TCP packets in the bottleneck are dropped due to buffer overflow. This attack exploits the TCP’s timeout mechanism that is if a large number of packets in a TCP window are dropped, the connection is forced to timeout. The attacker is quit between the burst of transmission as the TCP will not try to make a connection in timeout period. This is explained in the figure 3, Alice is trying to a connection to Bob through a bottleneck link A-B. This bottelneck link is common to Trudy (Attacker). Attacker sends the bursts of packets that coincides with Alice’s TCP RTO. Three main important parameters of

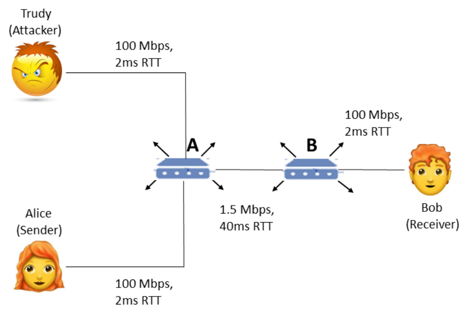

Figure 4: Illustration of Low Rate TCP-targeted DoS attack

the attack are: a) attack rate, b) burst length, and c) inter-burst period. attack rate is directly related with the number of packets sent in the burst length period, burst length is the time for which the packets are sent into the bottleneck link, and inter-burst period is the time when the attacker is quit because victim’s TCP is not trying to make any connection (figure 5) because of time out mechanism. However, when the TCP will try to make a connection after waiting for RTO time, then the attacker will send the bursts again.

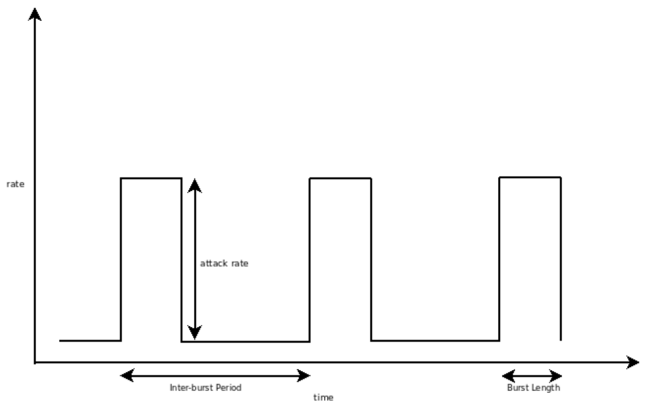

Figure 5: Important parameters of the DoS attack

Low rate is the special characteristic of this attack. The attacker needs to attack only at the times when TCP is sending packets. Even if the attacker does not have information of the bottleneck link, this attack will continue to work as long as the victim and attacker share the same bottleneck link.

V. Counter DOS Techniques

In this section we first explain the different methods available for prevention from low rate DOS attacks. The two main techniques that were suggested in paper shrew attack paper [4] are Router assisted techniques and the End-point minRTO Randomization techniques. In the paper authors have found that the router assisted methods are not very effective. And there were studies showing "Although this method can reduce the attack to a certain extent, it will bring many negative effects, such as when there is no LDoS attack, the RTO randomization will reduce the performance of TCP"[5]. Hence we decided to explore other techniques. As now a days ML is being used extensively in communication networks and network security, we also tried to use the ML based techniques for prevention of the LDOS attack. One more advantage of using ML based techniques is that it will not only learn to predict the LDOS attack but also it can be used to detect other type of attacks and hence defending against it.

A. Router-Assisted Mechanisms

Although DoS flows have a low average rate, they do deliver high-rate bursts for short periods of time. For investigating such traffic patterns router-based algorithms such as Flow Random Early Detection (FRED), CHOKe, Stochastic Fair Blue (SFB), the scheme of reference, ERUF, Stabilized RED (SRED), Random Early Discard algorithm improved with Preferential Dropping (RED-PD), etc are devised. We focus on RED-PD since it relies on packet drop history at the router to reliably detect high-bandwidth flows. RED-PD detects and monitors flows that exceed the configured target bandwidth. Packets from monitored flows are dropped with a probability based on the flow's excessive sending rate and the current throughput of the output queue. When there isn't enough demand from other traffic in the output queue, RED-PD suspends preferred dropping.
To give RED-PD an advantage over previous approaches in identifying high throughput flows, a small fraction of the flows must be responsible for the majority of the bytes sent (over small and long intervals) and the high bandwidth flows in one interval must be a good predictor of the high bandwidth flows in the next interval. The RED-PD technique has a reduced ambient drop rate because it selectively drops packets from a small number of high-bandwidth flows.

B. End-point minRTO Randomization

Randomizing RTO is one technique recommended for low-rate TCP-targeted DoS attacks. When RTO is not a fixed value, attackers are unable to foresee the next TCP timeout interval and prepare a timed periodic attack burst. The attacks' unwanted effects can be avoided by carefully selecting RTO ranges. Instead of using a deterministic value of 2^k for the k^{th} successive timeout we choose a random value uniformly between 2^k and $2^k + 1$. Various TCP connections create different RTOs following an attack burst, so the attacker is unable to synchronise the next set of timeouts. This RTO range selection follows the traditional technique, resulting in an RTO that is always greater than or equal to the value selected in legacy TCP. Statistically the average of the randomized RTO is 0.5 sec larger. There are other options that spread the range to both sides of 2^k. Specifically, we can consider the ranges: (a) $2^k - 0.5$ to $2^k + 0.5$, (b) $2^k * 0.5$ to $2^k * 1.5$.
It is a fact that RTO occasionally goes below 1 sec does not compromise the stability. Moreover, in most cases [2^k,$2^k + 1$] will suffice in defending against the attacks.

C. ML Based Approach

We know that low-rate attack are hard to detect because they looks similar to the legitimate network traffic from the victim point of view as discussed in section II(b). Machine learning based approach can be used as a preventive measure since this method improve performance based on statistical features. Conventional ML based techniques like K-Nearest Neighbor, Decision Tree, Support Vector Machine, etc. are limited by the shallow representation models. In this project, we have used LSTM (Long short term memory) and GRU (Gated recurrent unit) which are the models of RNN (Recurrent neural networks), these models can automatically extract high-level features from low-level ones and gain powerful representation and inferences. Moreover, these models are able to learn patterns from sequences of network traffic and trace network attack activities which is desired in our application. General architecture in detection of low rate DDoS attack is shown in figure (6). In the first step as shown in figure (6), the network

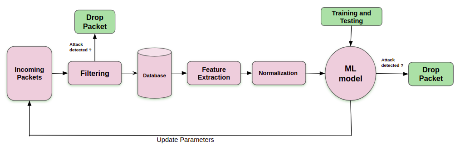

Figure 6: A of low rate DDoS attack detection

traffic is filtered by filtering algorithm and if attack is detected then drop that packet rest are stored in the database. Next step is feature extraction step in which the features will be extracted from the traffic like packet rate, protocol type, frame length, frame protocol, etc. After the features are extracted then the features are normalized to speed up the training process. Next, training data is trained using machine learning algorithms (LSTM/GRU in our case). Following this architecture, each packet will be classified as a DDoS attack or a legitimate one. In the last step, the system drops the DDoS packets detected by machine learning algorithm and parameters are updated.

VI. ML MODELS

A. Recurrent Neural Networks

Recurrent Neural Network are a special kind of neural network that are designed to work with sequential data. In our case we have sequential data in the form of network traffic. Refer to Figure 7, we see the architecture of a RNN unit. x_t and y_t are the sequential inputs at t time-step, H and O are the non-linearity applied (for e.g. ReLU, Sigmoid etc.) Two main equations governing the RNNs are:

$$\begin{aligned} h_t &= H(Wx_t + Vh_{t-1} + b^{(h)}) \\ y_t &= O(Ux_t + b^{(y)}) \end{aligned} \tag{1}$$

where, W is the learnable weights associated with input and hidden layers, V and U are also the learnable weights associated with hidden- hidden layers and hidden- output layers respectively and $b^{(h)}$ and $b^{(y)}$ are the learnable biases associated with hidden layer and output respectively. One main advantage of using RNNs is that the weights are shared between across the time. One major problem with RNNs is the vanishing gradient problem that is the gradients tends towards 0 when the time steps are large. Some gated versions of RNNs are proposed such as LSTMs [6] and GRUs [7] that we talk about next.

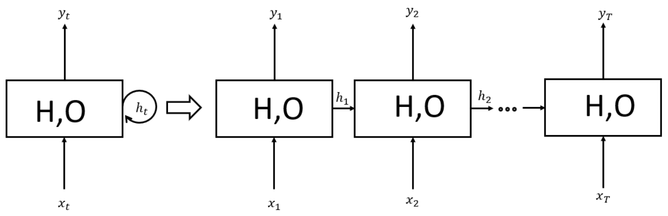

Figure 7: Demonstration of a RNN unit

B. Long Short Term Memory (LSTM)

An LSTM [6] has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells. The core concept of LSTM's are the cell state, and it's various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain and among various gates, most important is output gate which decides what the next hidden state should be. Remember that the hidden state contains information on previous inputs. The hidden state is also used for predictions. First, we pass the previous hidden state and the current input into a sigmoid function. Then we pass the newly modified cell state to the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry as shown in figure (8)

C. Gated Recurrent Unit

Again like LSTMs, GRUs also have the same workflow as RNNs and the operations inside a GRU unit is slight different. Just like the LSTMs, GRUs also gates and they are known as reset gates and update gate. Update gate decides if the cell state should be updated with the candidate state (current activation value) and reset gate decides whether the previous cell state is important or not. One unit of GRU is shown in Figure [7]

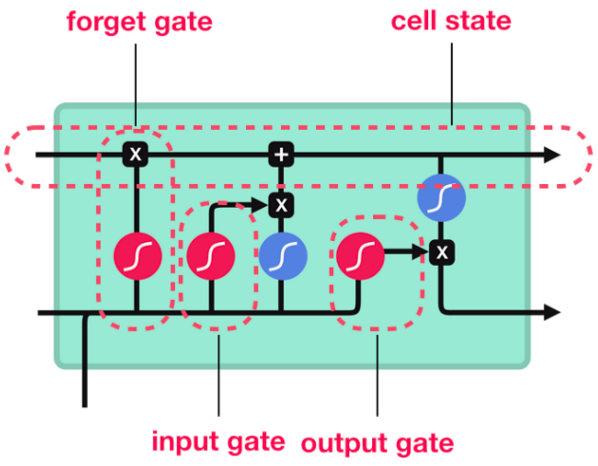

Figure 8: General architecture of a LSTM unit [8]

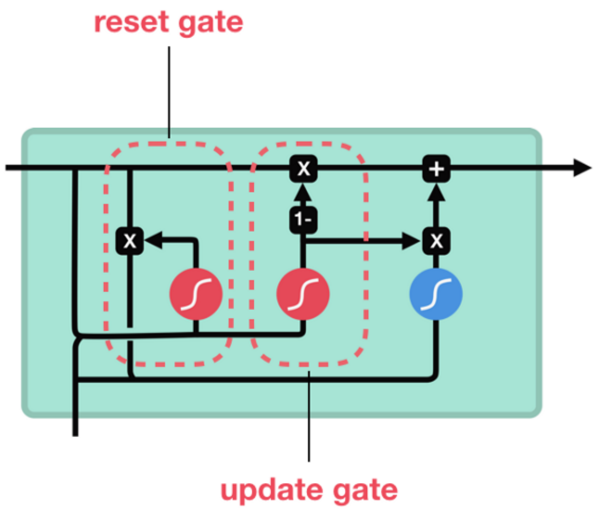

Figure 9: General architecture of a GRU unit [8]

VII. IMPLEMENTATION

In this section we show our simulation of Low rate DOS attack on NS-3 first then we move on to prevention of such attacks based on ML based techniques. In ML based techniques we implemented RNN, LSTM and GRU based classification techniques that will also be explained in this section.

A. TCP and Low rate DOS atack

The attack discussed in section I-A and IV-B has been simulated and animation of the same is visualized using NetAnim package of ns3. Figure 10 shows the implementation of TCP congestion window size variation which matches the theoretical diagram which is in shown previously in figure 1 of section I-A. In figure 11 and figure 12, the implementation

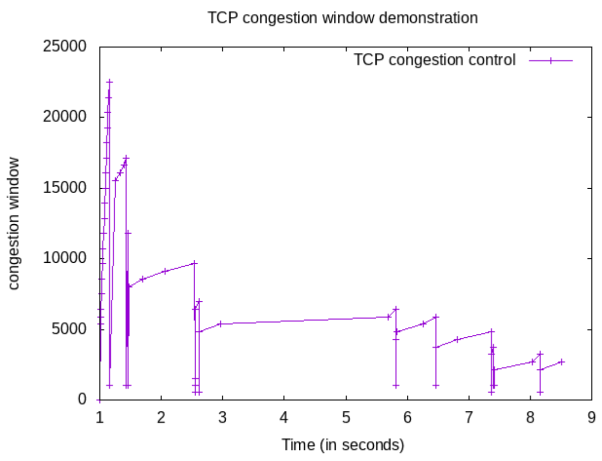

Figure 10: Congestion window demonstration

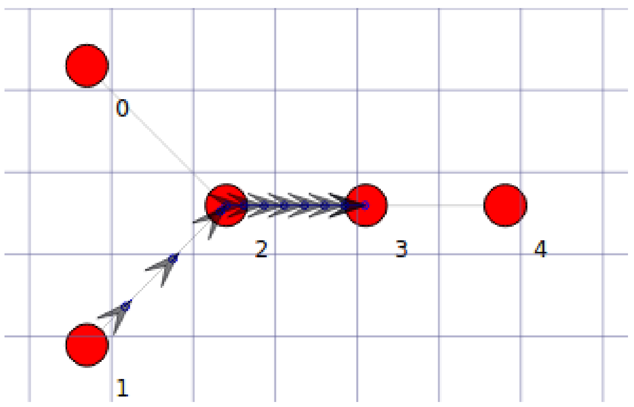

Figure 11: Implementation of TCP low-rate DoS attacks (intruder transferring data)

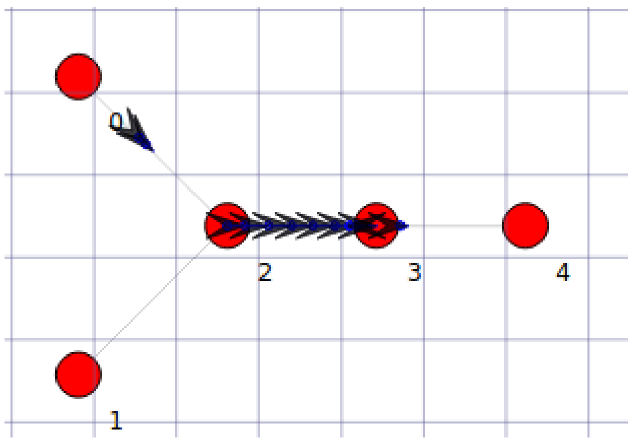

Figure 12: Implementation of TCP low-rate DoS attacks (legit user transferring data)

of low-rate DoS attack is shown with the which are produced using the NetAnim package of ns3. The difference between them is in figure 12, the legitimate user is sending the data whereas in figure 11, the intruder is shown interfering for a burst length of 0.25 ms, burst period of 1 ms and at an attack rate of 12000kb/s. This implementation is carried out for point to point protocol between all the nodes with the different data rate and RTT values between the nodes e.g 100 Mbps (data rate) and 2ms RTT between node 0 and node 1. Note that the intruder at node 1 uses UDP on-off application to create low-rate bursts whereas legitimate user uses TCP Bulk send application.

B. Prevention based on Machine Learning techniques

ML based classification models use data to learn the representations and predict the labels. In our experiments also we are doing the same we first explain about the dataset used in this experiment and then the model and results.

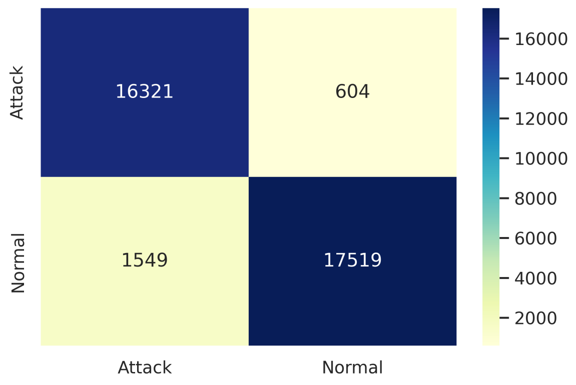

Figure 13: Confusion matrix for vanilla RNN model

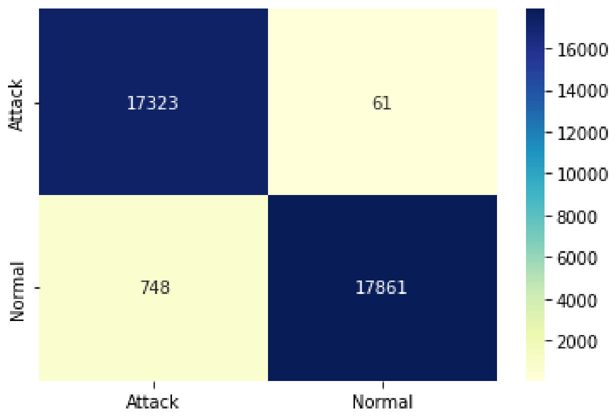

Figure 14: Confusion matrix for LSTM model

1) Dataset and Data preprocessing: We trained and evaluated our model on ISCX2012 intrusion detection evaluation dataset [9].This dataset contains recorded 7 days network traffic (legitimate and malicious), as mentioned in the data

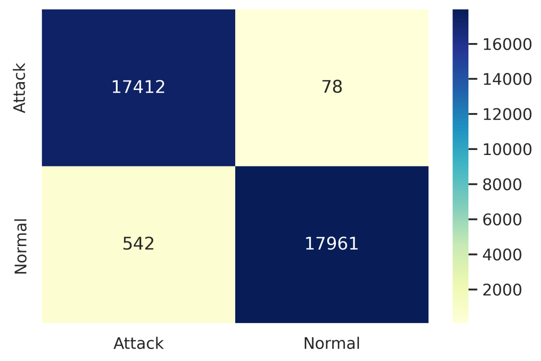

Figure 15: Confusion matrix for GRU model

description that attack occurs in only two days so we extracted the data of these two days. This dataset also contains the list of labels and it shows legitimate or attack network with it's various information like source IP, destination IP, time period, source port, destination port, etc. Since each packet is only listed in the data description, we first match the attack list with the network traffic and label each attack and legitimate packet. We observed that most network traffic are legitimate ones so to eliminate data skewness, we collect all attack packets and randomly choose legitimate ones to match the number of attack packets.

In the preprocessing section, first 60k packets are considered to train and test the model since they are giving good accuracy, the loaded packets of both malicious and legitimate user is concatenated to form a single array of 120k packets followed by shuffling them. Out of 120k packets, 84k are used for training and 36k are used for testing. Data is then standarized and fed to the model.

2) Model architectures: First the time series data were fed into the RNN and different variants of RNN that were explained in Section VI, then two fully connected neural network layer with 128 and 1 number of neurons were were used to classify if an attacker is present or not. Non linearity applied for the three are tanh, relu and sigmoid respectively. The loss used for training this model was binary cross entropy loss and the Adam optimizer [10] was used as an optimizer. Tensorflow was used for implementing the different ML models.

The accuracy and F-1 scores recorded on the 36k test samples by each model is shown in Table I. It is observed that the accuracy and F1-Score predicted by GRU model were the best and RNN gave the poorest result as expected maybe because of the vanishing gradient problem. Confusion matrix of different methods is also plotted in Figure 13, 14 and 15 for RNN, LSTM and GRU models respectively.

Table I: Accuracy and F-1 score of different models.

Model	Accuracy	F-1 Score
RNN	94.02%	93.81
LSTM	97.7%	97.92
GRU	98.28%	98.25

VIII. Conclusion

In this project we explored how the several mechanism of TCP work in real life. We also simulated the idea of Kuzmanovic and Knightly for Low rate DoS attack using ns3 simulations and visualized the same with animations using NetAnim pro interface. It is also discussed in this paper how the timeout mechanism is really exploited for this Low rate DoS attack to be possible. We feel there are also few limitations of this attack. Some of which are: 1) The attacker and host connecting to the network should be sharing the same bottleneck link, 2) Attacker should know in advance when the host will try to make a connection (precise timing), 3) The rate and other parameters of DoS attack depend on the parameters of bottleneck link. We observed from the results and animations in the Implementation section that even after these limitations, this attack is severely damaging for the network.

For the prevention mechanisms as it was experimentally proved by Kuzmanovic and Knightly that Router assisted techniques for prevention does not perform very well on such attacks and also studies has shown that end point randomization based prevention techniques may reduce the performance of TCP, we implemented a technique based on Machine Learning and Recurrent Neural network that do not suffer from the above problems. Also, ML based techniques may also provide prevention with other types of DOS attacks based on the data used for training. We observed experimentally that the best performing model for this task was GRU based classifiers.

References

[1] Wikipedia, "Dos attacks," https://en.wikipedia.org/wiki/Denial-of-service_attack, accessed 13-03-2022.
[2] project code bank, "Tcp congestion control," https://sites.google.com/site/projectcodebank/computer-engineering-notes/tcp-congestion-control-algorithm, accessed 14-03-2022.
[3] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-rate dos attacks, detection, defense, and challenges: A survey," *IEEE access*, vol. 8, pp. 43 920–43 943, 2020.
[4] A. Kuzmanovic and E. W. Knightly, "Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 75–86.
[5] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-rate dos attacks, detection, defense, and challenges: A survey," *IEEE Access*, vol. 8, pp. 43 920–43 943, 2020.
[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
[8] towards data science, "Lstm and gru," https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, accessed 02-05-2022.
[9] UNB, "Iscx intrusion detection evaluation dataset," https://www.unb.ca/cic/datasets/ids.html, accessed 01-05-2022.
[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.