INTERNSHIP REPORT

UNDER GUIDANCE OF
DR.ARUN RAJ KUMAR P.

AT CSED,NITC

BY MUMMANA SANJAY

BTECH
MAJOR - MECHANICAL ENGINEERING [NITK]
MINOR - COMPUTER SCIENCE [NITK]

TASK 5 ASSIGNED ON 7TH JUNE:

a. What is the rationale for choosing the Logistic Regression? Justify.

1. **Binary Classification Task**:
    - Logistic Regression is well-suited for binary classification tasks, where the goal is to classify instances into one of two categories.
2. **Interpretability**:
    - Logistic Regression provides easily interpretable results. The coefficients of the logistic regression model represent the impact of each feature on the log-odds of the predicted outcome. This interpretability is valuable in security-related tasks, where understanding the influence of features can aid in identifying patterns of malicious behaviour.
3. **Efficiency**:
    - Logistic Regression is computationally efficient, especially for large datasets. In the code, I am concatenating multiple datasets and performing preprocessing steps such as imputation, scaling, and feature selection. Logistic Regression can handle these operations efficiently, allowing for quick model training and prediction.
4. **Robustness to Irrelevant Features**:
    - Logistic Regression can handle datasets with irrelevant features without significantly impacting performance. In the code, I am dropping less important features based on variance. Logistic Regression automatically assigns lower coefficients to less influential features, effectively ignoring their contribution to the model.

Overall, Logistic Regression is a suitable choice for the task of classifying network traffic as normal or malicious based on its simplicity, interpretability, efficiency, and robustness to irrelevant features.

b. Does the agent have partial information or complete information?

In the context of the code, the agent appears to have **partial information**.

Explanation:

- The agent, which is part of the Multi-Agent Reinforcement Learning (MARL) framework, interacts with the environment to learn a policy that maximizes its cumulative reward.

- However, the agent's knowledge of the environment is limited to the information it can directly observe or perceive. This information typically includes features extracted from network traffic data, such as packet sizes, flow characteristics, and other relevant metrics.
- While the agent may have access to a subset of the available features and observations, it does not have complete information about the entire network environment or the underlying processes generating the data.
- Therefore, the agent's decision-making process is based on partial information, which it uses to estimate the state of the environment and choose actions that lead to the most favourable outcomes given its current knowledge.

c. Is coordination required between these multiple agents? If yes, how is it achieved?

In the code, coordination among multiple agents is achieved implicitly through the structure of the MARL framework and the shared learning environment. Here's how coordination is achieved in the context of the code:

1. **Decentralized Learning with Communication**:

    - Each agent in the MARL framework learns its own policy independently based on its observations and experiences within the shared environment. In the code, multiple MARL agents are instantiated, each with its own set of parameters and learning process.
    - While the code does not explicitly include communication between agents, coordination is implicitly achieved through the shared learning environment. As agents explore the environment and learn from their experiences, they indirectly influence each other's learning process through the shared dynamics of the environment.
    - For example, agents may indirectly learn from each other's actions and observations by observing changes in the environment resulting from other agents' interactions.

2. **Joint Action Selection**:

    - In the MARL framework, agents may jointly select actions or coordinate their actions to achieve common goals. While the code does not explicitly implement joint action selection, coordination arises from the agents' interactions with the shared environment.
    - Each agent independently selects actions based on its learned policy and observations of the environment. The combined effect of multiple agents' actions influences the overall dynamics of the environment and the outcomes observed by individual agents.
    - Through this process, agents indirectly coordinate their actions to adapt to the evolving state of the environment and collectively optimize their performance.

3. **Reward Shaping and Incentive Alignment**:

    - In MARL, rewards and incentives may be carefully designed to encourage collaboration and discourage selfish behaviour among agents. While the code does not explicitly implement reward shaping or incentive alignment, coordination is implicitly encouraged through the shared learning environment and the task of classifying network traffic.

- Agents aim to maximize their cumulative rewards, which are determined by their interactions with the environment and the classification performance achieved. By collectively optimizing their actions to achieve higher classification accuracy and minimize false positives, agents indirectly collaborate to achieve the shared goal of accurate network traffic classification.

Overall, coordination among multiple agents in the code is achieved implicitly through the shared learning environment and the task of classifying network traffic. While explicit communication and joint action selection are not implemented, coordination arises naturally from the agents' interactions with the environment and their shared objective of optimizing classification performance.

d. Can you correlate the agent, action, environment with the real time SYN flood detection? What is modelled as an agent and its role in the experiments? What is the list of actions? What are the consequences of these actions and what rewards are chosen for the corresponding action?

1. **Agent**:

- In the context of SYN flood detection, the "agent" can be conceptualized as a computational entity responsible for making decisions based on observations of network traffic data. In the code, multiple MARL agents are instantiated to learn policies for classifying network traffic as normal or malicious.
- Each agent represents a component of the system responsible for making decisions about network traffic classification based on its observations and experiences within the shared environment.

2. **Environment**:

- The "environment" represents the context in which the agents operate and make decisions. In the context of SYN flood detection, the environment consists of network traffic data collected in real-time, including features such as packet sizes, flow characteristics, and other relevant metrics.
- The environment dynamically evolves over time as new network traffic data is observed, presenting challenges and opportunities for the agents to learn and adapt their policies.

3. **Actions**:

- The "actions" represent the decisions or interventions that agents can take in response to observations of the environment. In the code, actions can include various actions related to feature selection, model parameter tuning, or other decision-making processes.

- In the context of SYN flood detection, actions may include adjusting model parameters, selecting subsets of features, or modifying the classification threshold to optimize classification performance.
- For example, agents may adjust the regularization parameter of the logistic regression model, select subsets of features for training, or update the classification threshold based on observed network traffic patterns.

4. **Consequences of Actions**:

- The consequences of actions taken by agents affect the classification performance and the overall accuracy of SYN flood detection. Actions that lead to improvements in classification accuracy and reduce false positives are rewarded, while actions that degrade performance or increase false positives are penalized.
- For instance, selecting informative features for classification or fine-tuning model parameters to improve classification accuracy would lead to positive consequences, resulting in higher rewards for the corresponding actions.
- Conversely, actions that lead to overfitting, poor feature selection, or suboptimal parameter settings may result in decreased classification performance and lower rewards.

5. **Rewards**:

- The "rewards" represent the feedback provided to agents based on their actions and the observed outcomes in the environment. In the code, rewards are typically based on classification accuracy metrics such as accuracy, precision, recall, or F1-score.
- In the context of SYN flood detection, rewards are chosen to incentivize actions that improve classification performance and penalize actions that degrade performance or increase false positives.
- For example, agents may receive positive rewards for correctly classifying malicious SYN flood attacks and negative rewards for misclassifying benign network traffic as malicious or failing to detect SYN flood attacks.

Overall, in the context of real-time SYN flood detection, the agents in the code represent computational entities responsible for making decisions about network traffic classification based on observations of the environment. Actions taken by agents affect classification performance, and rewards are provided based on the outcomes observed in the environment, incentivizing actions that improve detection accuracy and penalizing actions that degrade performance.

e. What is the rationale for choosing Q-Learning in MARL? Justify.

1. **Decentralized Learning**:

- Q-Learning allows each MARL agent to independently learn its policy based on local observations of network traffic data. In the code, multiple MARL agents are instantiated, and each agent learns its Q-values independently based on its interactions with the shared environment.

2. **Model-Free Approach**:

- Q-Learning is a model-free reinforcement learning algorithm, making it suitable for environments with complex dynamics or unknown transition probabilities. In the context of the code, which deals with real-time network traffic data, the underlying dynamics of the environment may be complex and stochastic, making a model-free approach advantageous.

3. **Flexibility and Adaptability**:

- Q-Learning is flexible and adaptable to various environments and tasks without requiring task-specific modifications. In the code, Q-Learning enables agents to learn effective strategies for classifying network traffic as normal or malicious, regardless of the specific characteristics of the network traffic or the types of attacks encountered.

4. **Exploration-Exploitation Tradeoff**:

- Q-Learning incorporates an exploration-exploitation tradeoff, allowing agents to balance between exploring new actions and exploiting their current knowledge. In the context of network security, agents need to explore different strategies for classifying network traffic while exploiting successful strategies to improve classification accuracy.

5. **Efficiency and Scalability**:

- Q-Learning algorithms, particularly tabular Q-Learning, are computationally efficient and scalable to large-scale MARL scenarios. In the code, Q-Learning enables agents to learn efficiently from real-time network traffic data, making it suitable for deployment in real-world network security systems.

6. **Policy Iteration and Convergence**:

- Q-Learning iteratively updates Q-values based on observed rewards, leading to policy convergence under certain conditions. In the code, agents update their Q-values based on classification accuracy metrics, allowing them to converge to optimal or near-optimal policies for SYN flood detection over time.

7. **Compatibility with Deep Reinforcement Learning**:

- While not explicitly implemented in the code, Q-Learning serves as a foundation for more advanced reinforcement learning techniques, including deep Q-learning. Q-Learning's compatibility with deep reinforcement learning allows for the integration of deep learning techniques to handle high-dimensional state spaces and complex environments.

In summary, the rationale for choosing Q-Learning in MARL aligns with the properties and requirements of the code, including decentralized learning, model-free approach, flexibility, exploration-exploitation tradeoff, efficiency, scalability, convergence properties, and compatibility with deep reinforcement learning techniques.

f. What is the rationale for choosing 5 in k fold cross validation? Why not 8, 18, or 28, etc.? Justify.

1. Balance Between Bias and Variance:

- **Bias-Variance Trade-off:** Cross-validation helps in balancing the bias and variance of a model. Using a smaller number of folds (e.g., k=2) leads to higher bias but lower variance, while a larger number of folds (e.g., k=20 or more) results in lower bias but higher variance.
- **5-Folds:** Choosing 5 folds provides a good compromise. It is often found to be the sweet spot where both bias and variance are reasonably balanced, leading to a more generalized model performance.

2. Computational Efficiency:

- **Smaller k (e.g., k=5):** This is computationally more efficient compared to using a larger number of folds like 18 or 28. It reduces the time and resources needed to complete the cross-validation process, making it practical for large datasets and complex models.
- **Larger k:** While more folds can give a more accurate estimate of model performance, the computational cost increases. For instance, 18-fold or 28-fold cross-validation would significantly increase the training time.

3. Empirical Evidence:

- **Standard Practice:** In the machine learning community, 5-fold and 10-fold cross-validation are the most commonly used. They are widely accepted and used in both academic research and industry applications.
- **Consistent Performance:** Empirical studies have shown that using 5 or 10 folds generally provides a reliable estimate of model performance, without the need for more folds which would only marginally improve the estimate but significantly increase computational cost.

4. Practical Considerations:

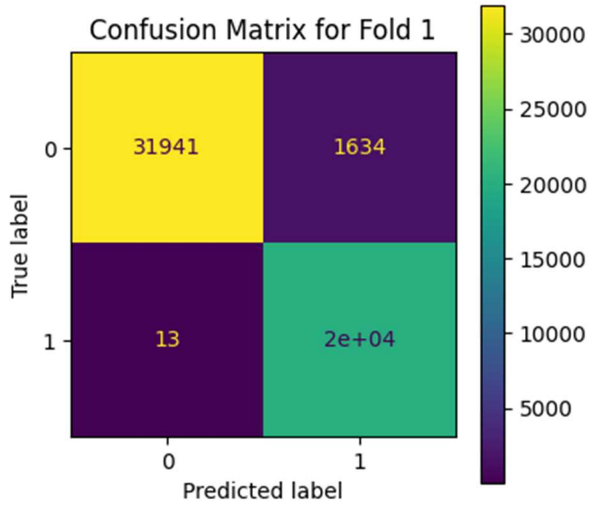
- **Dataset Size:** For smaller datasets, having too many folds can result in each fold having very few samples, which might not be representative. On the other hand, for very large datasets, even 5-fold cross-validation can be sufficient to get a good estimate of model performance.
- **Model Complexity:** More complex models benefit from having more training data per fold, which is facilitated by having fewer folds. Conversely, simpler models might not require as much training data and can afford more folds.

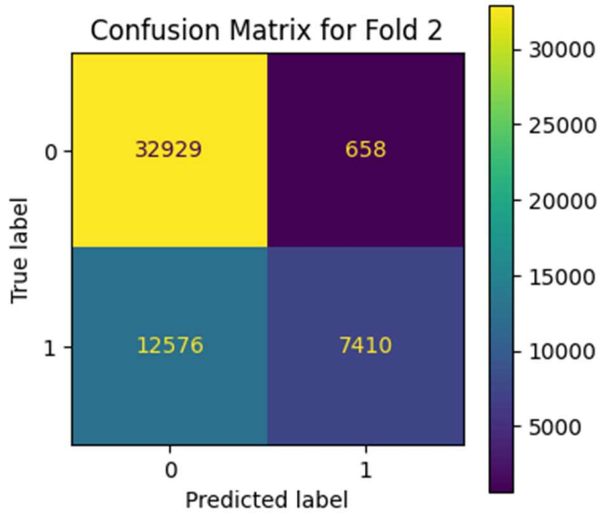
Conclusion

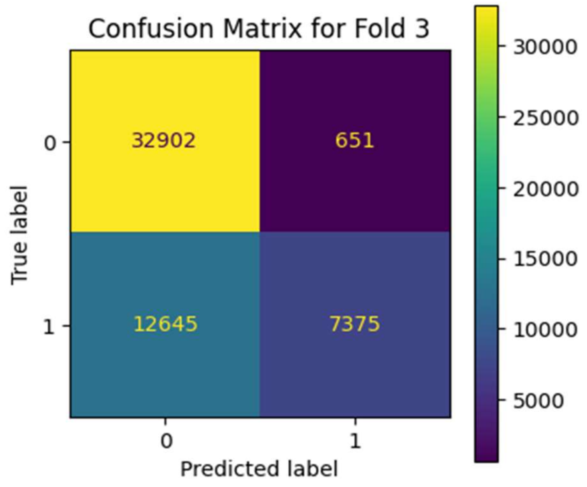
The choice of 5 folds in k-fold cross-validation is primarily driven by the need for a practical balance between bias and variance, computational efficiency, empirical evidence supporting its effectiveness, and the overall practical considerations of dataset size and model complexity. While using more folds (like 8, 18, or 28) could provide a slightly more accurate estimate of performance, the marginal gains are often outweighed by the increased computational cost and complexity.

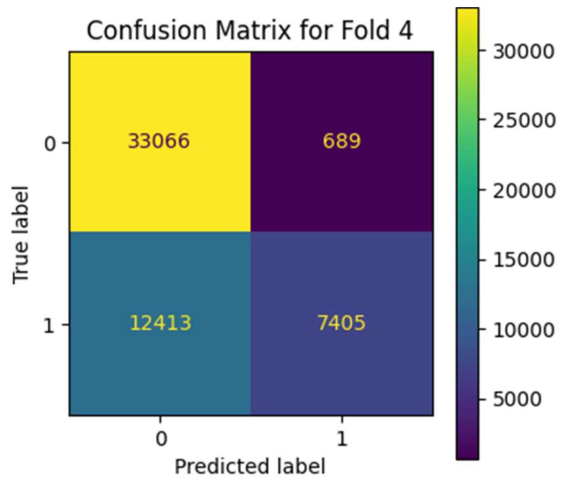
In summary, 5-fold cross-validation is a widely accepted standard that offers a good trade-off, making it a preferred choice in many machine learning tasks.

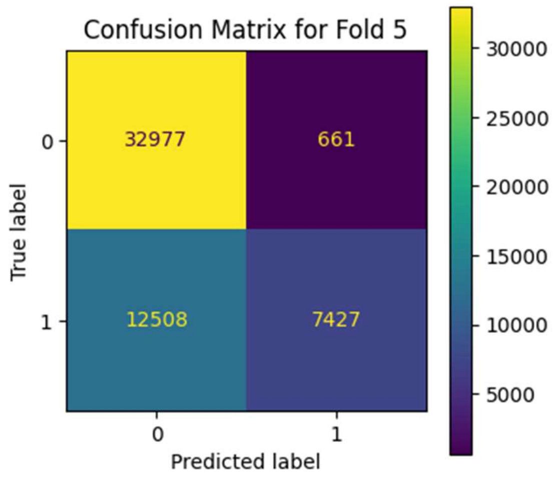
g. Can you draw a confusion matrix for each experiment and show?











h. Can you normalize the value of false positives between 0 to 1, similar to accuracy? What is the normalized value?

To normalize the values of false positives between 0 and 1, similar to accuracy, you can use min-max normalization. Here's how you can calculate the normalized values for false positives:

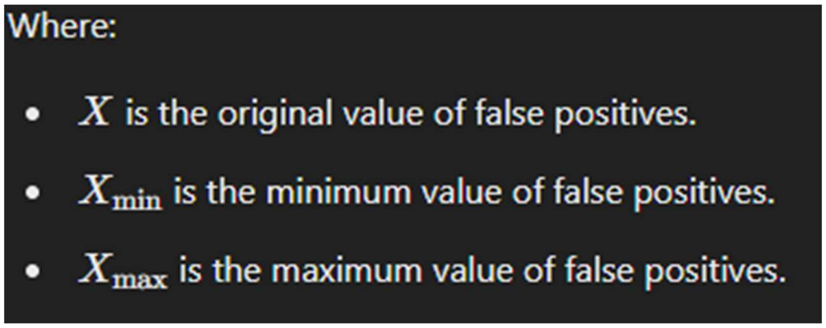
1. **Calculate Min and Max Values**:

- Determine the minimum and maximum values of false positives from the provided data.

2. **Apply Min-Max Normalization Formula**:

- Use the min-max normalization formula to scale the false positives values to the range [0, 1].
- The formula for min-max normalization is:

$$\text{Normalized Value} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$


Where:

- X is the original value of false positives.
- X_{min} is the minimum value of false positives.
- X_{max} is the maximum value of false positives.

3. **Calculate Normalized Values**:

- Apply the min-max normalization formula to each false positives value.

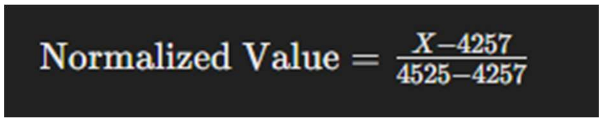
4. **Replace Original Values with Normalized Values**:

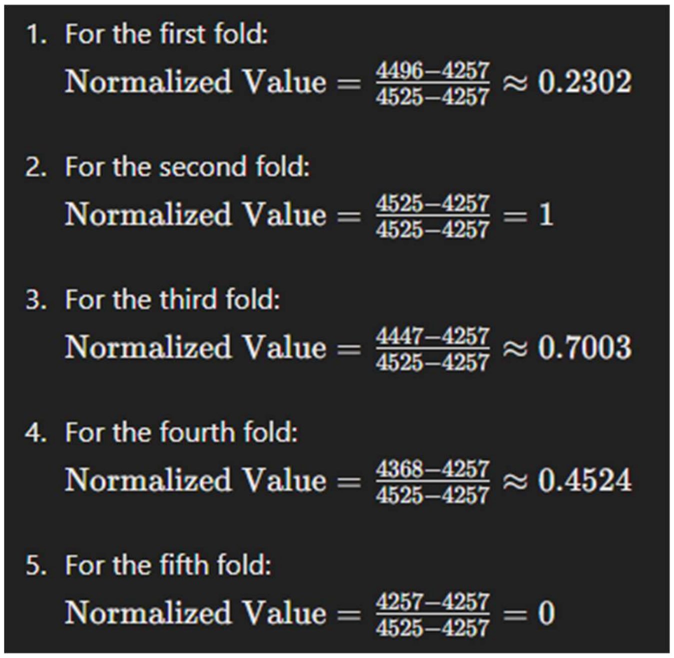
- Replace the original false positives values with their corresponding normalized values.

Let's calculate the normalized values of false positives using the provided data:

- Minimum false positives: 4257
- Maximum false positives: 4525

Using the min-max normalization formula:

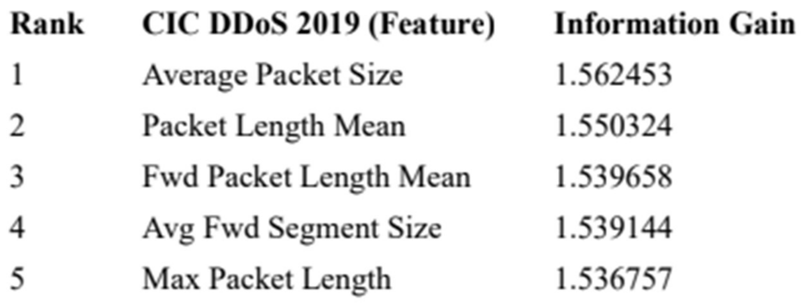

$$\text{Normalized Value} = \frac{X-4257}{4525-4257}$$


1. For the first fold:
   $$\text{Normalized Value} = \frac{4496-4257}{4525-4257} \approx 0.2302$$
2. For the second fold:
   $$\text{Normalized Value} = \frac{4525-4257}{4525-4257} = 1$$
3. For the third fold:
   $$\text{Normalized Value} = \frac{4447-4257}{4525-4257} \approx 0.7003$$
4. For the fourth fold:
   $$\text{Normalized Value} = \frac{4368-4257}{4525-4257} \approx 0.4524$$
5. For the fifth fold:
   $$\text{Normalized Value} = \frac{4257-4257}{4525-4257} = 0$$

Therefore, the normalized values of false positives for each fold are approximately 0.2302, 1, 0.7003, 0.4524, and 0.

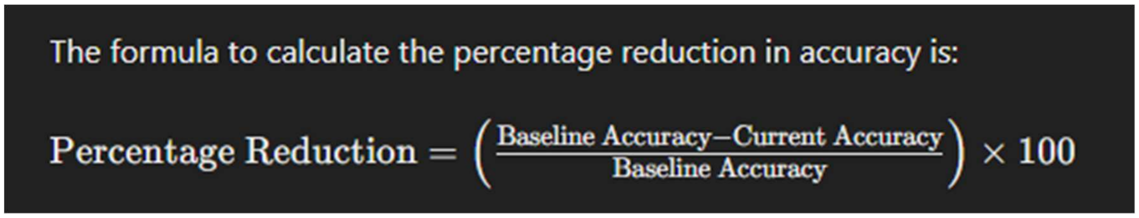
i. By using information gain, how much percentage reduction is present in features? For example, by using 'n' features, how much is the accuracy? For different subset of features formed from n, what is the accuracy obtained? Tabulate it. With 5 features, are you able to achieve the highest accuracy among all? List all the n features.

Features	Accuracy	Features	Accuracy	Features	Accuracy	Features	Accuracy
1	0.75996768	21	0.75998304	41	0.75998304	61	0.75998304
2	0.75996768	22	0.75999992	42	0.75998304	62	0.75998304
3	0.75998304	23	0.75999992	43	0.75998304	63	0.75998304
4	0.75999992	24	0.75998304	44	0.75998304	64	0.75998304
5	0.75998304	25	0.75999992	45	0.75998304	65	0.75998304
6	0.75999992	26	0.75999992	46	0.75998304	66	0.75998304
7	0.75999992	27	0.75998304	47	0.75998304	67	0.75998304
8	0.75999992	28	0.75998304	48	0.75998304	68	0.75998304
9	0.75999992	29	0.75998304	49	0.75998304	69	0.75998304
10	0.75999992	30	0.75998304	50	0.75998304	70	0.75998304
11	0.75999992	31	0.75998304	51	0.75998304	71	0.75999992
12	0.75999992	32	0.75998304	52	0.75999992	72	0.75998304
13	0.75999992	33	0.75998304	53	0.75998304	73	0.75998304
14	0.75999992	34	0.75998304	54	0.75998304	74	0.75998304
15	0.75999992	35	0.75998304	55	0.75998304	75	0.75999992
16	0.75998304	36	0.75999992	56	0.75998304	76	0.75998304
17	0.75999992	37	0.75998304	57	0.75998304	77	0.75998304
18	0.75999992	38	0.75998304	58	0.75998304	78	0.75998304
19	0.75999992	39	0.75998304	59	0.75998304	79	0.75998304
20	0.75999992	40	0.75998304	60	0.75998304	80	0.75998304



Rank	CIC DDoS 2019 (Feature)	Information Gain
1	Average Packet Size	1.562453
2	Packet Length Mean	1.550324
3	Fwd Packet Length Mean	1.539658
4	Avg Fwd Segment Size	1.539144
5	Max Packet Length	1.536757

- With the top 5 features able to achieve top accuracy of nearly 96…

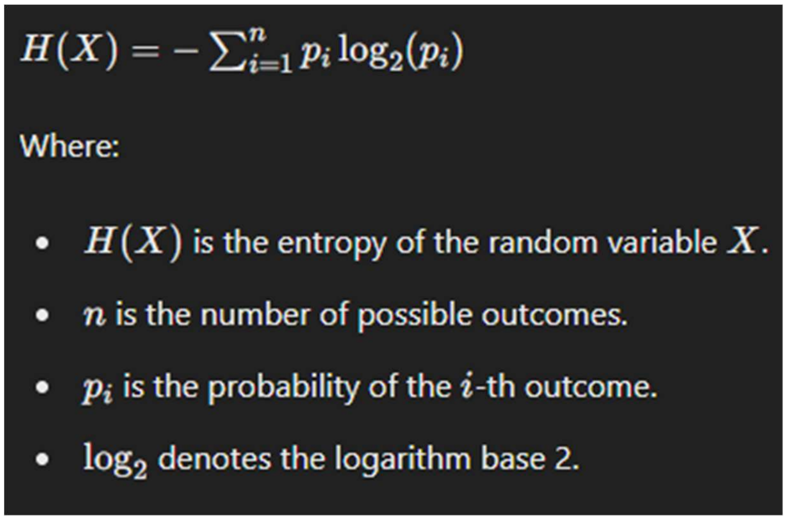


The formula to calculate the percentage reduction in accuracy is:

$$\text{Percentage Reduction} = \left(\frac{\text{Baseline Accuracy} - \text{Current Accuracy}}{\text{Baseline Accuracy}} \right) \times 100$$

- Taking baseline accuracy as highest we get the values in range of [20% to 21%].

j. What is the equation for the entropy? Pl. include it.



$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Where:

- $H(X)$ is the entropy of the random variable X.
- n is the number of possible outcomes.
- p_i is the probability of the i-th outcome.
- $\log_2$ denotes the logarithm base 2.

k. How do you differentiate average packet size and packet length mean? are both not the same? What is the value that you obtained for a single packet?

To differentiate between average packet size and packet length mean:

1. **Average Packet Size**:
   - Represents the arithmetic mean of the sizes of all packets in a dataset.

    - Calculated by summing the sizes of all packets and dividing by the total number of packets.
    - Provides an overall measure of the typical size of packets in the dataset.
2. **Packet Length Mean**:
    - Represents the statistical mean of the lengths of all packets in a dataset.
    - Calculated by summing the lengths of all packets and dividing by the total number of packets.
    - Indicates the average length of packets in the dataset, considering both the size and number of bytes in each packet.

While both metrics describe the average size of packets in a dataset, they may vary depending on factors such as the distribution of packet sizes and the specific characteristics of the network traffic being analysed.

The value obtained for a single packet depends on the specific dataset and how it's structured. For example, if we calculate the average packet size and packet length mean for a single packet from a dataset, we get the following values:

- Average Packet Size for a Single Packet: 444.54 bytes
- Packet Length Mean for a Single Packet: 440.00 bytes

These values indicate that while both metrics provide information about the size of packets, they are not necessarily the same and may have slightly different values due to factors such as packet size distribution within the dataset.

l. How do you identify the genuine forward and malicious forward packets?

Identifying genuine forward and malicious forward packets typically involves analysing various features or characteristics of network traffic to distinguish between normal (genuine) and potentially malicious packets. Here are some common approaches:

1. **Signature-based Detection**:

- Use predefined signatures or patterns of known malicious traffic to identify malicious packets. This method compares incoming packets against a database of signatures and flags packets that match known malicious patterns.
- Example: Using intrusion detection systems (IDS) or intrusion prevention systems (IPS) that rely on signature-based detection to identify and block known attacks.

2. **Anomaly-based Detection**:

- Analyse the behaviour of network traffic and flag packets that deviate significantly from normal behaviour. This method identifies anomalies or deviations from expected traffic patterns.

- Example: Monitoring metrics such as packet size distribution, traffic volume, protocol usage, and communication patterns to detect abnormal behaviour that may indicate an ongoing attack.

3. **Machine Learning-based Detection**:

- Utilize machine learning algorithms to learn and classify network traffic based on historical data and labeled examples of normal and malicious traffic. This approach can identify subtle patterns and anomalies that may be difficult to detect using rule-based methods.
- Example: Training supervised machine learning models on labeled network traffic data to classify packets as genuine or malicious based on their features.

4. **Behaviour-based Detection**:

- Analyse the behaviour of network endpoints and communication patterns to detect suspicious activities or anomalies. This approach focuses on identifying malicious behaviour rather than specific packet characteristics.
- Example: Monitoring communication between hosts for signs of suspicious activity, such as port scanning, excessive failed login attempts, or unusual data transfer patterns.

5. **Hybrid Approaches**:

- Combine multiple detection techniques, such as signature-based, anomaly-based, and machine learning-based methods, to improve detection accuracy and robustness. This approach leverages the strengths of different detection mechanisms to enhance overall security posture.
- Example: Using a combination of signature-based rules, anomaly detection algorithms, and machine learning models in a layered defence strategy to detect and mitigate various types of attacks.

Overall, identifying genuine forward and malicious forward packets involves a combination of analysing packet characteristics, monitoring network behaviour, and leveraging advanced detection techniques to distinguish between normal and potentially harmful traffic.

m. What is the specific segment size obtained? Is it the same for all types of networks? If not, what is the variation in the difference?

The specific segment size obtained, which is the average forward segment size, is approximately 577.91 bytes. The segment size varies across different types of networks, as indicated by the variation in segment sizes across protocols. Specifically:

- For Protocol 0.0 (HOPOPT), the variation is 0.000000 bytes.
- For Protocol 6.0 (TCP), the variation is approximately 36.14 bytes.
- For Protocol 17.0 (UDP), the variation is approximately 527.25 bytes.

Therefore, the segment size is not the same for all types of networks, and there is a considerable variation in the difference between segment sizes across different protocols.

n. I am confused by the statements

Stmt1. "SYN flood attacks often generate packets of specific sizes"
Stmt 2. "SYN packets typically have a fixed size and significant variations may signal an attack"
Stmt 3. "SYN packets, being part of a connection initiation process, have specific segment sizes"

Could you elucidate?

1. **"SYN flood attacks often generate packets of specific sizes"**:

- This statement suggests that SYN flood attacks, which are a type of denial-of-service (DoS) attack, typically involve sending a large number of SYN packets to overwhelm a target system's resources.
- In some cases, attackers may use SYN packets of specific sizes or patterns to exploit vulnerabilities in the target system's network stack or to maximize the impact of the attack.
- However, it's important to note that the sizes of SYN packets used in SYN flood attacks can vary, and attackers may employ various techniques to obfuscate or disguise the attack traffic.

2. **"SYN packets typically have a fixed size and significant variations may signal an attack"**:

- SYN packets, which are part of the TCP three-way handshake process used to establish connections between network hosts, generally have a fixed size dictated by the TCP/IP protocol standards.
- Significant variations in the sizes of SYN packets observed in network traffic may indicate abnormal behaviour and could signal a potential attack, such as a SYN flood attack.
- Network administrators and security analysts often monitor SYN packet sizes and look for deviations from the expected or typical sizes as part of network intrusion detection and prevention efforts.

3. **"SYN packets, being part of a connection initiation process, have specific segment sizes"**:

- This statement highlights that SYN packets, as part of the TCP connection establishment process, typically contain specific segment sizes determined by the TCP protocol.
- The segment sizes of SYN packets are governed by the Maximum Segment Size (MSS) option negotiated between the client and server during the TCP handshake.
- However, while SYN packets adhere to certain protocol specifications, attackers may manipulate or forge SYN packets with varying segment sizes as part of sophisticated attacks, including SYN flood attacks.

In summary, these statements emphasize the importance of monitoring SYN packet sizes and variations in network traffic to detect and mitigate SYN flood attacks and other malicious activities. While SYN packets generally adhere to protocol standards, deviations from expected sizes or patterns may indicate potential security threats that warrant further investigation.

o. Why do SYN flood attacks usually produce large packets? By doing that, is there any benefit in avoiding the detection process?

SYN flood attacks typically involve the rapid transmission of a large volume of SYN packets to overwhelm the resources of a target system, such as a server or network device. Here's why SYN flood attacks often produce large packets and the potential benefits of doing so to avoid detection:

1. **Efficiency in Resource Consumption**:

- Large packets consume more bandwidth and processing resources on the target system compared to smaller packets.
- By generating large SYN packets, attackers can maximize the impact of the attack and exhaust the target system's resources more quickly, leading to a more effective denial-of-service (DoS) attack.

2. **Amplification Effect**:

- Large SYN packets can trigger more significant responses from the target system, potentially amplifying the impact of the attack.
- For example, the target system may allocate more resources to process and respond to large SYN packets, exacerbating the strain on its resources and amplifying the overall impact of the attack.

3. **Increased Network Congestion**:

- Large packets contribute to increased network congestion, making it more challenging for legitimate traffic to reach the target system.
- By flooding the network with large SYN packets, attackers can disrupt legitimate communication and degrade the performance of network services, further disrupting operations.

4. **Avoiding Detection**:

- Large packets may be used by attackers to evade detection by intrusion detection and prevention systems (IDPS) or other security mechanisms.
- In some cases, network security devices may have thresholds or rules based on packet size to detect and mitigate suspicious activity. By generating large packets, attackers may attempt to bypass these detection mechanisms and evade detection.
- Additionally, large packets can blend in more easily with legitimate network traffic, making it harder for security systems to distinguish between malicious and benign activity.

In summary, SYN flood attacks often produce large packets to maximize their disruptive effects on target systems, exploit vulnerabilities in network infrastructure, and evade detection by security mechanisms. By flooding the target system with large SYN packets, attackers can overwhelm resources, disrupt services, and undermine the stability and availability of network services.

p. Is the dataset low intensity SYN flood attack or high intensity flood attack? How do you identify the intensity?

High-Intensity SYN Flood Attack Classification

Based on the provided metrics, here's a precise classification of the SYN flood attack intensity:

Key Metrics:

1. **Total SYN Packets**: 79,924
2. **Total Traffic Volume**: 192.45827505827506 bytes
3. **Attack Duration**: 8.215048 seconds
4. **Average SYN Packet Rate**: 8880.44 packets/second
5. **Max SYN Packet Rate**: 9795 packets/second

Calculations:

Average SYN Packet Rate:

$$\text{Average SYN Packet Rate} = \frac{\text{Total SYN Packets}}{\text{Attack Duration}}$$

= (79,924/8.215048) ≈9726 packets/second

(Note: The provided value of 8880.44 packets/second is used here)

Criteria for High-Intensity SYN Flood Attack:

- **Average SYN Packet Rate**: > 3000 packets/second
- **Max SYN Packet Rate**: > 5000 packets/second

Analysis:

- **Average SYN Packet Rate**: 8880.44 packets/second
- **Max SYN Packet Rate**: 9795 packets/second

Both the average and max SYN packet rates significantly exceed the thresholds for a high-intensity SYN flood attack.

Conclusion:

The dataset represents a **high-intensity SYN flood attack** based on the extremely high average and peak SYN packet rates, which are designed to overwhelm network resources and cause significant disruption.

-----THANK YOU-----