Altering network traffic characteristics for testing ML-based IDS

Bartan Oren, Tudor Dascalu, and Maitraiyi Dandekar

Eindhoven Institute of Technology

Abstract. Intrusion Detection Systems (IDS) are vital for ensuring network security by identifying unauthorized access and anomalies in network traffic. Machine Learning-based IDS (ML-based IDS) promises enhanced detection capabilities but faces challenges due to the inherent characteristics of network data, such as non-stationarity, imbalance, and context dependency. This project aims to develop a tool to manipulate these characteristics in network traffic captures, maintaining data realism and ground truth accuracy. The primary objectives are to explore methods for altering network traffic captures and adjusting corresponding ground truth labels, thereby facilitating more robust testing of ML-based IDS. The outcomes will support future research into the impact of these data characteristics on IDS performance and help refine IDS models to handle real-world network traffic more effectively.

Keywords: Intrusion Detection Systems (IDS) · Machine Learning-based IDS (ML-based IDS) · Network Traffic Characteristics · CICIDS-2017 dataset · Ground Truth Labels · Network Behavior Patterns · Imbalance · Context Dependency · Adversarial Behavior · Evolving Behavior · Unpredictable Behavior

1 Introduction

The increasing complexity and frequency of cyberattacks necessitate advanced security measures, among which Intrusion Detection Systems (IDS) play a crucial role. IDS are designed to detect unauthorized access and anomalies in network traffic, ensuring data protection and network security. There are two primary types of IDS: Network-based IDS (NIDS) and Host-based IDS (HIDS). The integration of Machine Learning (ML) into IDS (ML-based IDS) enhances their ability to adapt to new attack patterns and analyze large volumes of data in real time.

However, ML-based IDS face significant challenges due to certain characteristics of the data they analyze. These characteristics include non-stationarity, caused by a change in normal behavior, caused by a change in malicious behavior, and caused by the unpredictability of the network environment, and imbalance, where there are disproportionately more benign connections than malicious ones. Additionally, context dependency complicates the differentiation between benign and malicious behavior, as it requires additional contextual information for accurate detection.

2 Background context

The primary objective of this project is to develop a tool that can alter these characteristics in network traffic captures while maintaining the realism of the data and ensuring the validity of the ground truth. This tool aims to support the evaluation of ML-based IDS by providing a means to systematically vary the data characteristics and observe the effects on IDS performance. The project focuses on several key characteristics: imbalance between benign and malicious and within different types of malicious connections, unpredictable behavior, evolving behavior, adversarial behavior, and context dependency where benign and malicious behaviors are similar.

The key characteristics of network traffic relevant to this project are defined as follows:

1. **Imbalance**: This refers to the disproportionate number of benign versus malicious connections. Additionally, there can be an imbalance within malicious connections themselves, where certain types of malicious connections are overrepresented compared to others.
2. **Unpredictable Behavior**: This involves changes in network behavior caused by deviations from expected patterns. These deviations are not easily predictable and can affect the performance of IDS.
3. **Evolving Behavior**: This characteristic refers to changes in the behavior of monitored entities over time due to environmental evolution. For example, the typical usage patterns of network services may change as new features are added or as users adapt their behaviors.
4. **Adversarial Behavior**: This includes changes in the behavior of attackers who develop new evasion techniques or attack capabilities. These adaptive strategies by adversaries can complicate detection efforts by IDS.
5. **Context Dependency**: This characteristic highlights the similarities between benign and malicious behaviors that require additional contextual information for accurate detection. IDS must consider contextual factors to differentiate between benign anomalies and actual threats.

To achieve these goals, the project addresses two main research questions: (1) How can network traffic captures be efficiently modified to alter the degree to which a characteristic is present? (2) How can such modifications be reflected in adjustments to a ground truth that was valid before the modifications, ensuring the adjusted ground truth remains accurate? The answers to these questions guide the development of methodologies for altering network traffic captures and adjusting corresponding ground truth labels.

The significance of this project lies in its potential to enhance the robustness of ML-based IDS by enabling more comprehensive testing against varied network traffic scenarios. By systematically manipulating data characteristics and observing the impact on IDS performance, researchers can gain insights into the strengths and weaknesses of different IDS models. This research contributes to the broader goal of improving cybersecurity measures in an increasingly digital and interconnected world.

3 Dataset

The CICIDS2017 dataset, created by the Canadian Institute for Cybersecurity, was chosen for this project due to its comprehensive nature and realistic simulation of network traffic over five days. Unlike older datasets such as DARPA 1998/1999/2000, NSL-KDD, and MAWILab, which suffer from limitations like lack of traffic diversity, insufficient volumes, limited temporal coverage, and inadequate ground truth and labeling, the CICIDS2017 dataset provides a robust foundation for NIDS research.

Key characteristics of the CICIDS2017 dataset include:

- Realistic Traffic Simulation: The dataset simulates real-world network traffic, capturing various normal behavior patterns and multiple attack scenarios.
- Detailed Labels: It includes extensive labeling for both benign and malicious traffic, facilitating accurate training and testing of ML-based IDS.
- Comprehensive Benchmark: By incorporating diverse attack vectors and normal activities, it serves as an effective benchmark for evaluating IDS performance.
- Traffic Analysis Results: The dataset comes with network traffic analysis results using CICFlowMeter, which generates CSV files containing extracted connection flows and multiple features from raw PCAPs, labeled according to their maliciousness.
- The CICIDS2017 dataset's detailed labeling, traffic diversity, and volume make it a preferred choice for this project, ensuring that the developed tool can effectively test ML-based IDS under realistic conditions and varied network scenarios.

4 Methodology

In this project, the analysis and processing of network traffic data from PCAP[1] files utilized two primary tools: Suricata and Zeek. Both tools were selected for their specific capabilities and their prevalence in the field of network security, ensuring that the methodology remains generalizable and applicable in various contexts.

4.1 PCAP Pre-processing

Suricata: Suricata is a signature-based network intrusion detection system (IDS) that utilizes predefined rules to identify known malicious, suspicious, or abnormal network traffic [2]. It was chosen for this project due to its open-source nature, multi-threaded architecture, and widespread use in the industry, which supports the generalizability of the research. Suricata processes network traffic through four main steps: capturing packets at sensors, decoding these packets

[1] PCAP files are a common format for storing packet captures containing a copy of every byte of every packet as seen on the network

to extract information from different network layers, evaluating rules in parallel, and outputting the classification results.

Suricata's rule-based system allows it to make judgments on packet information in conjunction with contextual states. These states enable complex detection rules, making Suricata capable of handling varying network scenarios effectively. Features such as thresholds, flowbits, and xbits allow Suricata to maintain state information across multiple packets and flows, enhancing its detection capabilities.

For this project, Suricata was configured to focus on significant security events by extracting alerts and filtering out less critical events. Specifically, only events with an `event_type` of alerts were retained, and those containing `POLICY`, `INFO`, or `HUNTING` in their rule names were excluded. This filtering ensured that the analysis concentrated on critical alerts indicative of potential security threats. Additionally, not all Suricata rules included the `mitre_technique_id` field, which is essential for correlating alerts with specific `MITRE ATT&CK` techniques.

The `MITRE ATT&CK` framework [3] is a comprehensive matrix of tactics and techniques used by adversaries during cyberattacks. Each tactic and technique in the framework is assigned a unique identifier known as the MITRE Tactic ID which comprises several MITRE Technique IDs. This identifier helps standardize the categorization of attack methods, facilitating the correlation and analysis of security events across different tools and platforms. The alerts generated can be mapped directly to specific attack techniques described in the `MITRE ATT&CK` framework by tagging Suricata rules with the relevant MITER Technique IDs. This enhances the analytical capabilities by providing a structured and universally recognized context to the detected events.

Therefore, before running Suricata on the PCAP files, the rules were tagged to include the `mitre_technique_id`, ensuring comprehensive and accurate categorization of alerts. This step was critical for enabling detailed and standardized analysis of the detected security incidents.

Zeek: Besides Suricata, the PCAPs were also analyzed by Zeek [4] to log detailed information about network connections within the PCAP files. Zeek was chosen for its powerful and flexible logging capabilities, which provide extensive visibility into network activities. It records crucial details such as source and destination IP addresses, ports, protocols, and timestamps for each connection, which are essential for establishing a baseline of normal network behavior and understanding the context of network events. Unlike Suricata, which focuses on alerting for malicious activities, Zeek provides a broader context by summarizing all network events, thus enabling comprehensive network security monitoring.

Zeek processes network traffic by capturing packets and converting them into a stream of events that reflect network activities neutrally. Examples of logs produced by Zeek include connection logs summarizing TCP connections, DNS logs summarizing DNS queries, and SSH logs detailing SSH login attempts.

The integration of Suricata and Zeek facilitated a thorough and detailed analysis of the network traffic. Initially, Suricata generated alerts based on the predefined rules, which were then filtered to retain only those with significant security relevance. Simultaneously, Zeek logged all network connections. The next step involved correlating these logs with the filtered Suricata alerts by identifying common connections.

Ground Truth Filtering: To ensure the accuracy and relevance of the analysis, the correlated events were further filtered based on the ground truth available in the provided CSV files. These CSV files contained labeled data indicating the correct classification of connections as malicious or benign. Only these correctly classified connections were processed further. This step was crucial for maintaining the integrity of the ground truth and ensuring that subsequent analyses and alterations were based on accurate and reliable data.

By leveraging Suricata's rule-based detection and alerting capabilities and Zeek's comprehensive connection logging, this methodology provided a robust framework for analyzing network traffic. It focused on critical security events while ensuring the accuracy of the processed data through correlation with established ground truth labels. This approach not only enhanced the reliability of the analysis but also ensured that the findings would be applicable and valuable in broader network security contexts.

4.2 Characteristics Altering

Our implementation is based on a pipeline of different processors. Each processor can be initialized to alter one of the data characteristics along with the corresponding ground truth file. Thus, we developed a specific processor with its parameters for each characteristic. These processors as well as how the alteration is performed are described in this section.

Imbalance Class imbalance, where benign traffic vastly outnumbers malicious traffic, presents challenges for effectively assessing the performance of Intrusion Detection Systems (IDS). This imbalance skews performance metrics, making it difficult to distinguish between genuinely effective detection methods and those that perform well simply due to the predominance of benign traffic. The Imbalance Processor was designed to manipulate the PCAP files to address class imbalance. Class imbalance can be categorized into two types: imbalance between classes and imbalance within classes.

Imbalance Between Classes: This refers to the disproportionate number of benign versus malicious connections. Typically, the amount of benign traffic significantly exceeds the amount of malicious traffic, which can mislead the performance assessment of IDS. For instance, a dataset might have a ratio of 1 malicious connection to 100,000 benign connections, making it challenging to detect rare malicious activities accurately.

Imbalance Within Classes: This pertains to the uneven distribution of different types of malicious traffic. Within the malicious traffic class, certain attack types might be overrepresented compared to others. For example, a dataset might have a high number of network scan attacks but very few instances of more sophisticated attacks, affecting the IDS's ability to generalize across different attack types.

The Imbalance Processor manipulates PCAP files based on their flow keys to address class imbalance. A Flow Key is defined as a tuple consisting of source IP, destination IP, source port, destination port, and protocol. This key uniquely identifies a flow of packets within the network traffic. All packets sharing the same flow key are considered part of the same connection or session.

The processor is initialized with the following parameters:

1. `malicious_mutation_percentage`: This parameter determines the mutation ratio of malicious packets. Values between -1 and 0 indicate the removal of packets, with -1 removing all malicious packets. Values between 0 and 1 indicate the duplication of packets, with 1 duplicating all malicious packets.
2. `benign_mutation_percentage`: Similar to the malicious mutation percentage, this parameter determines the mutation ratio of benign packets. Values between -1 and 0 indicate the removal of packets, with -1 removing all benign packets. Values between 0 and 1 indicate the duplication of packets, with 1 duplicating all benign packets.
3. `mutation_chance`: This parameter defines the probability that any given flow in the mutation list will be mutated.
4. `intra_class_mutation_ratio`: This parameter controls the mutation ratio of intra-class activities. Values between -1 and 0 indicate the removal ratio of the most occurring attacks, while values between 0 and 1 indicate the duplication of the least occurring attacks.

The processor operates in two modes: *deletion* and *mutation*. It determines whether to operate in deletion or mutation mode based on the values of the `malicious_mutation_percentage` and `benign_mutation_percentage`.

The processor addresses the imbalance between classes by removing or duplicating malicious packets. Malicious connections are sampled based on the `malicious_mutation_percentage`; if negative, a fraction of malicious packets is removed; if positive, a fraction is duplicated. Similarly, benign connections are handled by adding benign flows to the removal list if in deletion mode or to the mutation list if in mutation mode.

For intra-class imbalance, the processor analyzes the distribution of different attack types. Attack types are derived from the `MITRE ATT&CK` framework, and are classified by MITRE Tactic ID. MITRE Tactic ID was preferred over the MITRE Technique ID because the focus was on the abstract classification of the attack. Tactics represent higher-level categories of malicious behavior, for e.g. privilege escalation or defense evasion, providing a broader understanding of the attack landscape. This approach avoids the need to re-classify individual techniques into broader categories, streamlining the analysis and manipulation of attack data. It identifies the most and least frequent attack types and samples flows based on the `intra_class_mutation_ratio` to either mutate less frequent attack types or remove more frequent ones.

During processing, each packet is evaluated against the removal and mutation lists. In deletion mode, packets in the removal list are stripped of their payloads. In mutation mode, packets in the mutation list have their timings and ports adjusted. The processor applies these mutations by shifting packet times and assigning new ephemeral ports to simulate realistic network traffic variations, maintaining the integrity of the original flows while addressing class imbalance.

Unpredictability Unpredictable behavior can be interpreted as how actual behavior deviates from expected behavior. Thus the behavior of individual hosts in a network may vary either on a short period, minutes/hours, or on a longer time frame. As the connections dataset is required to be kept as realistic as possible, it is expected that the hosts have multiple connections throughout the day that could be altered. Hence, for unpredictability, the change done by the processor will be on batches of 10.000 packets from the PCAP file. This design choice reflects the idea of altering different hosts in each batch to increase the irregularity and variations of connection properties at different time frames. The processor requires only one parameter:

1. `mutation_chance`: This represents the chance of a host to be altered in a batch. The type of the parameter should be a float which defines the probability of choosing the host for further processing.

Taking the previous into account, the very first action done by the processor is to sample a set of hosts depending on the `mutation_chance`. Using Zeek, we can retrieve all the hosts that are present in our PCAP file, and thus by applying a uniform random sample we get the hosts that will be altered. Additionally, the processor stores a dictionary containing all the flows of the hosts that will be mutated. Therefore, in each batch, we check to see if there is a flow that should be removed, and if we find one, then we simply remove it from the output PCAP file. As a flow is removed from a batch, we also ensure that the respective hosts are deleted from the list of mutations. This action enforces the variation in the dataset as in each batch different hosts will be altered.

Evolving Behaviour As large networks change over time, the evolving behavior characteristic is reflected in a change of content and a change in the interactions between hosts. Our approach aims at the interaction between hosts and alters the pattern of host pair connections. To this extent, the evolving processor we define modifies the behavior of hosts in a random time frame. As parameters, the processor takes the number of hosts that will be altered:

1. `n_hosts`: This reflects the percentage of hosts that will be modified. If the number is equal to 1, then only one host will be uniformly sampled from all the possible hosts present in the dataset. If the number is greater than 1 then it will represent the percentage of hosts selected for processing.

After selecting the hosts, we need to consider the timeframe in which their connections will be altered as the characteristic is dependent on time. As a solution to this, we generate a random timestamp for each host and then verify if that host has activity both before and after the timestamp. In this way, we ensure that the modifications done by the processor will influence the evolving characteristic of the dataset. In the case of having connections only before/after the timestamp, then a new random time is generated and the check happens again. This process of sampling hosts repeats until we end up with a valid set of hosts. Afterward, we use a coin toss for each host to randomly select if we modify the connections before or after the timestamp. After this part of preprocessing, we simply delete the connections of the hosts according to the random variables explained.

Adversarial Behaviour Adversarial behavior is characterized by three primary dimensions:

– Change in Attack Source: This captures changes in the origin of an attack. Attackers often use proxies or subvert IP addresses to mask their true location. The visible aspect is the gateway used for sending attacks, which can change over time to evade detection.
– Change in Attack Target: This captures changes in the victims of the attacks. An attacker may shift their focus to different hosts or types of hosts within a network, reflecting a change in the composition of targets.
– Change in Attack Vector: This captures changes in the methods or techniques used in attacks. An attack vector includes the specific vulnerabilities or exploits leveraged by the attacker. Changes in attack vectors may indicate the discovery of new methods or an expansion of the attack surface.

To address the adversarial behavior characteristic in network traffic, the Adversarial Processor manipulates PCAP files based on flow keys and specific parameters to simulate realistic adversarial tactics. A flow key—comprising source IP, destination IP, source port, destination port, and protocol—uniquely identifies a connection. The processor initializes with the following parameters to control how adversarial behaviors are manipulated:

1. `modifications`: A list of tuples specifying modifications for attack types and corresponding function indicators. Each tuple consists of an attack type and an integer indicating the modification function (e.g., ('misc-activity', 1)).
   Function indicators:
   1 : Increase attack occurrences over time (removal ends earlier).
      - > 1: packet removal ending even earlier.
      - $0 < value < 1$: packet removal ending later.
   0 : No change in attack occurrences (constant rate).
   -1 : Decrease attack occurrences over time (removal starts earlier).
      - < -1: packet removal starting even earlier.
      - < 0 and > -1: packet removal starts later

2. `attack_source_alter`: Frequency at which the attack source IP and port will change (e.g., 3 means the attacker source will change every 3 attacks).

The processor operates in two primary modes: *altering attack sources* and *modifying attack vectors* over time.

Altering Attack Sources: If `attack_source_alter` is set, the processor will change the source IP and port of the attacker at the specified frequency. This simulates the adversary changing tactics to evade detection. The processor generates new random IP addresses and ephemeral ports, ensuring the new IPs are of the same type as the original flows. This simulates the Change in Attack Source.

Modifying Attack Vectors Over Time: If `modifications` are provided, the processor adjusts the rate of attack occurrences over time based on the specified function indicators. This simulates the dynamic nature of adversarial behavior. The processor initializes filter rates for each attack type and modifies them according to the steepness parameter derived from the function indicators. Higher values indicate a more significant change in the rate of attack occurrences. This simulates a change in the attack vector.

During packet processing, the processor checks each packet against the flow keys to determine if it needs alteration. If in the attack source alteration mode, the processor changes the packet's source IP and port to the new values. If in the attack vector modification mode, the processor filters out packets based on the adjusted rates.

By manipulating the network traffic in these ways, the Adversarial Processor creates a more realistic simulation of adversarial behavior. This helps in testing and evaluating the robustness of Intrusion Detection Systems (IDS) against sophisticated and evolving attack strategies.

Context Dependency Network intrusion detection systems (NIDS) classify network traffic as either benign or malicious based on observed patterns. The decisions made by these systems depend heavily on the context in which the traffic is analyzed. To address the context dependency characteristic in network traffic, the Context Dependency Processor manipulates PCAP files based on flow keys and specific parameters to simulate realistic context-dependent and context-independent attack scenarios. The processor initializes with one parameter to control how context dependency behavior is manipulated:

1. `context_dependency_ratio`: A parameter that controls the deletion ratio of context dependency activities:
   - Values between -1 and 0 indicate the removal of context-dependent attacks.
   - Values between 0 and 1 indicate the removal of context-independent attacks.

For addressing context dependency, the MITRE Technique ID was preferred because context dependency is determined in terms of specific techniques. This approach allows for a detailed analysis of whether a particular technique is more context-dependent or context-independent. The processor operates by identifying context-dependent and context-independent attacks using these MITRE Technique IDs and then applying the specified mutations.

The processor identifies unique attack techniques from the malicious dataframe and categorizes them into context-dependent and context-independent attacks based on predefined sets of MITRE Technique IDs. Context-dependent attacks are those that require additional context for accurate detection (e.g., T1055, T1071), while context-independent attacks do not rely on such context.

Based on the `context_dependency_ratio`, the processor determines whether to remove context-dependent or context-independent attacks, filtering the malicious dataframe to create a list of flows that need to be removed according to the specified ratio. During packet processing, the processor checks each packet against the removal list, stripping the payloads of packets belonging to flows in the removal list and effectively removing them from the dataset. The processor maintains a list of all processed flows to ensure consistency and accuracy in manipulation.

By leveraging the context dependency characteristics and `MITRE ATT&CK` framework, the Context Dependency Processor manipulates network traffic to increase or decrease the contextual dependency of the dataset. This helps in testing and evaluating the robustness of Intrusion Detection Systems (IDS) against complex attack strategies that rely on contextual information for accurate detection.

5 Evaluation

For the evaluation of our implementation, a tool [5] provided by our supervisor was used to compute the different values of each characteristic before and after the alteration. During this validation part, the *Friday-WorkingHours.pcap* file was used from the CICIDS2017 dataset as it consists of a large number of packets along with a variety of attacks during the entire timeframe of the network traffic capture.

To get a better understanding of the results shown below, it is important to understand how to interpret them. The scores presented are normalized between 0 and 1 and for each characteristic the score reflects a specific dimension of the dataset. This minimal difference between scores of original and altered data is due to the alterations maintaining the realism of the data. Given that the PCAP files contain millions of packets, altering or deleting thousands of packets only caused a difference up to the third decimal.

5.1 Imbalanced Classes

The evaluation of these scenarios from Table 1 reveals that different mutation strategies can influence the imbalance characteristics in network traffic. Applying a -0.3 malicious mutation chance slightly reduces the aggregate imbalance and

Scenario	Aggregate %	Between %	Within %
Original	0.734057	0.983706	0.484408
-0.3 malicious mutation chance	0.73**39**	0.9837**84**	0.484**120**
0.5 malicious mutation chance and -0.01 benign mutation chance	0.73**2140**	0.98**1249**	0.48**3030**
-0.5 balancing within	0.73**3955**	0.9837**89**	0.484**120**

Table 1. Imbalanced Class scores for three different types of alteration

within-class variability without affecting the between-class imbalance. Increasing malicious traffic while slightly reducing benign traffic (0.5 malicious mutation chance and -0.01 benign mutation chance) has a more pronounced effect on reducing both aggregate and between-class imbalances, as well as within-class variability. Finally, balancing within the malicious class by removing the most frequent attack types (-0.5 balancing within) results in minor reductions in aggregate and within-class imbalances, with negligible impact on the between-class imbalance.

5.2 Unpredictable Behavior

For the unpredictable behavior, two dimensions can be measured to check the alteration we implemented. The first dimension is *periodicity* which describes the proportion of network connections that can be considered periodic. Thus, having a higher periodicity in the dataset will result in a negative impact on the score. The other dimension is *inconsistency* and reflects the irregularity and variations of connection properties at different points in time. In this case, a higher score will indicate a higher variance of the connections' presence throughout the entire dataset.

Scenario	Aggregate %	Periodicity (-) %	Inconsistency (+) %
Original	0.435742	0.539370	0.410855
0.1 chance	0.435**812**	0.5**42353**	0.41**3977**
0.2 chance	0.43**0254**	0.5**55397**	0.41**5906**

Table 2. Unpredictable Behavior scores showing the impact of the mutation chance

The results of 2 different alterations of the dataset are presented in Table 2. Firstly, we altered the original dataset with a 0.1 change of deleting a host from a batch. Then we checked to see if a bigger increase in the removal probability would significantly change the periodicity score. It can be noticed that for periodicity the score increases as the chance of deleting a host from a batch rises. This is the expected outcome since deleting one host will decrease the periodicity of that host in the dataset. Furthermore, the inconsistency score also increases but with a lower magnitude as we change only the presence of a host in a batch without modifying the connection properties from other batches.

5.3 Evolving Behavior

The evolving behavior characteristic assesses changes in interactions between network hosts. This evaluation is crucial for understanding how network interactions evolve, indicating emerging threats or shifts in network dynamics. This character is described by the significant change pattern of host pairs interaction [5].

Scenario	Aggregate %	Content (+) %	Interaction (+) %
Original	0.576460	0.946658	0.206261
3 hosts	0.5**80692**	0.9**59457**	0.20**1927**

Table 3. Evolving behavior scores for altering 3% of the total hosts

This change in interaction is illustrated in Table 3 as the score decreases when 3 hosts from the dataset have their connections altered. This is as expected since a part of the connections of these hosts is deleted resulting in a decrease of the interaction between these hosts and all the others. Thus, the overall interaction measured in the dataset decreases as well.

5.4 Adversarial Behavior

The evaluation of adversarial behavior characteristics involves analyzing the impact of different mutation scenarios on the network traffic. The provided data includes scenarios with different frequency modifications and policy violations. The metrics used for evaluation are the change in attack source percentage, the change in attack target percentage, and the change in attack vector percentage.

Scenario	Aggregate %	Target %	Vector %
Original	1.493207	4.479622	1.695968
2 frequency	**0.102746**	**0.308238**	**4.520193**
policy violation	1.4**06628**	4.**219884**	1.**336379**

Table 4. Adversarial behavior scores for two different types of alteration

The evaluation of these scenarios reveals that different mutation strategies can influence adversarial behavior characteristics in network traffic. The baseline scenario exhibits high variability in attack targets and moderate variability in attack vectors, with no change in attack source. The 2 Frequency scenario, demonstrates a significant reduction in overall variability, more consistent attack targets, and increased variability in attack vectors due to frequent changes in attack methods. The Policy Violation scenario shows a slight reduction in overall variability, marginally more consistent attack targets, and a minor reduction in the variability of attack methods. These findings suggest that while various mutation strategies can affect adversarial behavior characteristics, the specific choice of strategy will depend on the desired outcome—whether to target overall adversarial behavior variability, consistency in attack targets, or variability in attack methods.

5.5 Context Dependency

The evaluation of context dependency in network traffic involves assessing how external and internal factors influence the classification decisions of a Network Intrusion Detection System (NIDS) [5]. Context dependency is critical in understanding the extent to which external information and previous packet evaluations impact the accuracy of NIDS. This is quantified using two primary metrics: the influence of external factors and the influence of internal states.

Internal States refer to the influence of previous packet evaluations on the classification of subsequent packets. NIDS often uses state machines and persistent variables to track the context over time, which can affect the classification of new packets based on prior observations.

$$\text{Context}_{\text{internal}} = \frac{\text{no. of state-dependent events}}{\text{total no. of events}}$$

External factors measure the influence of information unobserved or unconsidered by the NIDS on the true classification of packets. This includes external sources of information that human analysts might use to make final classification decisions, such as experiences, unstructured data, and other resources not directly accessible to machines.

$$\text{Context}_{\text{external}} = \frac{\text{(no. of positives) - (no. of true positives)}}{\text{no. of positives}}$$

Scenario	Aggregate %	Internal State %
Original	0.0024301	0.0048602
+0.2	0.002**9154**	0.00**58309**

Table 5. Context dependency scores

The evaluation of context dependency characteristics involves calculating the influence of external factors and internal states on the classification decisions made by NIDS [5]. The provided data in Table 5 shows that increasing internal state dependency slightly results in a slight increase in both aggregate and internal state percentages. While the tool successfully computed the impact of internal states, it indicated no influence from external factors in the original PCAP file. Although this might require some modification to account for external factors more accurately, it is reasonable to focus on the internal states for now. These calculations provide insights into the dependency of NIDS on context, helping to identify potential areas for improvement in handling contextual information.

6 Future Work

This section presents possible improvement points that could be implemented to enhance our tool. The main challenge we currently face is related to the running time of the processors when big changes in the dataset are required. This is something that happens when the parameters of a processor have high values, meaning more data needs to be altered. To address this issue, we explored different types of data structures that could be used to speed up the running time of our tool. One example is switching from using a *list* datatype to store all the flows that need to be altered into a *set* datatype which is much more efficiently handled. This change only decreased the running time of our tool almost by half.

Moreover, another potential improvement for our tool would be regarding how the modifications in the PCAP file are reflected in the adjustment of ground truth. Currently, we are modifying the ground truth database after the entire PCAP file was altered. This means, that we have to wait for the processing of the network connections to be done before we can start changing the ground truth files. Thus, one could parallelize these two processes such that the overall running time of both altering the PCAP and the ground truth would lower. Additionally, we need to consider further validation of our tool on a larger variety of datasets. As the CICIDS2017 dataset was used for the development and validation of our tool, the next step would be altering different datasets to check the reliability of our implementation.

Besides the previous, there are a few features that could be implemented for easier usage of our tool. In its current state, our tool can be used by running different commands on the machine's terminal. To this extent, there is the possibility of creating a user interface to simply the usage of the tool. This UI can serve both for running the alteration as well as running the metrics tool. Furthermore, it could also extend the current functionalities by processing metrics files to output a visual representation of the score changes of each characteristic.

7 Conclusion

In this project, we developed a tool to manipulate the characteristics of network traffic captures to facilitate robust testing of Machine Learning-based Intrusion Detection Systems (ML-based IDS). The primary goal was to ensure data realism while maintaining the accuracy of the ground truth, enabling a comprehensive evaluation of IDS performance under various conditions. By addressing key characteristics such as imbalance, context dependency, unpredictable behavior, evolving behavior, and adversarial behavior, the tool provides a systematic approach to altering network traffic and adjusting corresponding ground truth labels.

The CICIDS2017 dataset was chosen for its realistic simulation of network traffic and comprehensive labeling, making it an ideal foundation for our experiments. The evaluation using Suricata and Zeek demonstrated the effectiveness of the tool in generating realistic traffic variations and provided valuable insights into the strengths and weaknesses of different IDS models.

The findings from this research contribute to the broader goal of improving IDS robustness and adaptability in real-world scenarios. By systematically manipulating data characteristics and observing the impact on IDS performance, researchers can identify areas for enhancement and develop more resilient detection models. Future work could focus on optimizing the tool's performance and extending its validation to additional datasets. This would further verify the methodology and ensure its applicability across various network environments.

In conclusion, this project underscores the importance of realistic data manipulation in testing and refining ML-based IDS, ultimately contributing to more effective network security measures in an increasingly digital and interconnected world.

References

1. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017
2. The Open Information Security Foundation, Suricata, https://suricata.io/
3. The MITRE Corporation, MITRE ATT&CK, https://attack.mitre.org/
4. The Zeek Project, https://zeek.org/
5. N.J.J. Verbeek, Quantifying Network Characteristics for Intrusion Detection using Passive Observables, TU/e [Unpublished master's thesis], March 2024