AI-Powered Behavior-Based Malware Detection Using Advanced Temporal and Process-State Features: A Robust Explainable Framework.

NHM Hassan Imam Chowdhury[1*], Md. Ezharul Islam[2], Md. Sunjid Hasan[3], Abdullah Al Mehedi[4]

[1, 3, 4]Department of ICT, Bangladesh University of Professionals, Dhaka, Bangladesh

[2] Professor, Department of CSE, Jahangirnagar University, Dhaka Bangladesh

[*]Corresponding Author:

NHM Hassan Imam Chowdhury

Email: rafichowdhury60@gmail.com

Orchid ID: https://orcid.org/0009-0007-2913-9611

Abstract

Malware detection remains a critical challenge in modern cybersecurity due to the rapid evolution of attack techniques and the proliferation of adversarial threats. This study introduces a robust, explainable framework for malware detection that leverages advanced temporal and process-state features. Using VirusTotal, a large dataset of practical metrics, including system, memory, and process metrics, was created. Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) architectures were implemented and frequently tested to model this sequential behavioral data. GRU outperformed the others regarding robustness and performance, with 99.92% accuracy on original data and 92.61% on adversarial data after retraining with adversarial examples. It also highlights the importance of interpretability by incorporating SHAP (Shapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) to provide global and local feature importances. It found that key system characteristics such as maj_flt (major faults) and time (user CPU time) were essential for classification, which indicates that behavioral patterns may be more important than static implementation in malware detection. Furthermore, the adversarial robustness testing phase highlighted resilience against such feature perturbations, proving the model's adaptability towards realistic attack scenarios. This framework sets a new benchmark in behavior-based malware detection, offering a reliable and interpretable solution for modern cybersecurity challenges.

Keywords: Malware Detection, Adversarial Robustness, Explainable AI, LSTM, GRU

1. Introduction

Malware's increasing complexity and frequency have emerged as major challenges in the current cybersecurity landscape (Charmet et al., 2022). As such, malware poses a significant danger, continually advancing its ability to exploit system vulnerabilities, circumventing traditional safeguards, and executing malicious activities undetected (Amer et al., 2021). The advances in malware design (polymorphic and zero-day attacks) make the traditional detection mechanism based on signatures ineffective (Galli et al., 2024). One of the big problems with traditional systems is that they rely on signature patterns to identify a threat, making it challenging to follow new or evolving types of malware accurately. The static defenses are not adaptive, so the systems are vulnerable to new adaptive threats, as hackers at each period find new ways by keeping the attack protocols. Behaviour-based detection methods have become the best practice over signature-based methods (Chamola et al., 2023). Instead, these approaches analyze process and system state dynamic behaviors at runtime to produce more elaborate signatures that can be more descriptive of malicious actions. Behavior-based methods rely on understanding how malware behaves with the system's resources, which allows them to identify abnormal results and discover never-seen threats by spotting anomalies. A malicious code would run, e.g., abnormal CPU utilization, memory accessed strangely, or process priorities changed (Mehrban et al., 2023). Proactive defense mechanisms can help keep systems alive and react to these threats in real-time. However, behavior-based detection requires more complex algorithms due to the system behaviors' sequential and temporal nature. Sequential data modeling using machine learning is another possible domain that could help behavior-based malware detection progress. Due to their ability to learn long-term temporal dependencies, Recurrent Neural Networks (RNN) such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) have specific gusto for modeling temporal dependencies in the behaviors of the system (Maniriho et al., 2024). With these models, we can determine very complex behavioral anomalies that indicate the presence of malware in noisy, challenging environments. Nevertheless, the vulnerability of machine learning models to adversarial attacks implies that a strong framework should be built to resist adversarial attacks (Bhaskara et al., 2023). The rapid advancements in artificial intelligence (AI) have significantly transformed the landscape of malware detection. Most traditional methods rely almost solely on static signatures or heuristic analysis, limiting the ability to detect new or polymorphic malware. On the other hand, AI-driven systems—especially those based on ML—bring a more agile and scalable solution, as they can be trained to detect advanced and evolving threats (Ambekar et al., 2024). An efficient emerging approach is behavior-based detection, exploiting AI and other tools to replay deep real-time system activity analyses. AI models can spot deviations from normal system behavior by monitoring system and memory metrics and flag anomalous behavior that could indicate malware (Afifah & Stiawan, 2019). The most significant benefit of AI-based malware detection is that it detects patterns from dynamic system states instead of relying on defined signatures. Detection models benefit from incorporating temporal and process-state features, which can also improve their ability to detect maliciously. Temporal features, like the time series of CPU usage or when system calls are performed, help to capture complex interactions between processes and the system (Guendouz et al., 2023). Process-state features (resource usage (memory, CPU, disk), process priorities, page faults) are key to understanding program operations in benign and malicious scenarios. Longitudinal monitoring of these features enables the detection of even minor deviations from regular, at times more subtle than those visible to the human eye, raising a more accurate identification of novel malware attacks than that offered by AI models trained on static non-functional code (Malhotra, 2021). The novelty of using these time-related and process-state features is that they incorporate signatures of the processes that traditional methods cannot capture. Signature-based techniques rely on existing malware signatures to detect known malware, but behavior-based AI models can detect abnormal system activity patterns learned through training even when the attack is new and known to no one. Organizations can use AI and ML-based defense strategies to ensure their IT infrastructure remains intact even when it faces novel threats from unknown malware. Detecting these new and unknown threats is especially important since malware authors constantly change tactics to avoid detection (Wang et al., 2021). The AI models must be robust enough to withstand adversarial attacks, as with all ML systems (Danilevsky et al., 2020). One such threat is adversarial attacks, which involve small, carefully designed perturbations added to the input data to misdirect the model and make a wrong prediction. Within practice, a malware author can camouflage malware to evade detection systems by slightly changing its code or behavior. Thus, defending AI models from this kind of adversarial attack is pivotal for the trustworthiness and utility of these models when facing the newly emerging threat (Kuppa & Le-Khac, 2020).

This research aims to develop an explainable, behavior-based malware detection model to overcome the limitations of traditional signature-based methods. Temporal characteristics (e.g., CPU time, process timing) and process-state-based metrics (e.g., memory utilization, page faults) will be utilized to describe malware conduct to feed into deep learning models (e.g., LSTM and GRU) used to learn from sequences of data. Through retraining with adversarial data, the study will examine model performance under normal and adversarial conditions, improving resistance to adversarial attacks. SHAP for global feature importance and interpretability, while LIME will explain local predictions, thus providing transparency and trust. The end goal is a strong, scalable, and explainable malware detection framework.

2. Literature Review

In Since traditional defensive methods are being challenged by the growing complexity and sophistication of cyber-attacks, malware detection is a fundamental component of modern cybersecurity (Naseer et al., 2021). Traditional detection techniques are no longer suitable because malware techniques have increased, such as obfuscation, polymorphism, and zero-day attacks (Wang et al., 2021). This section discusses advancements in malware detection techniques with an emphasis on behavior-oriented methodologies, ML and DL applications, adversarial robustness, and the crucial role of XAI in increasing model trustworthiness (Charmet et al., 2022; Chamola et al., 2023).

2.1 Malware Detection Approaches

In traditional malware detection techniques, Signature- and heuristic-based detection are mainly used (Maniriho et al., 2024). This involves matching patterns or parts of this malicious code against a database, also known as signature-based detection, which is known for its speed but ineffective in stopping new or polymorphic malware that modifies itself to avoid detection (Namavar et al., 2020). Heuristic approaches try to derive rules for distinguishing between normal and suspicious behavior, but their performance is often limited due to high false positive rates and flexibility against advanced threats (Ahmed et al., 2022). On the contrary, behavior-based detection, where false behavior is monitored to detect malware, has become a solid alternative, especially in monitoring run-time actions such as system logs, memory access, and energy usage (Alaeiyan et al., 2023). Behavior-based approaches create a defensive posture that prevents obfuscation and new attacks by identifying anomalies, i.e., peak CPU activity, a combination of page faults, or unexpected accesses to a memory region (Finder et al., 2022). Behavior-based methods incorporate machine learning (ML) to automate pattern recognition, where the model can now apply classifiers to identify malware as subtle changes in the behavior of what is malicious and what is benign (Galli et al., 2024).

2.2 AI and Deep Learning in Malware Detection

Artificial intelligence (AI) and deep learning (DL) have revolutionized malware detection by enabling systems to learn complex patterns in system behaviors (Singh & Singh, 2021). Sequential data can be naturally modeled by RNN, LSTM, and GRU (e.g., system logs and temporal features are also a sequence) (Kolosnjaji et al., 2016). They can capture dependencies in the time-series data reflecting malware analysis, such as Memory Utilization, CPU allocation, and process scheduling (Naseer et al., 2021). Previous works indicated that LSTM and GRU are effective for features like page faults, resource counters, and state transitions (Venkatraman et al., 2019). For example, GRU models have performed well on API call sequences, while LSTMs exhibited SOTA results on temporal anomalies (Mpanti et al., 2018). To further enhance precision, the recent work combines deep learning with hybrid methods (Liras et al., 2021), e.g., convolutional layers fused with RNNs, to learn the spatial and temporal patterns in malware binaries. To circumvent these restrictions and extend competencies, behavior-driven AI models use system logs, process behaviors, and temporal metrics (Maniriho et al., 2024).

2.3 Adversarial Testing in AI Models

Adversarial attacks significantly threaten the robustness of AI-based malware detection (Bhaskara et al., 2023). Adversarial (bounded) attacks are a type of attack that generates small intentional perturbations of input data types—for instance, slight modifications to sequences of Application Programming Interface (API) calls or changes in resource usage metrics—that can result in models misclassifying malicious samples as benign (Ambekar et al., 2024). Adversarial robustness is crucial for safeguarding the integrity of AI systems, especially in high-stakes cybersecurity contexts (Afifah & Stiawan, 2019). The extant methods that attempt to counter adversarial threats include adversarial training, a technique that retrains a model on data altered in an adversarial direction to make the model more robust, and defensive distillation that makes a model less sensitive to small perturbations by refining its decision boundaries (Cui et al., 2019). Robustness is also enhanced with hybrid frameworks that use combinations of static, dynamic, and behavior-based features, increasing multiple layers of protection against evasion techniques (Venkatraman et al., 2019). However, many contemporary models do not focus on adversarial robustness, which places detection systems at risk of sophisticated attacks (Bhaskara et al., 2023). Tools like SHAP and LIME can assist in identifying potential adversarial weaknesses by revealing features integral to the model's decision-making process (Charmet et al., 2022).

While there has been much advancement in behavior-based malware detection, there is still a significant research gap, especially in explainability. Most of the proposed solutions have interpretable frameworks (Lundberg et al., 2017), while deep learning models were considered black boxes that cannot justify their predictions, thus questioning trust and usability (Kuppa et al., 2020). SHAP or LIME are tools that have not been leveraged often; consequently, transparency is hard to achieve (Chamola et al., 2023). Also, adversarial robustness has not been addressed broadly, with only a few works covering adversarial training or performance analysis (Singh & Singh, 2021). Likewise, temporal and process-state features, such as millisecond-level timing, state transitions, and memory metrics, are also not being fully leveraged, thus limiting the ability to capture dynamic system behaviors (Venkatraman et al., 2019) in entirety. This also hinders the generalization in detection models (Naseer et al., 2021), along with a lack of multiple labeled data. These gaps must be addressed to create strong, interpretable, and adversarially robust malware detection systems (Bhaskara et al., 2023). To address these gaps, this study proposes an explainable, adversarially robust detection framework using GRU and LSTM-based architectures (Andrade et al., 2023).

3. Methodology

3.1 Dataset Description

This study uses a dataset from VirusTotal, a widely relied-upon malware analysis system, which contains a rich feature set meant to capture the temporal, system-level, and behavioral attributes of processes. It is a binary classification dataset of two classes: malware and benign. The features fall into four broad categories to capture system and process behaviors more completely. These metrics are temporal (timestamps in milliseconds) or system-related (prior, static_prio) metrics. The state features give important context on what is happening with the processes (e.g., whether they are running, sleeping, or being stopped) and can help identify malicious activity (Cui et al., 2019). Another set of core features is memory statistics, detailing trends in resource usage often leveraged by malware. For instance, metrics such as cached_hole_size, free_area_cache, and task_size provide information about memory management that malware could abuse (Mpanti et al., 2018). Page fault counters are measured together with nvcsw (walk-through context switches), nivcsw (involuntary context switches), maj_flt (major page faults), and min_flt (minor page faults), and they can provide insight into how often the system is being disrupted or distorted by rogue malware (Ucci et al., 2018). Lastly, timing properties and features, such as time (user CPU time), time (system CPU time), and time (thread executed time), provide a temporal view of process behavior, which can help in the detection of subtle, rare abnormalities in resource consumption (Kaya et al., 2024).

3.2 Data Preprocessing

Data pre-processing is an important step to ensure that the dataset's quality is good enough and that the dataset is in a format that can be fed into a sequential deep-learning model (Sarna et al., 2024). All numerical features have been normalized between 0 and 1 using MinMax scaling (Namavar et al., 2020). The first normalization step was needed to account for differences in the scales of the various features since the values of the inputs for the model training and convergence processes needed to be as efficient as possible, specifically memory use, CPU times, and the number of page faults. Data in the sets were segmented into three categories: training set, validation set, and application set (Liras et al., 2021). The training set (just over 75 - 85% of the data) was used to optimize model parameters, and the validation set was used to evaluate performance during training and to tune hyperparameters (Guendouz et al., 2023). The test set was not used until this final evaluation step to allow for an unbiased estimate of model generalization ability (Wang et al., 2021). The data is temporally ordered, so sequences were preserved in the train/test splits temporally.

3.3 Model Architectures

LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) networks are two of the most common architectures of recurrent neural networks that can be applied to analyze sequential data and were deployed in this study for malware detection. The reason we chose these two models specifically was that they are both based on long short-term memory (LSTM), which has proven effective in modeling long-term dependencies in temporal data, an important characteristic to guarantee the detection of patterns in system behaviors and anomalous detection of any malicious activity (Alaeiyan et al., 2023) refer to LSTM and GRU networks as indirect representations and found that these networks use hidden states over time to process data organized as sequences. LSTM models are particularly effective at modeling long-term dependencies thanks to their unique gating mechanisms, which allow them to limit how much information is allowed through (Manthena et al., 2023). The GRU architecture is arguably simpler, and GRU models with fewer parameters have been shown to achieve competitive performance and are computationally cheaper than the LSTM, which has no drop in accuracy (Galli et al., 2024).

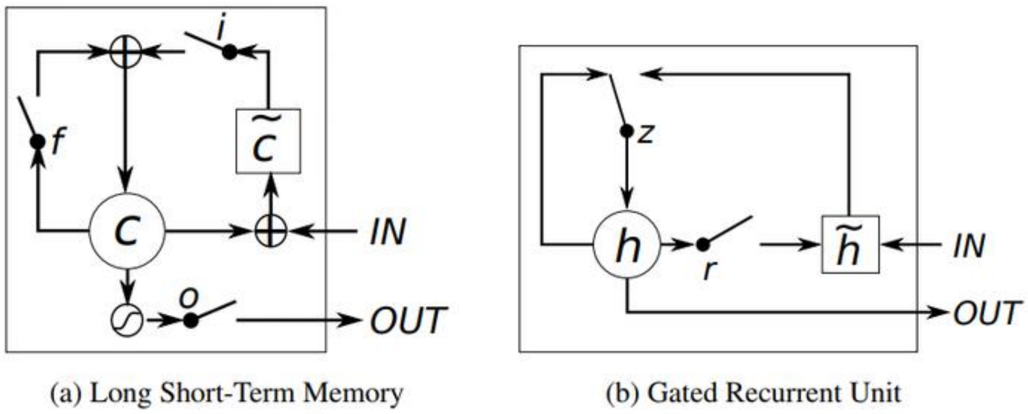


Fig. 1: *Illustration of (a) LSTM and (b) gated recurrent units. (a) i, f and o are the input, forget and output gates, respectively. c and c~ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and h~ are the activation and the candidate activation.(gabormelli)*

Both architectures are suitable for sequentially related input features like system logs, resource usage logs and process state (Kaya et al., 2024). The layered architecture of the models is optimized for binary classification (Kolosnjaji et al., 2016). This first layer processed normalized feature vectors for temporal and process-state metrics (Danilevsky et al., 2020).

Parameter	LSTM	GRU
Activator	Sigmoid	Sigmoid
Optimizer	Adam	Adam
Learning Rate	Default (Adam)	Default (Adam)
Loss Function	Binary Cross-Entropy	Binary Cross-Entropy
Layers	2	2
Neurons per Layer	64 (1st layer), 32 (2nd layer)	64 (1st layer), 32 (2nd layer)
Dropout Rate	0.2	0.2
Batch Size	32	32
Epochs	10	10
Patience for Early Stopping	3	3

Table 1: *Description of LTSM and GRU Architecture*

To learn sequential dependencies, we used two recurrent layers: the first laid down 64 LSTM/GRU units with return sequences set to true for the layers to continue looking for patterns after this (Galli et al., 2024). In contrast, the second one contained 32 units to refine the patterns even more (Kolosnjaji et al., 2016). To avoid overfitting, we introduced dropout layers with a value of 20%

after each recurrent layer (Mane & Rao, 2021). The final dense layer used a sigmoid activation function that gave the sample probability to be in a vibration or benign class (Namavar et al., 2020). We used binary cross-entropy loss function for both models as it is the suggested loss function for binary classification tasks. We used the Adam optimizer (Gupta et al., 2025), which adapts the learning rate, allowing the model to converge more quickly and stably. Training was stopped if the validation loss did not improve for three epochs (early stopping). Training consisted of a maximum of 10 epochs, using a batch size of 32 to balance compute efficiency with performance. This means that the data was reshaped to the sequential dimension, which will be needed for all the models I am going to present so that every model architecture can expect its input in that specific format. The approach included adversarial testing and retraining to strengthen the LSTM and GRU models (Galli et al., 2024). Most of the generated adversarial samples are perturbed samples based on a small offset of feature values, adding Gaussian noise while ensuring they still look realistic. This method focused on important system behaviors, including memory use, page faults, and CPU timings (Alaeiyan et al., 2023), to evaluate model sensitivity to perturbations. The general dataset was then supplemented with these adversarial samples to enhance the training set and model robustness (Ribeiro et al., 2016). Instead of directly exposing the models to adversarial examples, the models were retrained with clean and adversarial data (Kaya et al., 2024), learning to differentiate malware characteristics from noise. In order to preserve transparency and ensure the model's reliability, explainability tools (Lundberg et al., 2017) such as SHAP and LIME were employed.

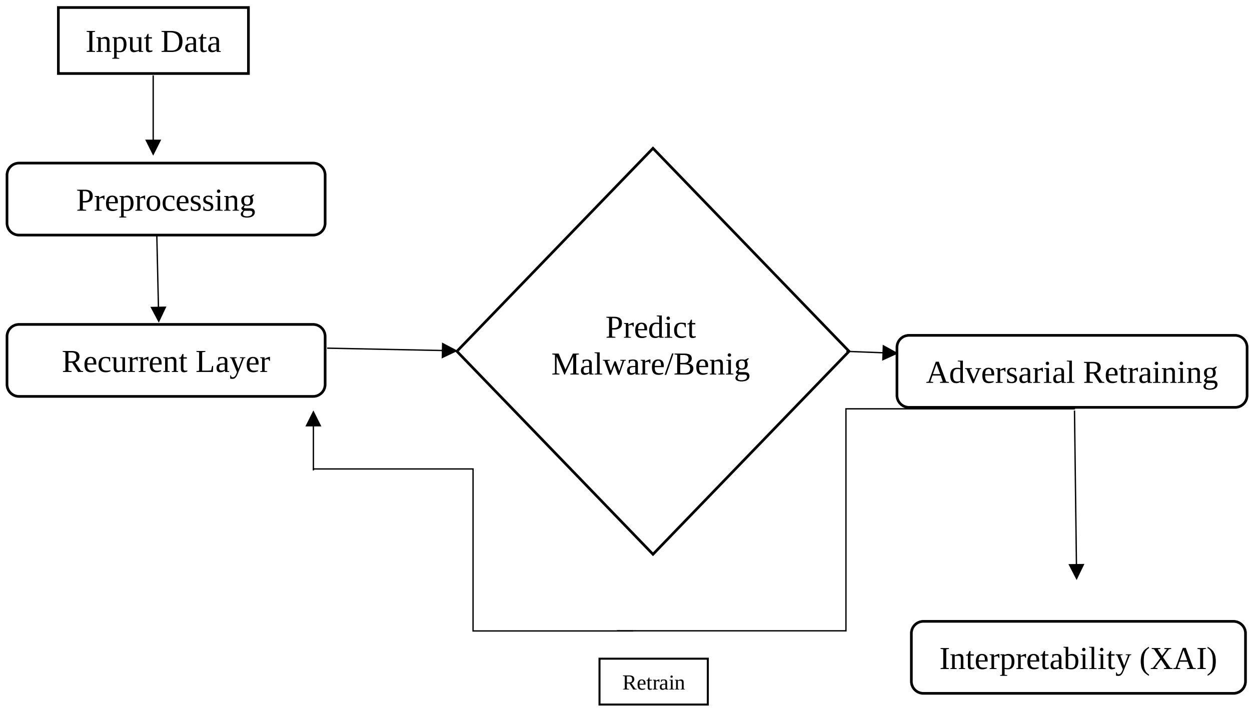


Fig. 2: *Methodological workflow of proposed malware detection system*

SHAP gave a global map of feature significance reporting high-significance features like major page faults (maj_flt), user CPU time (time), and process priority (prior) in malware detection

(Afifah & Stiawan, 2019). An example of local interpretations would be the LIME (Local Interpretable Model-agnostic Explanations) algorithm, which perturbs specific input parts and fits a simple model to explain specific predictions (Lundberg & Lee, 2017). Furthermore, the framework maintained both global and local interpretability by integrating SHAP and LIME to close the transparent versus performance gap in cybersecurity applications (Kuppa & Le-Khac, 2020).

4. Result Analysis

4.1 Model Performance

The performance of the LSTM and GRU models was evaluated using key metrics: accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). These metrics were analyzed under three scenarios: original data, adversarial data, and retrained adversarial data.

Metric	**LSTM**	**GRU**
Accuracy	99.44%	99.92%
Precision	99.39%	99.92%
Recall	99.49%	99.93%
F1-Score	99.44%	99.92%
AUC-ROC	99.96%	99.99%

Table 2: *Comparison of accuracy, precision, recall, F1-score between LTSM and GR*U

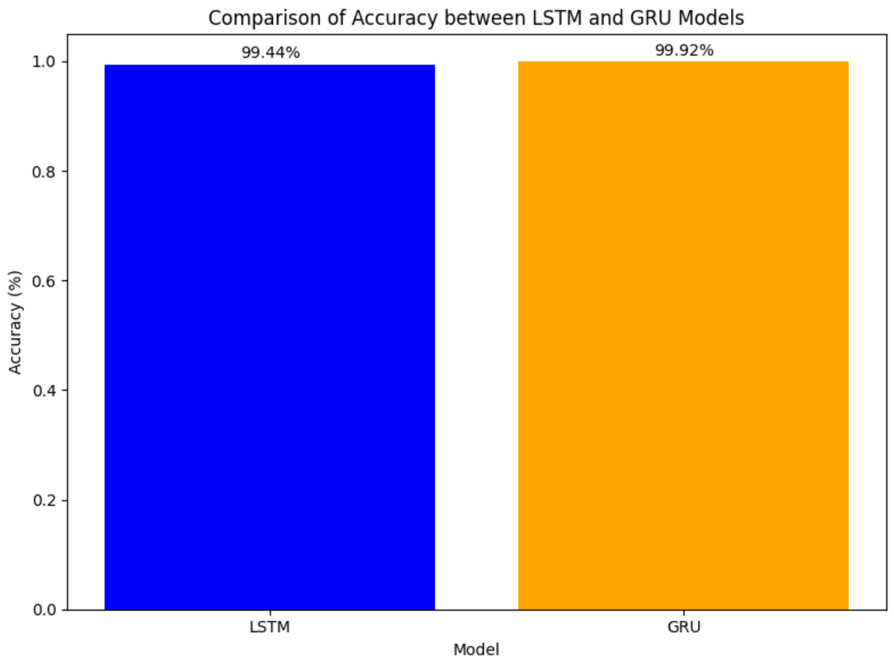


Fig. 3: *Accuracy comparison between LTSM and GRU*

Both models achieved very good classification performance on the raw data. Figure 3 indicates that the performance of the GRU with an accuracy of 99.92% slightly outperformed LSTM, 99.44%. Both models achieved AUC-ROC values greater than 0.99, demonstrating extreme discrimination between malware and benign samples. GRU showed greater precision and recall by producing the least false positive and false negative instances, which certifies that the models could be successfully generalized with the clean datasets. Adversarial robustness plays a crucial role in evaluating the reliability of AI models in cybersecurity by ensuring they are robust to attempts to tamper and evade detection. This work focuses on adding especially noise into input features in an adversarial testing method, e.g., effective va, reliable, effective memory usage, page fault, timings of process, kernel and process states, etc. As can be seen from Figure 4, both models performed worst on our dataset. GRU managed to retain 77.96% accuracy, while LSTM was 74.65%. Although this reduced accuracy, GRU showed better robustness overall, achieving higher accuracy and recall than LSTM, demonstrating its capacity to detect malware in adversarial environments. Our models both achieved an AUC-ROC above 0.85, indicating that our models could still discriminate malware from benign examples under adversarial perturbation.

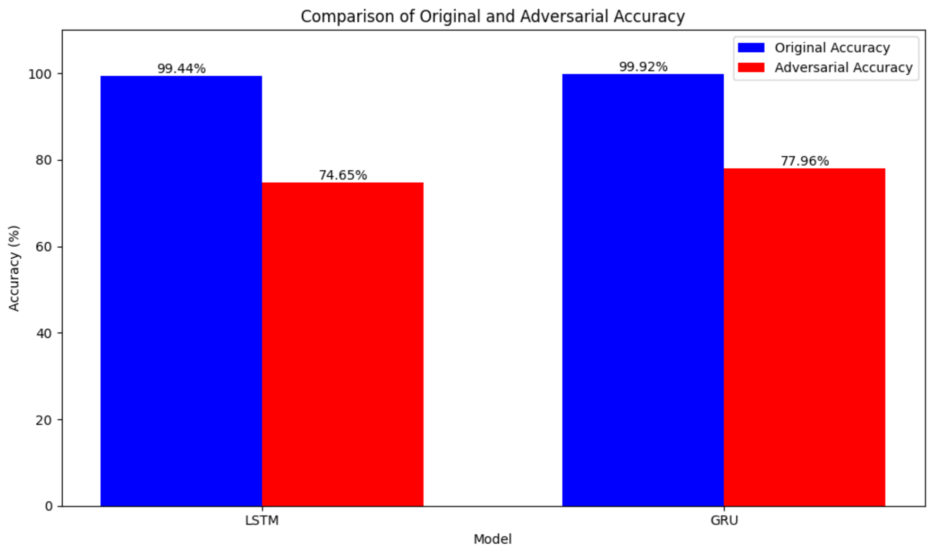


Fig. 4: *Comparison of original and adversarial accuracy between LTSM and GRU*

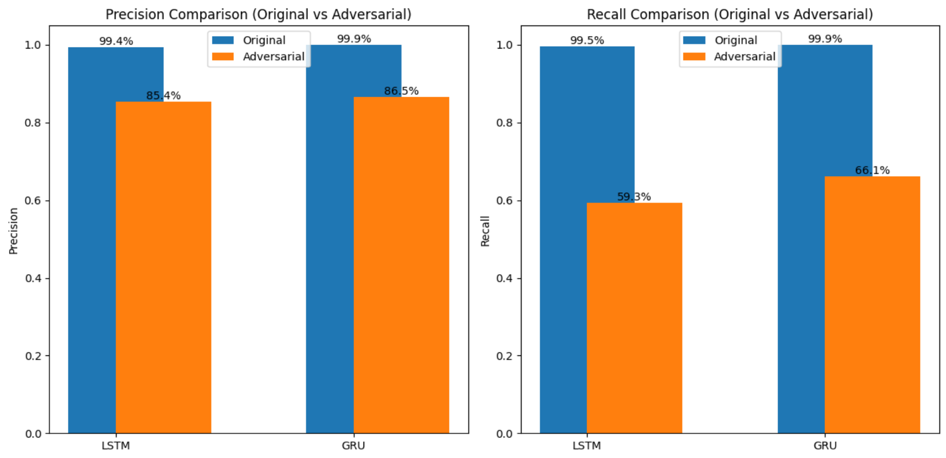


Fig. 5: *Precision and recall comparison for both original and adversarial between LTSM and GRU*

Under adversarial test conditions, both models' precision and recall performance were severely affected, meaning the ability of the models to classify the test samples correctly was significantly reduced. If we discuss the LSTM model, as shown in Figure 5, the accuracy was 99.4 % for the original and 85.4 % for the adversarial. Similarly, the initial recall of 99.5% of LSTM decreased very quickly to 59.3% in adversarial conditions, accounting for more false negatives. GRU, on the other hand, was less affected. Initially, the precision was 99.9%, adversarial precision dropped to 86.5%, and the model performed better than LSTM.

GRU also achieved better recall than LSTM, with GRU's original recall being 99.9% and adversarial recall of 66.1%, suggesting that GRU is better at identifying malware after intended perturbation. Both models saw a sharp drop in performance, with recall taking the biggest hit. In contrast, GRU achieved consistently improved performance over LSTM when manipulating adversarial data, achieving higher precision and recall; it appears GRU was the more stable solution to the adversarial problem. The ROC curve further corroborated these findings and AUC score analysis, demonstrating the clear degradability the models could provide for malware and benign samples under both original and adversarial scenarios. **Figure 6** shows that the AUC of LSTM was 0.9996 without adversarial perturbations and decreased to 0.8556 with adversarial perturbations. However, GRU retained an ideal AUC of 1.0000 on original data, decreasing slightly to 0.8681 under adversarial conditions, indicating enhanced robustness. The performance of either model was near perfect in original conditions on the task, with both attaining almost a perfect AUC score. However, when evaluated on the adversarial ROC curves, the degradation was visible, with GRU demonstrating noteworthy resilience over LSTM.

Adversarial retraining significantly improved the models' performance under adversarial conditions. By incorporating adversarial samples into the training process, **Figure 7** shows that the GRU model achieved an accuracy of 92.61%, while LSTM slightly outperformed it with 92.69%. Enhanced recall and F1 scores highlighted the models' improved ability to reliably detect malware in adversarial scenarios.

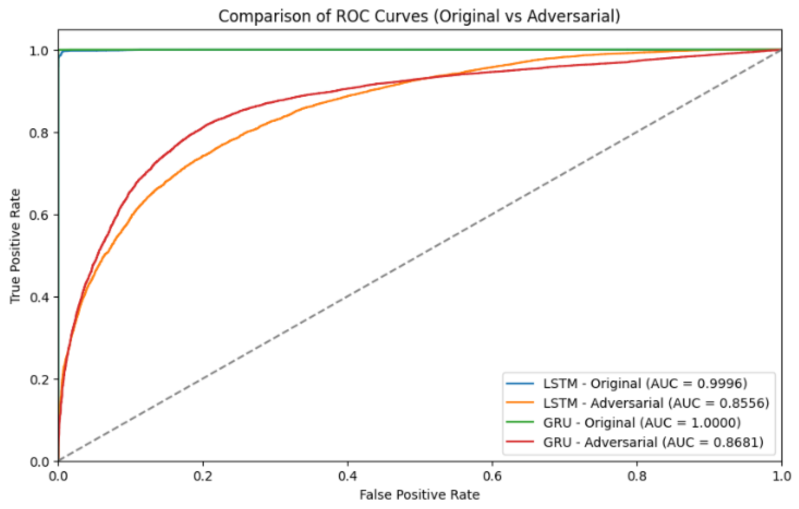


Fig. 6: *ROC-AUC comparison between LTSM and GRU*

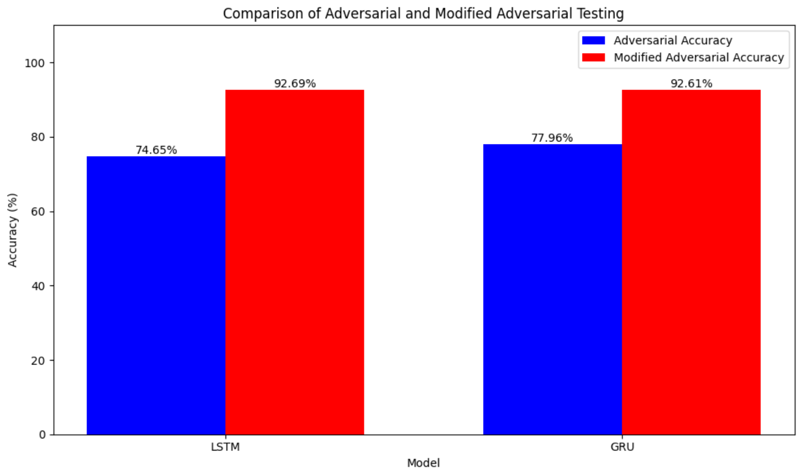


Fig. 7: *Comparison of adversarial retraining for LTSM and GRU*

Figure 8 presents the testing accuracy for both models under original and modified adversarial conditions. In the first figure (original performance), both models demonstrate strong performance from the start, with GRU outperforming LSTM in the early epochs, converging faster, and achieving nearly perfect accuracy (~99.9%) by the 10th epoch. Both models stabilize around the seventh epoch, suggesting excellent capacity for behavior-based malware detection on the original dataset. Both models show steady improvement in the second figure (modified adversarial performance), with GRU maintaining a slight advantage throughout. GRU achieves a final accuracy of 96.57%, slightly higher than LSTM's 96.21%, reflecting its superior robustness to adversarial conditions. Both models converge after the eighth epoch, demonstrating the effectiveness of adversarial training in enhancing their resilience to adversarial manipulation. To mitigate these vulnerabilities, adversarial retraining was conducted using a combined dataset of original and adversarial samples.

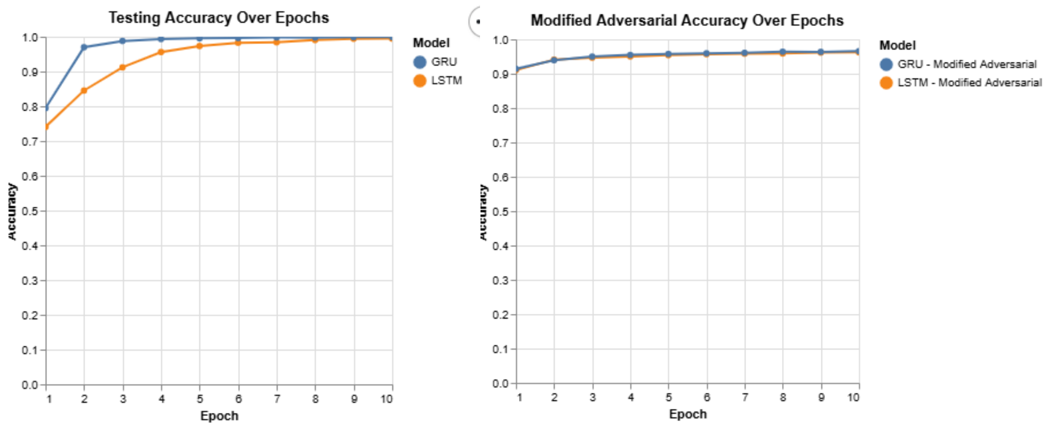


Fig. 8: *Comparison of LSTM and GRU models' accuracy over epochs for original and modified adversarial data, highlighting GRU's superior performance.*

As illustrated in **Figure 9**, both the retrained GRU (92.61%) and LSTM (92.69%) models realized superior degrees of precision. Both models consistently outperformed others on recall and F1, reinforcing their robustness to adversarial attacks. These results highlight the need for adversarial retraining to enhance the robustness of malware detectors, ensuring that these systems perform as intended in dangerous real-world environments.

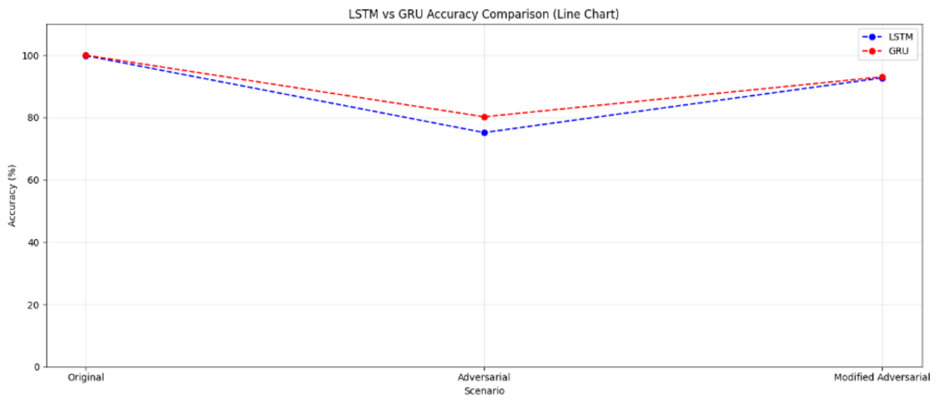


Fig. 9: *LTSM and GRU accuracy comparison*

This will enable the existing models to generalize better to unseen perturbations and thus offer protection against sophisticated evasion techniques now seen in some of the most advanced malware variants. We can firmly assert that this assertive behavior is a strength in the adversarial setup, which, whenever, bounds the performance reliability and effectiveness of the framework, which serves as one of the prime criteria for any modern-day malware detection techniques.GRU's consistency across all conditions, combined with its computational efficiency, makes it a strong candidate for real-world, behavior-based malware detection.

4.2 Feature Importance Analysis for LSTM and GRU using Explainable AI (SHAP)

A behavior-based malware detection framework developed in this study relied on a feature importance analysis using SHAP to better understand the data analysis results. This gave us a global overview of the most influential features for the LSTM and GRU models, contributing to interpretability and confirming the feature set. The LSTM model showed key features such as major page fault (maj_flt), which indicates abnormal memory access patterns; user CPU time (time), which reflects resource-intensive behavior of malware; static version (static_prio), which signals system-level manipulation; and time delta (time delta) that aids identifying suspicious behaviors, including zero-day threats. The distribution of importance of the features used in the LSTM model is shown in the SHAP summary plot (Figure 10), further revealing the prominent significance of the temporal and system-level metrics involved in accurate malware detection. As with the LSTM model, the GRU model also highlighted key features — such as major page faults (maj_flt), user CPU time (time), and static priority (static_prio) — as being significant in classifying benign and malicious behaviors. The GRU model also used the memory users (mm_users) feature, which gives an even more comprehensive description of the process behavior, raising the model performance on both architectures. The SHAP summary plot for GRU (Figure 10) is consistent with LSTM as a reliable model consistent with temporal and process-state features.

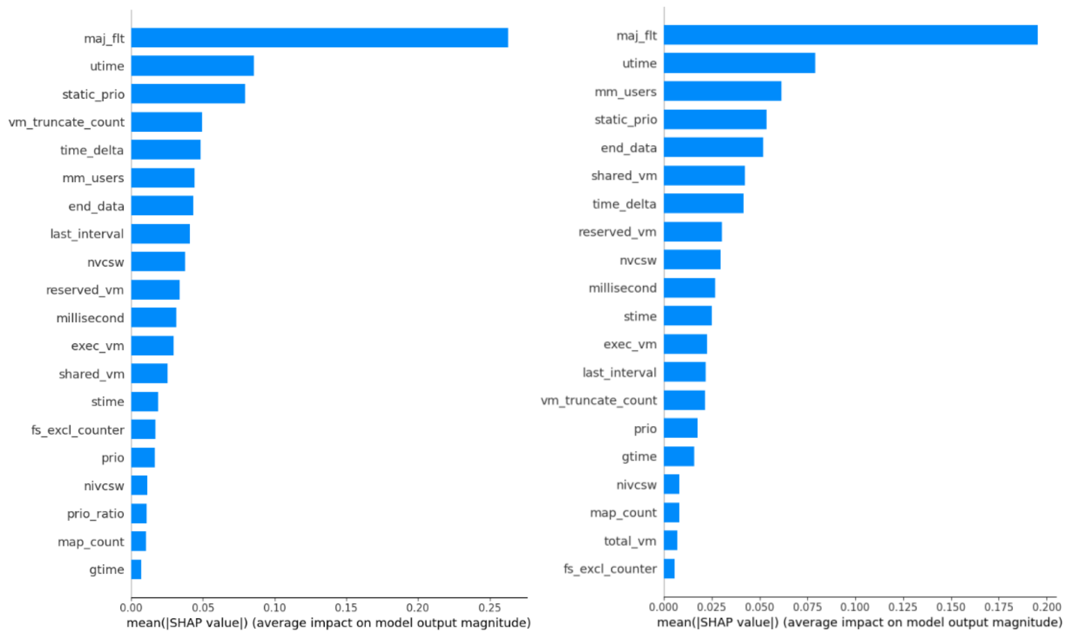
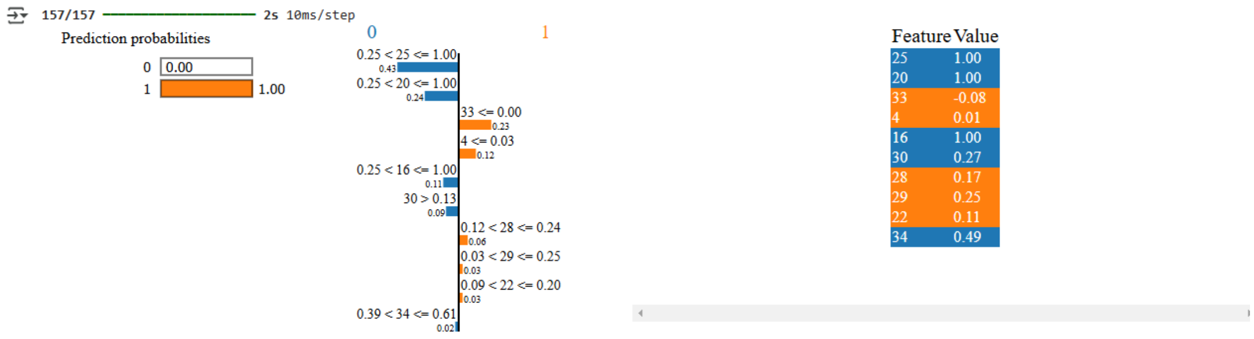


Fig. 10: *LSTM (left) and GRU (right) SHAP Summary*

The LSTM and GRU models deemed some of the same features important (maj_flt, time, static_prio), yet the GRU had higher SHAP values for values related to memory use (e.g., mm_users) and resource allocation features, indicating its responsiveness to complex and subtle behaviors. The disparity, along with the CRUB's computational efficiency, makes GRU well suited for real-time malware detection. In general, the analysis of feature importance using SHAP indicates that the significance of features, such as maj_flt, time, and static_prio, in the behavior-based malware detection scheme is inherent and remains unchanged, thus substantiating the stability and credibility of the proposed architecture for high-impact cybersecurity applications.

4.3 Local Interpretability with Explainable AI (LIME) for LSTM and GRU

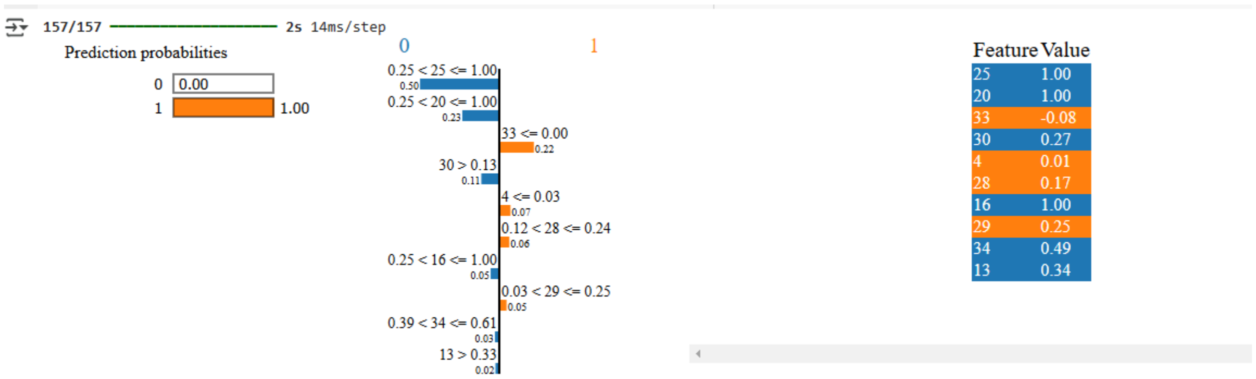
Local interpretability via LIME (Local Interpretable Model-Agnostic Explanations) enriches the understanding of how each prediction of the LSTM and GRU models was derived, thus increasing transparency and trust when risk is high, as with cybersecurity applications. For LIME, the explanations are generated via perturbing input samples and determining how the model predictions change. In a single instance studied for the LSTM model, a sample identified as "malware" was predominantly characterized by an increase in major page faults (maj_flt), indicating unusual behavior regarding memory usage, followed by an increase in the user CPU time (utime) which suggested that the process performed intensive resource usage typical for malware and deviations in the static priority (static_prio), which suggested manipulation of system resources by apparitional software.


```
157/157 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step
```

Feature	Value
25	1.00
20	1.00
33	-0.08
4	0.01
16	1.00
30	0.27
28	0.17
29	0.25
22	0.11
34	0.49

Fig. 12: *LSTM LIME*

In the LIME explanation, the bar chart in Figure 12 effectively shows how these features positively contribute to classifying the sample as malware. For instance, it explained more than 40% of prediction probability, followed by time, which explained about 25%. The greater granularity of potential insights allows cybersecurity analysts to extrapolate specific model features that indicate the rationale behind the model's action and decide on further actions. A similar case study for the GRU model highlighted the effectiveness of LIME in providing local interpretability. In this instance, a sample classified as "malware" was primarily influenced by major page faults (maj_flt), mirroring the LSTM model's reliance on this feature. Additionally, the GRU model demonstrated increased sensitivity to memory users (mm_users), capturing nuanced memory allocation patterns, and flagged subtle irregularities in CPU timing (time_delta), emphasizing its ability to detect temporal anomalies. This reinforces the GRU model's proficiency in identifying complex behaviors indicative of malware. The LIME explanation (Figure 13) visualizes features' positive and negative contributions. For example, maj_flt alone contributed more than 35% in malware classification, while mm_users and time_delta contributed another large portion. Those discrete classes provide action points for analysts to understand the behaviors that made the GRU model perceive malware.



```
157/157 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step
```

Feature	Value
25	1.00
20	1.00
33	-0.08
30	0.27
4	0.01
28	0.17
16	1.00
29	0.25
34	0.49
13	0.34

Fig. 13: *GRU LIME*

Despite the similarities, both models were focused on different features. The maj_flt and time remained consistent across both models, but the GRU model placed a more significant emphasis on anything related to memory usage, such as mg_users. This further emphasized its superior capacity to generalize and see patterns across resource allocations. This difference stresses GRU's superiority in interpretability and robustness in behavior-based malware detection.

4.4 Comparison with Prior Work

Study	Model	Accuracy on Clean Data (%)	Adversarial Robustness (%)	Explainability (XAI Tools)	Feature Set
Proposed Framework (GRU)	GRU	99.92	92.61	SHAP, LIME	Temporal and process-state features (e.g., maj_flt, utime, prio)
(Gupta et al., 2025)	GRU	99.70%	N/A	None	Android malware detection temporal features
(Manthena et al., 2023)	SDEL	99.87%	N/A	SHAP, LIME, LRP	Pixel representations of malware binaries.
(Galli et al., 2024)	LSTM, GRU	99.43 (LSTM)	N/A	SHAP, LIME, Attention, LRP	Behavioral API call sequences (e.g., API call importance in malware prediction)
(Shakib, 2023)	GRU (GenetícAI)	99.77	N/A	None	Android malware detection features

Zhang et al. (2021)	CNN + GAN	94.5	85.4	None	Malware images generated by GAN
Venkatraman et al. (2019)	GRU + CNN	97.5	85.2	None	API calls, system events

Table 3: *Comparison with existing work*

This table highlights that the Proposed Framework (GRU) outperforms many existing methods, particularly excelling in accuracy on clean data and demonstrating robust adversarial performance. Unlike previous works, it integrates SHAP and LIME for explainability, ensuring transparency in predictions and addressing the black-box nature of many deep learning models. Additionally, the study leverages temporal and process-state features, which significantly enhance the detection of both known and novel malware, surpassing traditional signature-based and handcrafted feature methods.

5. Implications, Limitations and Future Work

Behavior-based malware detection architecture has a significant advantage in real-life security systems owing to its wide applicability when combined with IDSs and EDSs to detect better malicious behavior (Afifah & Stiawan, 2019). With behavior-based detection, unlike signature-based systems, novel malware detection becomes a challenge. Based on real-time, behavior-driven detection (Andrade et al., 2019), the framework minimizes malware damage and its explainability with LIME and SHAP, which builds sufficient trust for in-the-wild deployment in high-risk environments (Shakib, 2023). The GRU model is included to facilitate data processing (Galli et al., 2024). However, some limitations are reliance on the VirusTotal dataset, which does not cover all types of malware, and scalability issues with the increasing datasets that must be addressed. Computational complexity (especially GRU and LSTM models have posed challenges using low-resource devices). Future work should also focus on hybrid architectures combining GRU and attention mechanisms to improve detection accuracy and adversarial testing to overcome more sophisticated evasion strategies. Moreover, generalizes and strengthens the framework to react to cybersecurity threats, thus increasing the diversity of the sources included in the dataset. So, ongoing refinement and extension of the framework will eventually make it relevant to all environments, thus enhancing the reliability of the framework in dynamic and real-world settings. This continuous development will ensure it stays relevant and can combat new and more advanced malware threats.

6. Conclusion

This study introduces a robust, explainable malware detection framework that integrates temporal and process-state features with advanced GRU and LSTM models. The framework utilizes these models to capture sequential dependencies and process system behaviors over time, significantly improving malware detection over traditional approaches. In particular, the GRU model also gave superior outperformance in adversarial situations, recording a 92.61% accuracy in adversarial retraining, making it resilient against attacks. It is important in real-world cyber security systems that the detection of evasive adversaries is accurate, and showing strong performance against these adversarial attacks highlights the framework's utility. On the other hand, the integrated SHAP and LIME interpretability tools give additional confidence and transparency with important global features and explanations of local models. They allow cybersecurity practitioners to understand how predictions are derived through a model, thus empowering confidence in employing AI-based detection systems during critical times. Using SHAP, we aggregated features that filtered out malware from benign processes, and significant features were identified as important (e.g., maj_flt, time, and prior). Besides demonstrating the framework's flexibility, LIME has exposed the frame's single predictions, which are also very important in tuning security protocols for improvement and better opportunity prediction. Its potential to withstand adversarial manipulations while maintaining high detection performance positions it as a promising solution for real-world, high-risk malware detection and prevention applications.

8. References

Afifah, N., & Stiawan, D. (2019). The implementation of deep Neural Networks algorithm for malware classification. Computer Engineering and Applications Journal, 8(3), 189–202. https://doi.org/10.18495/comengapp.v8i3.294

Ahmed, M., Islam, S. R., Anwar, A., Moustafa, N., & Pathan, A. K. (2022). Explainable artificial intelligence for cyber security. In Studies in computational intelligence. https://doi.org/10.1007/978-3-030-96630-0-0

Alaeiyan, M., Parsa, S., & P., V. (2023). Sober: Explores for invasive behaviour of malware. In Journal of Information Security and Applications (Vol. 74, p. 103451). Elsevier BV. https://doi.org/10.1016/j.jisa.2023.103451

Ambekar, N., Devi, N.N., Thokchom, S. et al. (2024). TabLSTMNet: enhancing android malware classification through integrated attention and explainable AI. Microsyst Technol. https://doi.org/10.1007/s00542-024-05615-0

Amer, E., Zelinka, I., & El-Sappagh, S. (2021). A Multi-Perspective malware detection approach through behavioral fusion of API call sequence. Computers & Security, 110, 102449. https://doi.org/10.1016/j.cose.2021.102449

Andrade, E. O., et al. (2019). A model based on LSTM neural networks to identify five different types of malware. Procedia Computer Science, 159, 182-191. https://doi.org/10.1016/j.procs.2019.09.173

Bhaskara, A., Mitchell, F., Smith, P., & Kasera, S. K. (2023). Exploring Adversarial Attacks on Learning-based Localization. ACM Digital Library, 15–20. https://doi.org/10.1145/3586209.3591398

Chamola, V., Hassija, V., Sulthana, A. R., Ghosh, D., Dhingra, D., & Sikdar, B. (2023). A review of Trustworthy and Explainable Artificial Intelligence (XAI). IEEE Access, 11, 78994–79015. https://doi.org/10.1109/access.2023.3294569

Charmet, F., Tanuwidjaja, H. C., Ayoubi, S., Gimenez, P., Han, Y., Jmila, H., Blanc, G., Takahashi, T., & Zhang, Z. (2022). Explainable artificial intelligence for cybersecurity: a literature survey. Annals of Telecommunications, 77(11–12), 789–812. https://doi.org/10.1007/s12243-022-00926-7

Chen, L., Ye, Y., & Bourlai, T. (2017). Adversarial Machine Learning in Malware Detection: Arms Race between Evasion Attack and Defense. 2017 European Intelligence and Security Informatics Conference (EISIC), 99–106. https://doi.org/10.1109/eisic.2017.21

Cui, Z., Du, L., Wang, P., Cai, X., & Zhang, W. (2019). Malicious code detection based on CNNs and multi-objective algorithm. Journal of Parallel and Distributed Computing, 129, 50–58. https://doi.org/10.1016/j.jpdc.2019.03.010

Danilevsky, M., Qian, K., Aharonov, R., Katsis, Y., Kawas, B., & Sen, P. (2020). A Survey of the State of Explainable AI for Natural Language Processing. Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, 447–459. https://doi.org/10.18653/v1/2020.aacl-main.46

Finder, I., Sheetrit, E., & Nissim, N. (2022). Time-interval temporal patterns can beat and explain the malware. Knowledge-Based Systems, 241, 108266. https://doi.org/10.1016/j.knosys.2022.108266

Frontier - https://frontiere.io/insights/artificial-intelligence-for-businesses/how-explainable-ai-can-help-overcome-mistrust-regarding-the-adoption-of-artificial-intelligence/

Gabormeli - https://www.gabormelli.com/RKB/Gated_Recurrent_Unit_%28GRU%29

Galli, A., La Gatta, V., Moscato, V., Postiglione, M., & Sperlì, G. (2024). Explainability in AI-based behavioral malware detection systems. Computers & Security, 141, 103842. https://doi.org/10.1016/j.cose.2024.103842

Gupta, B. B., Gaurav, A., Arya, V., Bansal, S., Attar, R. W., Alhomoud, A., & Psannis, K. (2025). Earthworm Optimization Algorithm based Cascade LSTM-GRU model for Android Malware Detection. Cyber Security and Applications, 100083. https://doi.org/10.1016/j.csa.2024.100083

Guendouz, M., & Amine, A. (2023). A new feature selection method based on Dragonfly algorithm for Android malware detection using machine learning techniques. International Journal of Information Security and Privacy, 17(1), 1–18. https://doi.org/10.4018/ijisp.319018

Kaya, Y., Chen, Y., Saha, S., Pierazzi, F., Cavallaro, L., Wagner, D., & Dumitras, T. (2024). Demystifying Behavior-Based Malware Detection at Endpoints (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2405.06124

Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016). Deep learning for classification of malware system call sequences. In Lecture notes in computer science (pp. 137–149). https://doi.org/10.1007/978-3-319-50127-7_11

Kuppa, A., & Le-Khac, N. (2020). Black Box Attacks on Explainable Artificial Intelligence(XAI) methods in Cyber Security. 2022 International Joint Conference on Neural Networks (IJCNN), 1–8. https://doi.org/10.1109/ijcnn48605.2020.9206780

Liras, L. F. M., De Soto, A. R., & Prada, M. A. (2021). Feature analysis for data-driven APT-related malware discrimination. Computers & Security, 104, 102202. https://doi.org/10.1016/j.cose.2021.102202

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems (pp. 4765–4774). https://doi.org/10.48550/arXiv.1705.07874

Malhotra, A. (2021). The postpandemic future of work. Journal of Management, 47(5), 1091–1102. https://doi.org/10.1177/01492063211000435

Mane, S., & Rao, D. (2021). Explaining network intrusion detection system using explainable AI framework. arXiv preprint arXiv:2103.07110. https://doi.org/10.48550/arXiv.2103.07110

Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2024). A systematic literature review on Windows malware detection: Techniques, research issues, and future directions. Journal of Systems and Software, 209, 111921. https://doi.org/10.1016/j.jss.2023.111921

Manthena, H., Kimmel, J. C., Abdelsalam, M., & Gupta, M. (2023). Analyzing and explaining Black-Box models for online malware detection. IEEE Access, 11, 25237–25252. https://doi.org/10.1109/access.2023.3255176

Mehrban, A., & Ahadian, P. (2023). Malware Detection in IoT Systems using Machine Learning Techniques. International Journal of Wireless & Mobile Networks, 15(6), 13–23. https://doi.org/10.5121/ijwmn.2023.15602

Mpanti, A., Nikolopoulos, S. D., & Polenakis, I. (2018). Malicious software detection and classification utilizing temporal-graphs of system-call group relations. arXiv preprint arXiv:1812.10748. https://doi.org/10.48550/arXiv.1812.10748

Namavar, A., et al. (2020). Two-hidden layered extreme learning machine for IoT malware detection. IEEE Internet of Things Journal, 7(5), 3995-4003. https://doi.org/10.1109/JIOT.2020.2977345

Naseer, M., Rusdi, J. F., Shanono, N. M., Salam, S., Muslim, Z. B., Abu, N. A., & Abadi, I. (2021). Malware Detection: Issues and challenges. Journal of Physics Conference Series, 1807(1), 012011. https://doi.org/10.1088/1742-6596/1807/1/012011

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1135-1144). https://doi.org/10.1145/2939672.2939778

Sarna, F. Z. & Chowdhury, N. H. I. (2024). Mediating Role of Brand Perception and Big Data Analytics between Consumer Experiential Components and Consumer Behavior, International Journal of Science and Business, 42(1), 193-211. DOI: https://doi.org/10.58970/IJSB.2492

Shakib, M. (2023). Android Malware Detection Approach's Based on Genetic Ai, Cnn, Rnn, Lstm, Gru, and Active Learning. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4611916

Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. Journal of Systems Architecture, 112, 101861. https://doi.org/10.1016/j.sysarc.2020.101861

Ucci, D., Aniello, L., & Baldoni, R. (2018). Survey of machine learning techniques for malware analysis. Computers & Security. https://doi.org/10.1016/j.cose.2018.11.001

Venkatraman, S., Alazab, M., & Vinayakumar, R. (2019). A hybrid deep learning image-based analysis for effective malware detection. Journal of Information Security and Applications, 47, 377–389. https://doi.org/10.1016/j.jisa.2019.06.006

VirusTotal : https://www.virustotal.com

Wang, Z., Fok, K. W., & Thing, V. L. (2021). Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. Computers & Security, 113, 102542. https://doi.org/10.1016/j.cose.2021.102542

Zhang, Z., Li, Y., Wang, W., Song, H., & Dong, H. (2022). Malware detection with dynamic evolving graph convolutional networks. International Journal of Intelligent Systems, 37(10), 7261–7280. https://doi.org/10.1002/int.22880