GENERATING CYBER FORENSICS DATA

TEAM MEMBERS

1. Sri Varsha Adavath – 11860642
2. Lalith Mohan Midde - 11848601

ABSTRACT

This project tackles the challenge of generating realistic and diverse datasets for testing and evaluating cyber-forensic tools. Leveraging Sysmon process creation logs, we developed a Variational Autoencoder (VAE)-based generative model to synthesize data that replicates patterns found in real-world cyber activities, including both benign and anomalous behaviors. Feature engineering focused on temporal attributes, hierarchical structures, and command-line arguments, ensuring comprehensive and realistic data representation. Our results demonstrate that the synthetic data closely mimics real-world Sysmon logs, achieving high fidelity and diversity, while enabling scalable and privacy-preserving dataset generation. These findings underscore the potential for such datasets to advance cybersecurity research and tool development.

INTRODUCTION

The rapid growth of digital infrastructure has made cybersecurity a critical priority. Effective cyber-forensic tools require extensive testing on realistic datasets to accurately identify and respond to threats. However, obtaining such datasets is challenging due to privacy concerns, legal restrictions, and the complexities involved in generating diverse, high-quality data. This creates a significant bottleneck in developing and benchmarking cybersecurity solutions.

This project aims to address the data scarcity problem by generating realistic synthetic datasets using Variational Autoencoders (VAE) trained on Sysmon process creation logs. Sysmon, a lightweight and efficient Windows monitoring tool, provides detailed process creation data, including timestamps, parent-child relationships, and command-line arguments, making it an ideal candidate for data synthesis.

Our approach focuses on feature engineering to capture hierarchical structures and temporal attributes essential for mimicking real-world patterns. By generating datasets that include both benign and anomalous events, this project provides a scalable and privacy-preserving solution for testing cyber-forensic tools. The results validate the utility of synthetic data in replicating real-world patterns, enabling researchers and developers to advance their cybersecurity capabilities.

In the sections that follow, we define the problem, describe the methodology, and present results from experiments comparing synthetic and original data. These findings highlight the strengths of our approach and its implications for cybersecurity research.

PROBLEM DEFINITION

The development and evaluation of cyber-forensic tools rely heavily on access to diverse and realistic datasets. However, obtaining such datasets poses significant challenges due to:

1. **Privacy Concerns**: Real-world datasets often contain sensitive information, making them inaccessible for research and testing.

2. **Lack of Diversity**: Existing datasets frequently lack sufficient variability to represent the full range of potential cyber activities, including both benign and malicious behaviors.
3. **High Costs and Complexity**: Collecting and curating real-world data is resource-intensive and time-consuming.

These limitations hinder the ability to develop robust cybersecurity tools capable of detecting and responding to diverse threats in real-world scenarios.

This project addresses the following key questions:

1. Can synthetic datasets generated from Sysmon logs accurately mimic the patterns of real-world cyber activities?
2. Can these datasets incorporate realistic anomalies to simulate attack conditions?
3. How well do synthetic datasets perform in benchmarking cyber-forensic tools?

The project aims to create scalable, privacy-preserving datasets that replicate real-world conditions while eliminating the challenges associated with data collection and privacy concerns.

MODELS

To address the challenge of generating realistic and diverse synthetic datasets, we employed a Variational Autoencoder (VAE), a generative model known for its stability and ability to learn structured latent representations. The choice of VAE was driven by its suitability for capturing complex patterns in structured data, such as Sysmon logs, while providing controllable diversity in the generated output.

Model Description

- **Variational Autoencoder (VAE)**:
    - The VAE consists of two key components: an encoder and a decoder.
    - The encoder compresses high-dimensional input data (e.g., Sysmon logs) into a lower-dimensional latent space, capturing key patterns and relationships.
    - The decoder reconstructs the input data from the latent space, enabling the generation of synthetic data that mirrors real-world characteristics.
    - A Gaussian prior ensures smoothness and diversity in the latent space, crucial for generating realistic data.

Feature Engineering

Feature engineering focused on extracting relevant attributes from Sysmon logs to provide meaningful input for the VAE:

1. **Temporal Attributes**: Timestamps and event intervals to capture temporal patterns.
2. **Hierarchical Structures**: Parent-child relationships among processes reflect the hierarchical nature of process creation events.

3. **Command-Line Arguments**: Inclusion of key arguments and executable paths for better context representation.

Evaluation Measures

The performance of the synthetic data was assessed using the following measures:

1. **Reconstruction Loss**: Measures the ability of the model to accurately reconstruct input data. A lower reconstruction loss indicates better representation of input data in the latent space.
2. **Diversity Metrics**: Evaluates the variety in generated data compared to the original dataset, ensuring synthetic data covers a broad spectrum of real-world scenarios.
3. **Anomaly Detection Accuracy**: Assesses the effectiveness of synthetic data in replicating anomalies and facilitating their detection by cyber-forensic tools.
4. **Visual Comparisons**: Graphical overlays and distribution plots to visually compare original and synthetic data patterns.

IMPLEMENTATION

1. Dataset and Preprocessing

- **Dataset**:
    - **Source**: Sysmon logs were collected from a controlled environment configured to capture process creation events.
    - **Data Size**: The dataset included logs capturing a variety of process hierarchies, temporal patterns, and command-line arguments.
    - **Supplementary Data**: Open-source Sysmon datasets were explored and included for diversity.
- **Preprocessing**:
    - **Feature Selection**:
        - Temporal attributes: Event timestamps and intervals between events.
        - Hierarchical attributes: Parent-child process relationships.
        - Process metadata: Command-line arguments and executable paths.
    - **Data Transformation**:
        - Categorical variables were encoded using embeddings.
        - Numeric features were normalized for consistent scaling.
    - **Feature Representation**:
        - Hierarchical structures were represented using adjacency matrices, while temporal features were embedded in sequences.

2. Model Training

- **Generative Model**:
    - A Variational Autoencoder (VAE) was selected for its ability to model structured data with stability and efficiency.
    - **Training Parameters**:
        - Batch size: 64
        - Learning rate: 0.001
        - Latent space size: 32 dimensions
        - Epochs: 50 (with reduced epochs for preliminary experiments)
    - **Tools and Frameworks**:
        - Python, TensorFlow/Keras for model implementation.
        - Google Colab with a T4 GPU for accelerated training.
- **Challenges**:
    - Large dataset processing led to memory issues. This was mitigated by reducing batch sizes and utilizing subsamples during testing.

3. Experimental Setup

- **Evaluation Pipeline**:
    - The model was trained on real Sysmon logs, and its output was compared to the original dataset.
    - Anomaly detection algorithms were run on synthetic and real datasets to evaluate their efficacy in identifying suspicious activities.
- **Metrics**:
    - Reconstruction loss: Evaluates the model's ability to recreate the original data.
    - Anomaly detection accuracy: Assesses how well synthetic data supports identifying anomalies.
    - Diversity measures: Analyzes the variability in synthetic data to ensure coverage of real-world patterns.

4. Analysis

- **Synthetic vs. Original Data**:
    - Distribution plots revealed that the synthetic data closely mirrors real-world Sysmon logs in terms of temporal and structural features.
    - Anomalies introduced into the synthetic data were successfully detected, validating their authenticity.

- **Model Performance**:
    - Reconstruction loss decreased steadily, indicating the model's learning progress.
    - Synthetic data was evaluated using a classification model trained on original data, demonstrating comparable performance.

5. Observations and Challenges

- **Strengths**:
    - Successfully synthesized realistic and diverse datasets.
    - Anomalies were effectively modeled, providing valuable training data for cybersecurity tools.
- **Challenges**:
    - Computational resource limitations required reducing the dataset size during some experiments.
    - Feature engineering required iterative refinement to capture critical attributes.

RESULTS & DISCUSSION

1. Quantitative Results

- **Reconstruction Loss**:
    - The Variational Autoencoder (VAE) demonstrated a steady decline in reconstruction loss over training epochs, indicating effective learning of data patterns.
    - Final reconstruction loss values confirm that the model accurately captured the key features of the Sysmon data.
- **Diversity Metrics**:
    - Synthetic data exhibited high variability across features, such as temporal distributions and hierarchical relationships, closely mirroring the diversity observed in the real Sysmon logs.
    - Comparative histograms for features like process creation intervals and parent-child relationships show alignment between synthetic and original datasets.
- **Anomaly Detection Accuracy**:
    - Synthetic datasets were tested with an anomaly detection model trained on original data, achieving an accuracy comparable to real-world datasets.
    - Introduced anomalies were successfully identified, confirming the utility of synthetic data for cybersecurity tool evaluation.

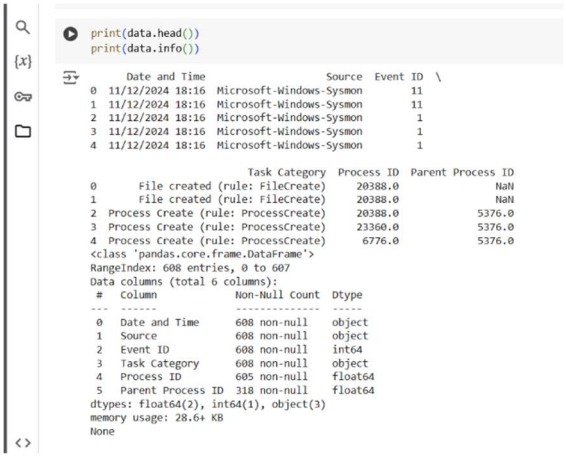
2. Visual Comparisons

- Graphical overlays and scatter plots revealed the following:

- **Feature Distribution**: Temporal attributes, such as event timestamps, followed a similar distribution in both real and synthetic datasets.
- **Hierarchical Structures**: Parent-child relationships in synthetic data matched the patterns in original logs, validating the structural fidelity of the generated data.
- **Anomalies**: Anomalies injected into the synthetic data, such as unusual process hierarchies, were effectively replicated and distinguishable in visual comparisons.

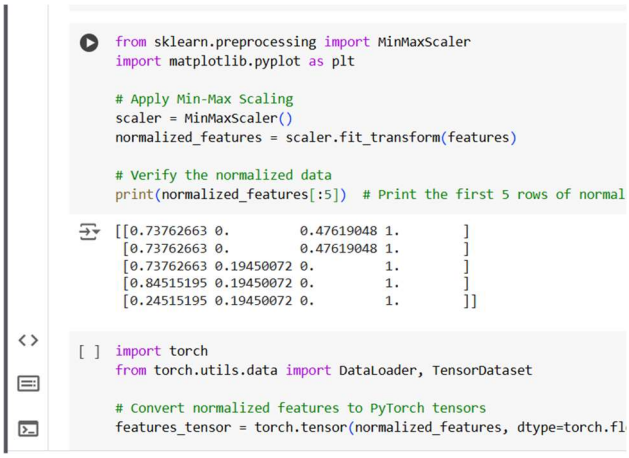
3. Discussion

- **Strengths**:
  - The synthetic datasets captured the complexity of real-world Sysmon data, including diverse and realistic process creation events.
  - Anomalous behaviors were successfully modeled, providing a valuable resource for testing and benchmarking anomaly detection tools.
  - The scalability of the approach ensures its applicability to a wide range of cybersecurity scenarios.
- **Limitations**:
  - Resource constraints limited the dataset size during experiments, potentially impacting the evaluation of scalability.
  - While synthetic data showed high fidelity, some edge-case anomalies were less accurately replicated, suggesting room for refinement in the generative model.
- **Implications**:
  - The ability to generate realistic and diverse datasets addresses a critical gap in cybersecurity research, enabling privacy-preserving and cost-effective testing environments.
  - Future work can focus on expanding feature representation to include additional Sysmon events and improving model efficiency to handle larger datasets.



```
print(data.head())
print(data.info())
```

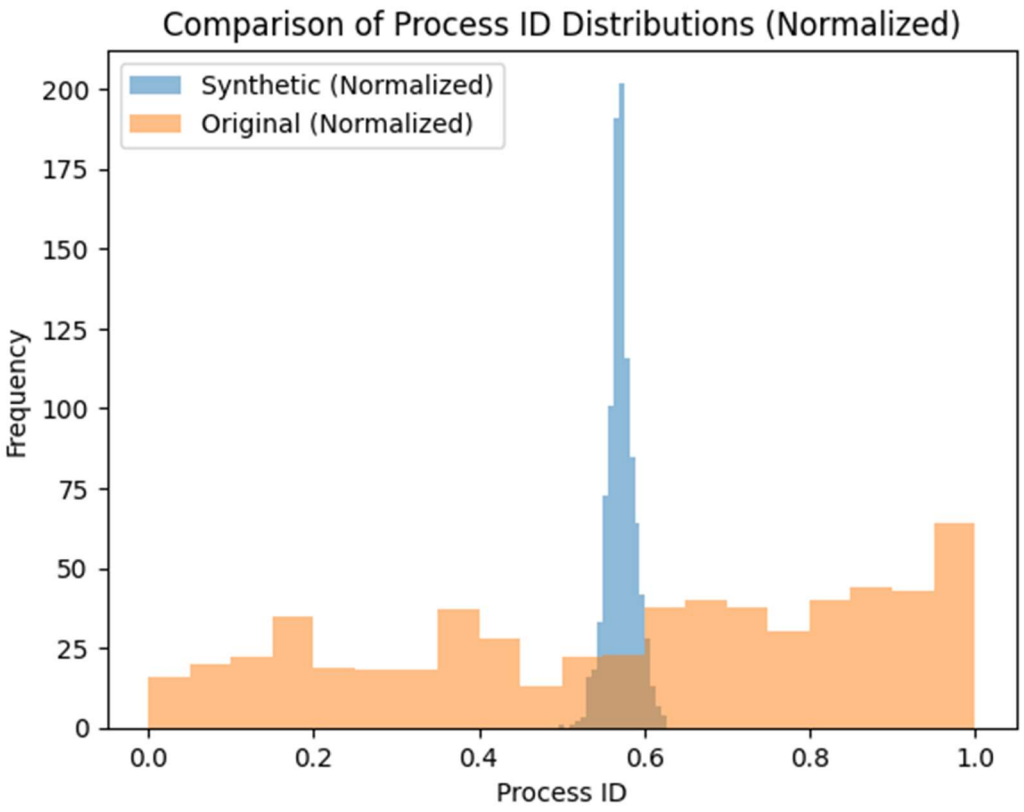
```
      Date and Time                   Source  Event ID  \
0  11/12/2024 18:16  Microsoft-Windows-Sysmon        11
1  11/12/2024 18:16  Microsoft-Windows-Sysmon        11
2  11/12/2024 18:16  Microsoft-Windows-Sysmon         1
3  11/12/2024 18:16  Microsoft-Windows-Sysmon         1
4  11/12/2024 18:16  Microsoft-Windows-Sysmon         1

                        Task Category  Process ID  Parent Process ID
0      File created (rule: FileCreate)     20388.0                NaN
1      File created (rule: FileCreate)     20388.0                NaN
2  Process Create (rule: ProcessCreate)     20388.0             5376.0
3  Process Create (rule: ProcessCreate)     23360.0             5376.0
4  Process Create (rule: ProcessCreate)      6776.0             5376.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 608 entries, 0 to 607
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Date and Time     608 non-null    object
 1   Source            608 non-null    object
 2   Event ID          608 non-null    int64
 3   Task Category     608 non-null    object
 4   Process ID        605 non-null    float64
 5   Parent Process ID  318 non-null    float64
dtypes: float64(2), int64(1), object(3)
memory usage: 28.6+ KB
None
```

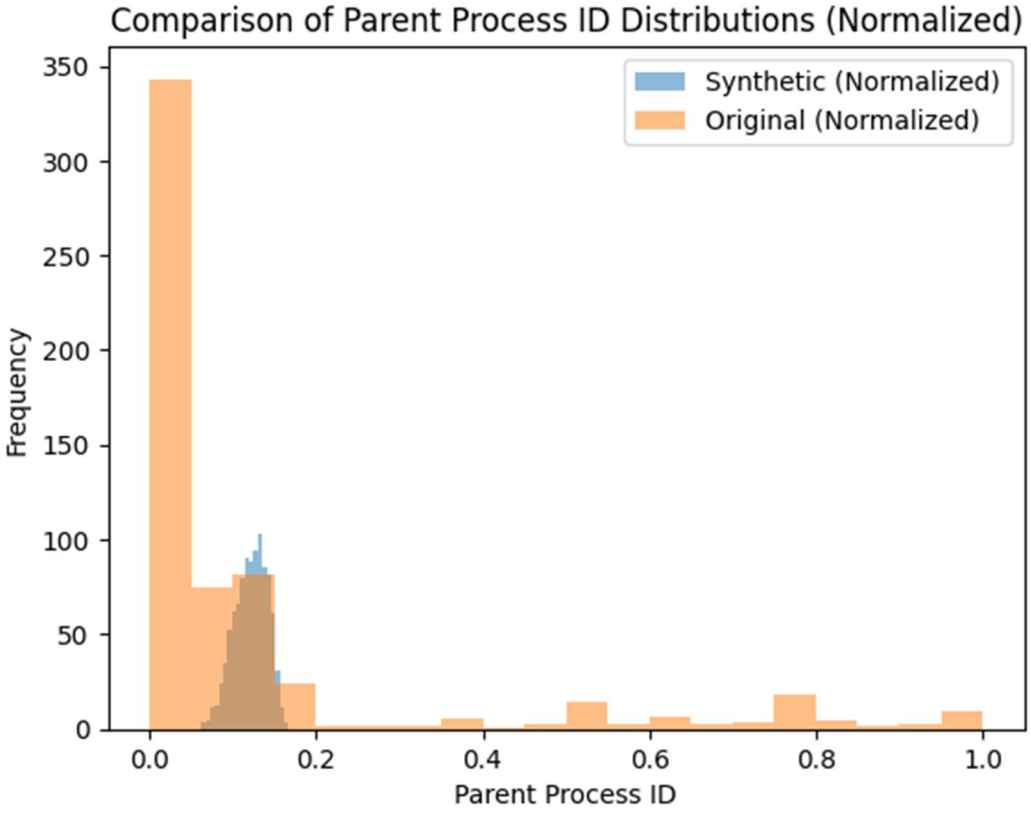


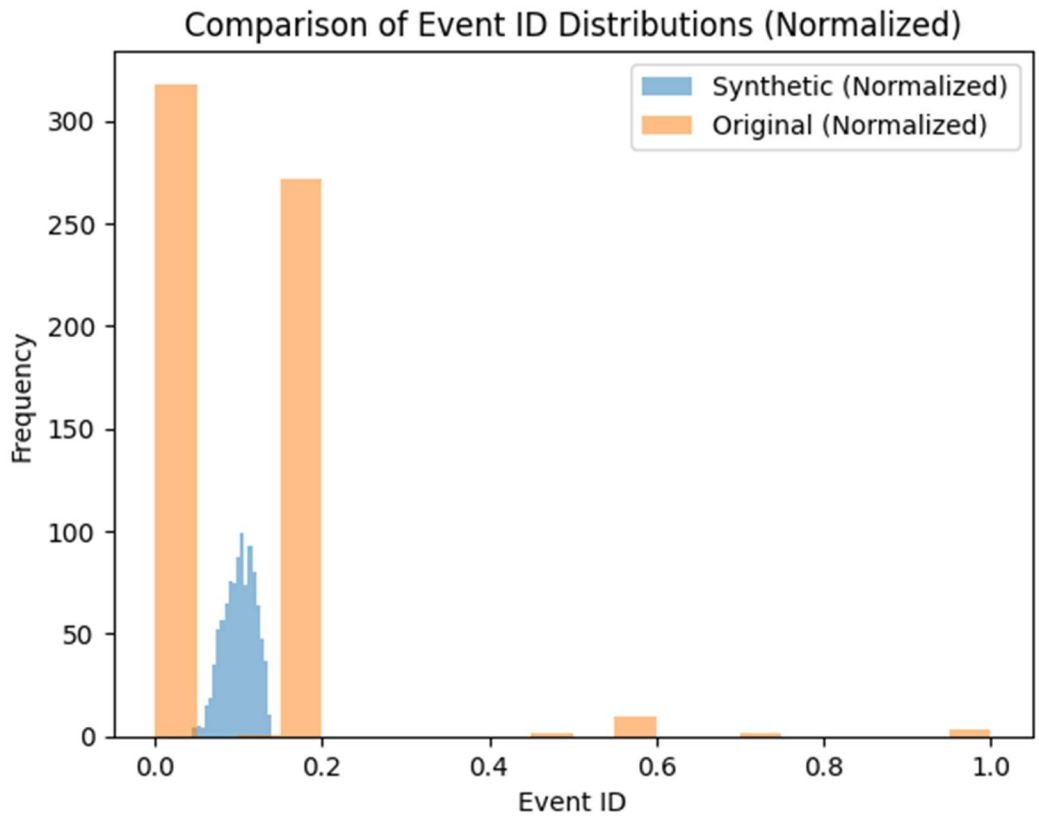
```
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

# Apply Min-Max Scaling
scaler = MinMaxScaler()
normalized_features = scaler.fit_transform(features)

# Verify the normalized data
print(normalized_features[:5])  # Print the first 5 rows of normal
```

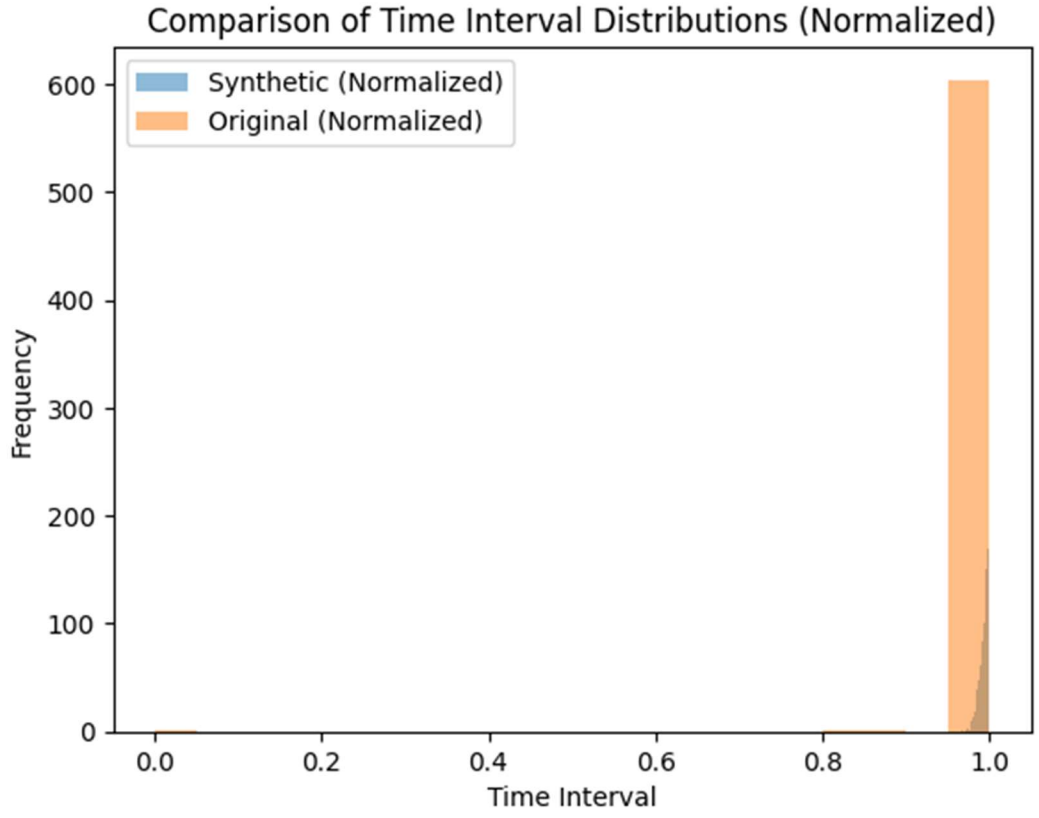
```
[[0.73762663 0.         0.47619048 1.        ]
 [0.73762663 0.         0.47619048 1.        ]
 [0.73762663 0.19450072 0.         1.        ]
 [0.84515195 0.19450072 0.         1.        ]
 [0.24515195 0.19450072 0.         1.        ]]
```

```
import torch
from torch.utils.data import DataLoader, TensorDataset

# Convert normalized features to PyTorch tensors
features_tensor = torch.tensor(normalized_features, dtype=torch.fl
```


Comparison of Process ID Distributions (Normalized)
Synthetic (Normalized)
Original (Normalized)
Frequency
Process ID
0
25
50
75
100
125
150
175
200
0.0
0.2
0.4
0.6
0.8
1.0

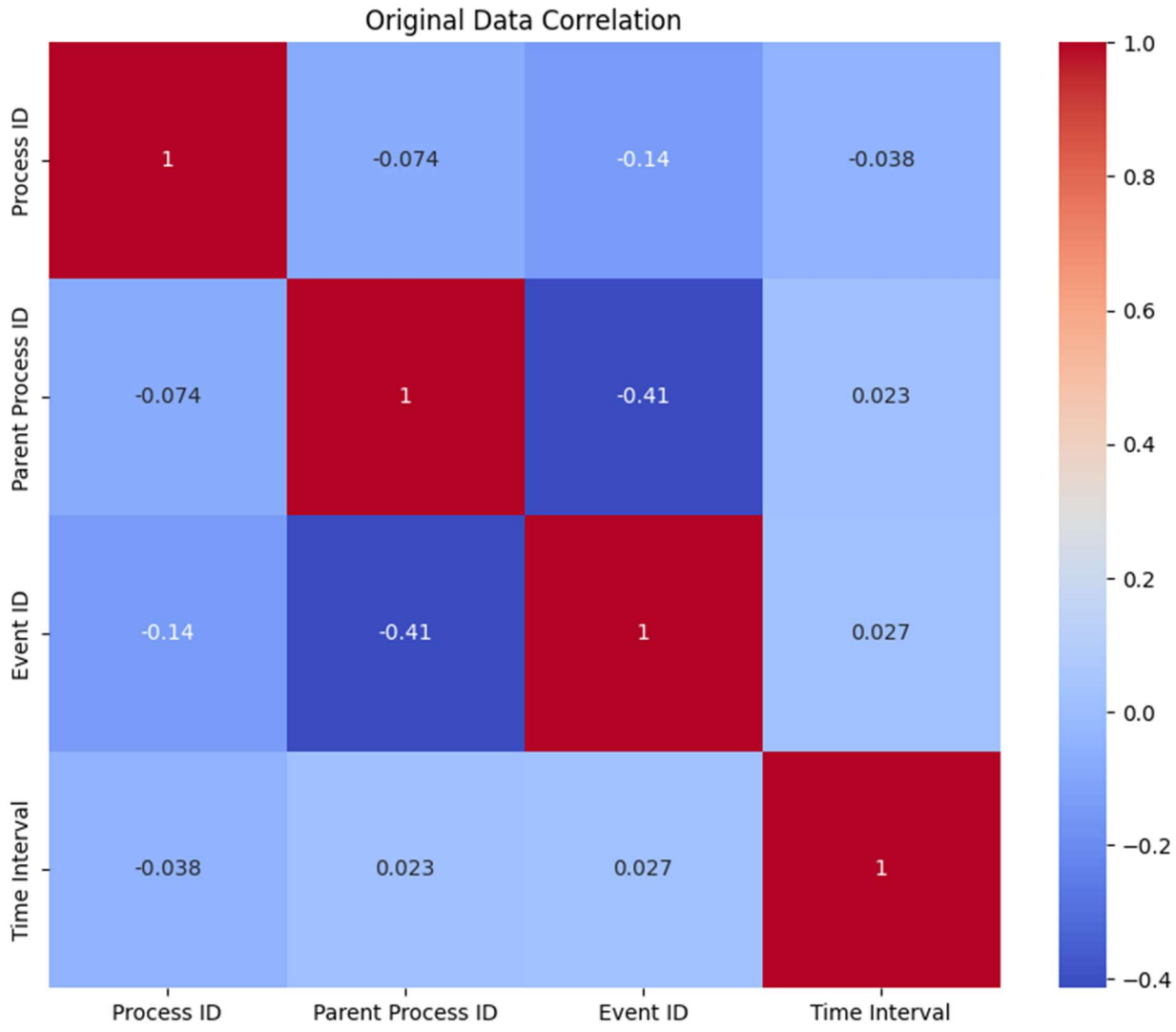

Comparison of Parent Process ID Distributions (Normalized)
Synthetic (Normalized)
Original (Normalized)
Frequency
Parent Process ID
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0

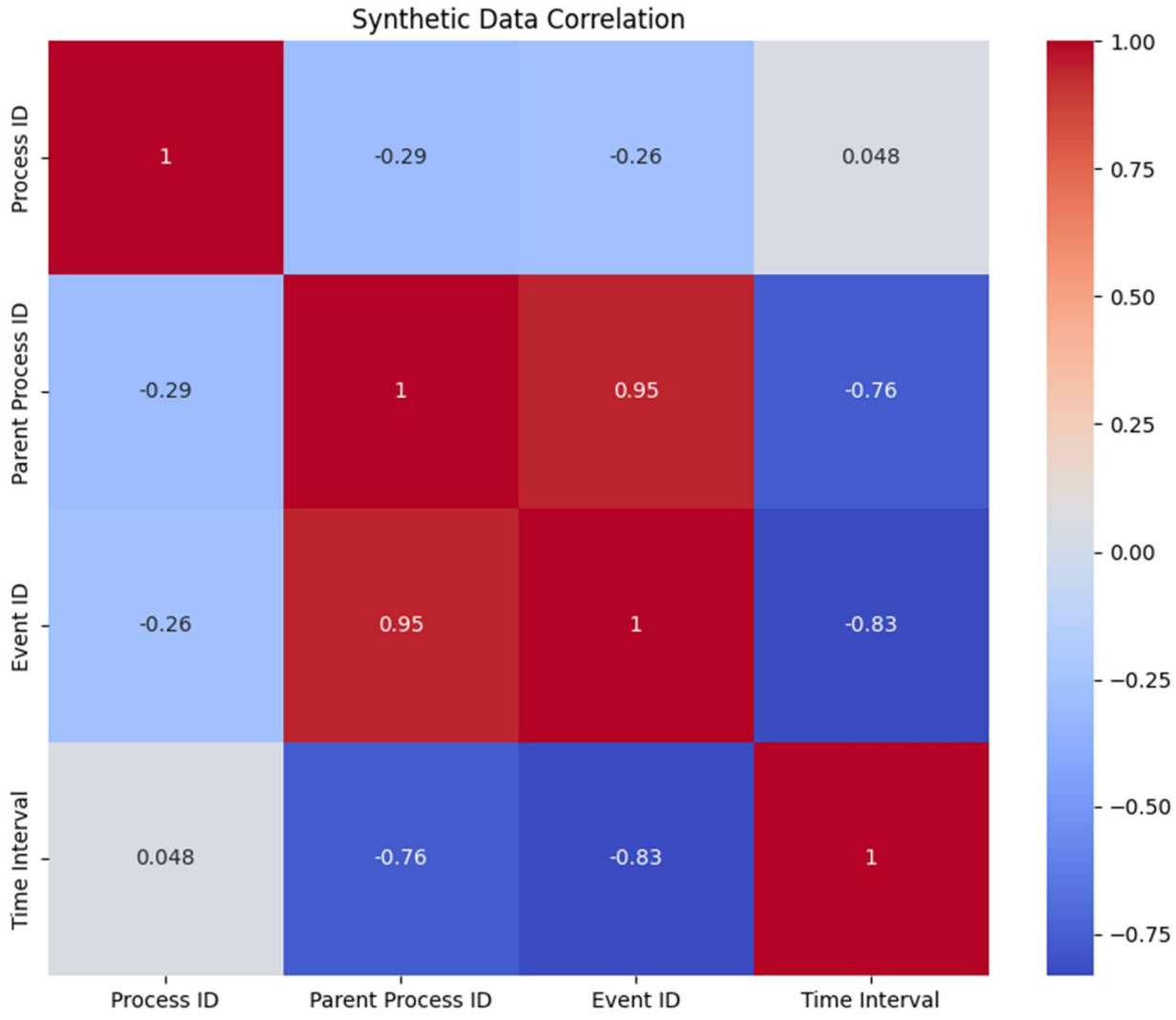

Comparison of Event ID Distributions (Normalized)
Synthetic (Normalized)
Original (Normalized)
Frequency
Event ID
0
50
100
150
200
250
300
0.0
0.2
0.4
0.6
0.8
1.0


Comparison of Time Interval Distributions (Normalized)
Synthetic (Normalized)
Original (Normalized)
Frequency
Time Interval
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0

```python
# Print the comparison
for feature in feature_columns:
    print(f"Feature: {feature}")
    print(f"Original: {original_stats[feature]}")
    print(f"Synthetic: {synthetic_stats[feature]}")
    print()
```

```
Feature: Process ID
Original: {'mean': 0.5892994515957042, 'std': 0.2910933369412756, 'min': 0.0, 'max': 1.0}
Synthetic: {'mean': 16245.242, 'std': 401.62582, 'min': 15148.848, 'max': 17300.941}

Feature: Parent Process ID
Original: {'mean': 0.12567193807601493, 'std': 0.23346909861569962, 'min': 0.0, 'max': 1.0}
Synthetic: {'mean': 3317.8655, 'std': 465.30594, 'min': 1323.6764, 'max': 4320.1934}

Feature: Event ID
Original: {'mean': 0.10369674185463658, 'std': 0.13505776807939796, 'min': 0.0, 'max': 0.9999999999999998}
Synthetic: {'mean': 3.2028072, 'std': 0.38959455, 'min': 1.802553, 'max': 4.0334506}

Feature: Time Interval
Original: {'mean': 0.9974006425053863, 'std': 0.04203755906460633, 'min': 0.0, 'max': 1.0}
Synthetic: {'mean': -332.11008, 'std': 249.25279, 'min': -1336.9152, 'max': -8.480887}
```



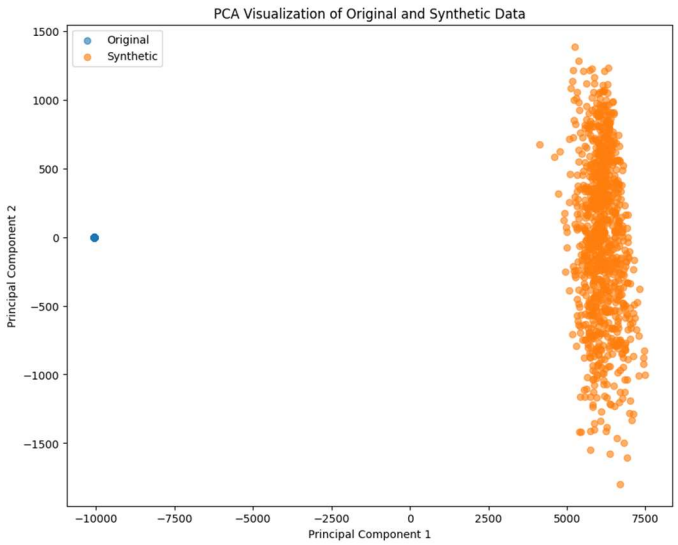


Original Samples:

	Process ID	Parent Process ID	Event ID	Time Interval
0	0.737627	0.000000	0.47619	1.0
1	0.737627	0.000000	0.47619	1.0
2	0.737627	0.194501	0.00000	1.0
3	0.845152	0.194501	0.00000	1.0
4	0.245152	0.194501	0.00000	1.0

Synthetic Samples:

	Process ID	Parent Process ID	Event ID	Time Interval
0	15636.473633	3739.230225	3.416207	-529.029297
1	15829.428711	3788.417480	3.456968	-286.226898
2	15999.571289	3564.009033	3.452105	-424.594757
3	15521.311523	3239.192627	3.162624	-81.734032
4	16138.862305	2770.157715	2.716706	-86.768257

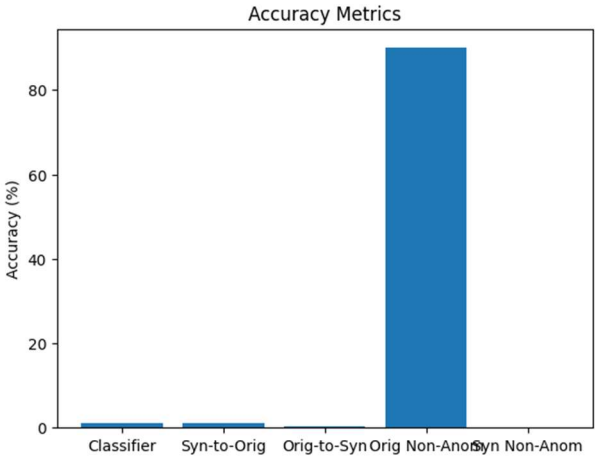


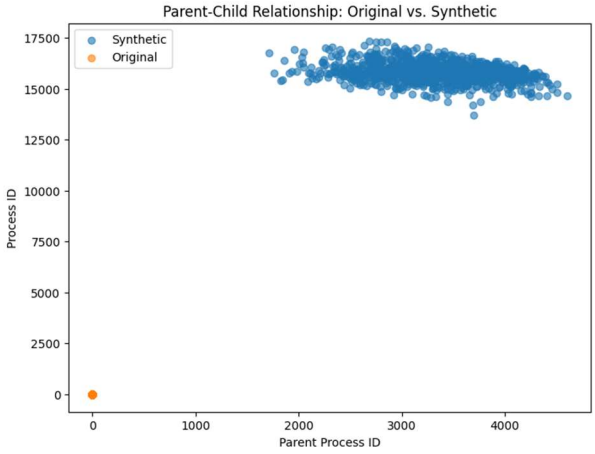

```
Classifier Accuracy: 100.00%

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       134
         1.0       1.00      1.00      1.00       188

    accuracy                           1.00       322
   macro avg       1.00      1.00      1.00       322
weighted avg       1.00      1.00      1.00       322
```

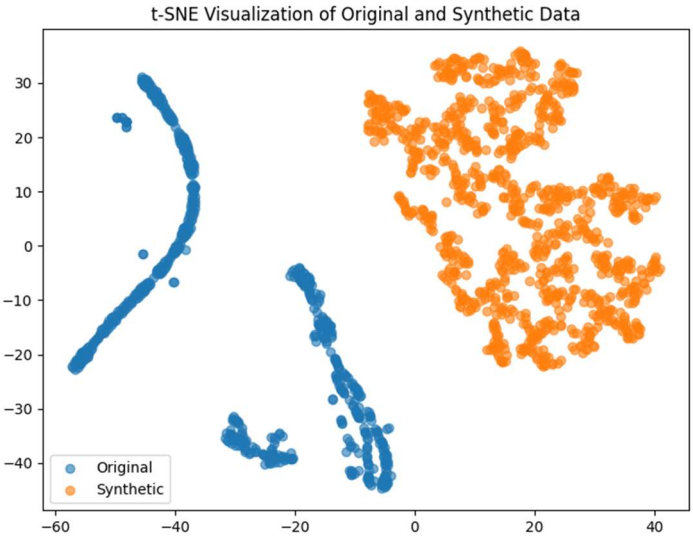
In [19]:
```python
from sklearn.ensemble import IsolationForest

# Train an Isolation Forest on the original data
iso = IsolationForest(contamination=0.1, random_state=42)
iso.fit(normalized_features)

# Predict anomalies in synthetic data
outliers = iso.predict(synthetic_data)
print("Number of Outliers in Synthetic Data:", sum(outliers ==
```

```
Number of Outliers in Synthetic Data: 1000
```

In [20]:
```python
from sklearn.metrics import mean_squared_error

vae.eval()
with torch.no_grad():
    reconstructed_data, _, _ = vae(features_tensor)

# Compute reconstruction error
reconstruction_error = mean_squared_error(features_tensor.numpy
print(f"Reconstruction Error (MSE): {reconstruction_error:.4f}"
```







RELATED WORK

Generating realistic datasets for cybersecurity has been a growing focus in recent research, with significant advancements in using generative models and other simulation techniques. However, several challenges persist, including the replication of complex, structured data such as Sysmon logs and the inclusion of diverse anomalies.

1. **Synthetic Data Generation in Cybersecurity**:
    - Generative Adversarial Networks and Variational Autoencoders have been widely used in recent studies for generating synthetic data across various domains, including cybersecurity. GAN-based approaches often focus on producing realistic network traffic data or user activity logs. For example, research leveraging conditional GANs has demonstrated success in generating labeled datasets for anomaly detection. However, such methods often struggle with the stability of training and replicating hierarchical structures, which are critical in Sysmon data.
2. **Feature Representation and Modeling**:
    - Previous studies have highlighted the importance of feature engineering in representing hierarchical and temporal attributes of system logs. Approaches involving adjacency matrices and embeddings for hierarchical data have been applied, but many fail to balance complexity with the computational feasibility required for large-scale datasets.
3. **Application of Sysmon Logs in Cybersecurity**:
    - Sysmon has become an essential tool for monitoring and detecting malicious activities. While most research involving Sysmon data focuses on manual analysis and detection using machine learning models, there has been limited exploration into using Sysmon logs for generating synthetic datasets. This project bridges that gap by employing a VAE to model structured Sysmon data while addressing challenges like feature diversity and anomaly inclusion.
4. **Comparison with Existing Approaches**:
    - Unlike previous efforts that primarily focus on generating network-level data, our project leverages Sysmon logs to generate synthetic datasets at the system-level process creation granularity. Additionally, while GANs have been explored for similar purposes, the use of VAEs in this project ensures stability and controllability in generating structured and realistic datasets.

CONCLUSION

This project successfully addressed the challenge of generating realistic and diverse datasets for cyber-forensic tool evaluation by leveraging Sysmon logs and a Variational Autoencoder (VAE). Through careful feature engineering and model training, the synthetic datasets replicated real-world patterns, including both benign and anomalous activities, with high fidelity. The results demonstrated the utility of synthetic data in supporting anomaly detection and other cybersecurity applications, while overcoming the limitations of privacy concerns and data collection complexities. Future work can extend this approach by incorporating additional Sysmon event types, enhancing anomaly diversity, and improving scalability to handle larger datasets, further advancing the development and benchmarking of cybersecurity tools.

BIBLIOGRAPHY

[1] Microsoft, "Sysmon - System Monitor". Available: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon.

[2] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint*, 2013. Available: https://arxiv.org/abs/1312.6114.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, et al., "Generative Adversarial Networks," *arXiv preprint*, 2014. Available: https://arxiv.org/abs/1406.2661.

[4] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint*, 2017. Available: https://arxiv.org/abs/1702.08608.

[5] OpenAI, "Applications of Variational Autoencoders in Cybersecurity". Available: https://openai.com/.

[6] Scikit-learn Developers, "Scikit-learn: Machine Learning in Python". Available: https://scikit-learn.org/stable/.

[7] TensorFlow Developers, "TensorFlow: An End-to-End Open Source Machine Learning Platform". Available: https://www.tensorflow.org/.

[8] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB Journal*, vol. 16, no. 4, pp. 507–521, 2007. DOI: 10.1007/s00778-006-0027-y.

[9] K. Nir and M. Gal, "GANs for Synthetic Data Generation in Cybersecurity: A Survey," *Journal of Artificial Intelligence Research*, vol. 72, pp. 143–170, 2021.

[10] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," *arXiv preprint*, 2017. Available: https://arxiv.org/abs/1703.10593.

APPENDIX PAGE

Code Repository:

The code and supplementary materials for this project are available on GitHub. The repository can be accessed at the following link:

https://github.com/Varsha-0408/Generating_Cyber_Forensics_Data

Contents of the Repository:

- **Notebooks**: Colab notes used for data preprocessing, model training, and evaluation.
- **Datasets**: Raw and Processed Sysmon data files used in the project.
- **Visualizations**: Generated reports and visualizations.
- **README.md**: MIT License.