RansFighter: a GRU-based Tool for Ransomware Detection

Jaouhar Fattahi[1] ; Mohamed Mejri[1] ; Ridha Ghayoula[2] ; Sawssen Jalel[3];
Laila Boumlik[1] and Feriel Sghaier[4]

***Abstract*— In the current landscape of IT, ransomware attacks pose a major threat to cybersecurity resulting in significant monetary losses and data breaches. The detection of ransomware in time presents a challenge due to its constant evolution and complex strategies for escaping detection. This study introduces a deep learning tool —named RansFighter—based on Gated Recurrent Unit (GRU) specifically developed for ransomware detection. Our model shows, at test time, an Accuracy of 96.67%, a Precision of 97.01%, a Recall of 96.37%, an F1-Score of 96.69% and an Area Under the Curve (AUC) of 96.67%. It showcases the potential of GRUs as valuable assets to safeguard systems against ransomware threats.**

***Index Terms*— Ransomware, Detection, Deep Learning, Cybersecurity.**

I. Introduction

Ransomware attacks have become increasingly common [1]–[3]. They are considered one of the cybersecurity threats of our era. Servers and computers affected by ransomware face a situation where their data is encrypted by hackers using many techniques like fishing [4]–[6]. They are asked to pay a ransom for its decryption. This has led to operational losses not only for individuals but also for organizations and governmental bodies. With the rise in both the frequency and complexity of these attacks, the usual security defense mechanisms are no longer sufficient. Detecting ransomware poses a challenge because cybercriminals constantly adapt their tactics to evade detection measures effectively. Current ransomware strains use methods like encryption and obfuscation to conventional detection systems that rely on signatures or rules. Moreover, ransomware can infect networks quickly and cause harm in a short period of time [7], [8]. These obstacles underscore the importance of having detection tools that can identify ransomware. Therefore, it is crucial to develop systems that can detect them in real time in order to eliminate their risks. Machine learning and deep learning methods have become assets in bolstering cybersecurity [9]–[13] in such a scenario. In this context, Gated Recurrent Unit (GRU) [14]–[16] has become quite popular due to its effectiveness in handling both sequential data and tabular data. This type of data includes things like system logs network behavior and patterns which often hold clues about their activities. GRUs stand out because of their gating mechanisms that allow them to grasp both prolonged connections within this kind of information making them a choice for detecting ransomware incidents. Traditional methods typically require complex feature engineering to function, however, GRUs enable models to extract pertinent features from data and predict the presence of ransomware without the necessity for other manual and unreliable means of surveillance. The GRU model was introduced as an alternative to Long Short-Term Memory neural networks (LSTMs) [17]–[20] and incorporates two gates —the update gate and the reset gate—to control information flow effectively. This allows the model to preserve crucial details while filtering out unnecessary or outdated data elements without losing track of essential information in between layers in a neural network setup. GRUs possess attributes that render them well suited for real time uses like identifying ransomware where both accuracy and speed are of importance. In this study, we suggest a GRU-powered tool to spot incidents by making use of its abilities in analyzing data to spot suspicious actions effectively. In this paper, we propose a GRU-based tool, named Ransfighter, crafted to go through system records and network traffic details to uncover related trends that point ransomware activities. Through the integration of training methods and efficiency enhancements, our tool attains a high accuracy rate along with high precision, recall, AUC and F1-score rates, rendering it a reliable option for practical uses.

[1]Jaouhar Fattahi and Mohamed Mejri and Laila Boumlik are with the Department of Computer Science and Software Engineering, Laval University, Quebec, Canada. Jaouhar.Fattahi.1@ulaval.ca; Mohamed.Mejri@ift.ulaval.ca; Laila.Boumlik.1@ulaval.ca

[2]Ridha Ghayoula is with the Faculty of Engineering, University of Moncton, New Brunswick, Canada. Ridha.Ghayoula@umoncton.ca

[3]Sawssen Jalel is with TEKUP University, Tunis, Tunisia. Sawssen.Jalel@tek-up.tn

[4]Feriel Sghaier is with the Carthage National Engineering School, Carthage University, Tunis, Tunisia. Feriel.Sghaier@enicar.ucar.tn

II. GRU models

The Gated Recurrent Units have been created as an alternative to LSTMs—while proposing a more reduced complexity and fewer gates—and to overcome the limitations of conventional RNNs [21], reputed as having a too short memory. A GRU uses two key gates: the update gate and the reset gate, which govern the flow of information through the network.

A. Update Gate (z_t)

The update gate decides how much of the previous hidden state (h_{t-1}) should be kept and how much of the new information ($\tilde{h}_t$) should be integrated. This is expressed by Equation 1.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{1}$$

Where :

- x_t: Input at time t.
- h_{t-1}: Previous hidden state.
- W_z, U_z: Weight matrices.
- b_z: Bias.
- σ: Sigmoid activation function, which outputs values between 0 and 1.

B. Reset Gate (r_t)

The reset gate decides how much of the past information should be forgotten when defining the new candidate of the hidden state ($\tilde{h}_t$). This is expressed by Equation 2.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2}$$

C. Candidate Hidden State ($\tilde{h}_t$)

The candidate hidden state is determined using the reset gate. The reset gate r_t defines how much of the previous hidden state to use. This is expressed by Equation 3.

$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{3}$$

Where :

- $\odot$: Element-wise product.
- $\tanh$: Hyperbolic tangent activation function.

D. Final Hidden State (h_t)

The final hidden state combines the previous hidden state (h_{t-1}) and the candidate hidden state ($\tilde{h}_t$), weighted by the update gate. This is expressed by Equation 4.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \tag{4}$$

Table I shows some advantages of the GRU model compared to other comparative models.

III. Experiment

A. Dataset

In our experiment, we utilized the Ransomware Detection Dataset described in [26]. The dataset comprises 62,485 entries, each with 15 features of type int64. Table II summarizes the dataset features and their descriptions. The dataset includes a single binary class labeled as "Benign," where '0' indicates "Not Ransomware" and '1' represents "Ransomware". In our experiment, we used a sample of 3000 entries selected randomly while conserving the sample virtually balanced ("Not Ransomware" : 1495 entries, "Ransomware" : 1505 entries) as described in Fig. 1. Fig. 2 shows the data distribution among training, validation and test.

B. Model architecture

Our model architecture consists of a GRU-based one. Its first layer is an input one which accepts a 15-dimensional feature vector. The input is reshaped into a 2D tensor with dimensions (15, 1) to make it compatible with the GRU layer. The GRU layer, consisting of 64 units, processes this sequential data and outputs a feature representation. This representation is passed to a fully connected dense layer with 32 units and ReLU activation for further processing. A dropout layer with a 30% rate is applied to mitigate overfitting. The final output layer, comprising a single neuron with sigmoid activation, generates a probability for binary classification. The model is compiled using the Adam optimizer and a binary cross-entropy loss. Fig. 3 shows the architecture of our model.

Fig. 1: Class distribution

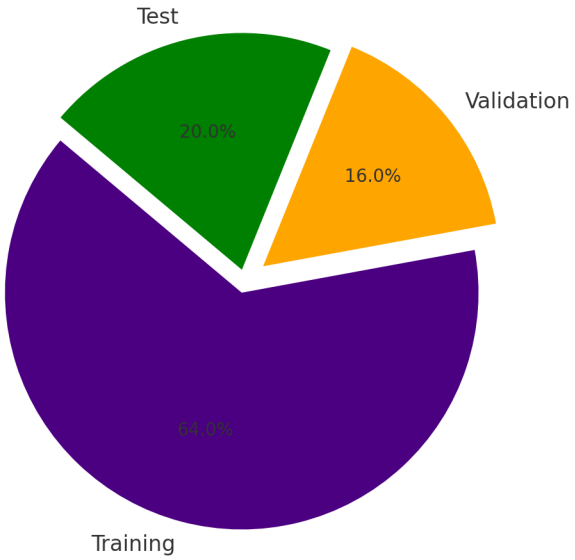


Fig. 2: Dataset repartition

C. Used metrics

To assess our model, we used the following metrics to assess our model: Accuracy, F1-Score, Precision, Recall, and AUC (Area Under the Curve).

- Accuracy : calculates the rate of accurately predicted instances (true positives and true negatives) to the total number of predictions, as expressed by Equation 5.

TABLE I: Advantages of GRU models compared to other models.

Aspect	Advantages of GRU models	Comparison to other models
Simplicity	GRUs have a simpler architecture with fewer parameters, reducing computational complexity.	LSTMs [17], [18] have more gates, increasing complexity and training time.
Training efficiency	GRUs train faster due to fewer parameters and lower memory requirements.	LSTMs and Transformers require more resources and training time.
Handling of sequential data	GRUs effectively capture temporal dependencies in sequential data.	Fully connected models lack this capability; LSTMs perform similarly but are more complex.
Avoiding overfitting	The reduced number of parameters in GRUs lowers the risk of overfitting on small datasets.	Transformers may overfit [22] due to their large number of parameters.
Performance on Small Datasets	GRUs perform well with limited data due to their simplicity and efficient learning.	LSTMs and Transformers often require larger datasets for optimal performance [23], [24].
Memory usage	GRUs use less memory, making them suitable for resource-constrained environments.	LSTMs and Transformers are more memory-intensive.
Gradient flow	GRUs mitigate the vanishing gradient problem effectively, ensuring stable training.	Vanilla RNNs suffer from vanishing gradients [25]; LSTMs handle it similarly but with more gates.

TABLE II: Dataset features

Feature Name	Description
Machine	The type of target machine (architecture) for which the file is intended.
DebugSize	Size of the debug information.
DebugRVA	Relative Virtual Address (RVA) of the debug information.
MajorImageVersion	The major version number of the image.
MajorOSVersion	The major version number of the required operating system.
ExportRVA	RVA of the export table.
ExportSize	Size of the export table.
IatVRA	RVA of the Import Address Table (IAT).
MajorLinkerVersion	The major version number of the linker.
MinorLinkerVersion	The minor version number of the linker.
NumberOfSections	The number of sections in the file.
SizeOfStackReserve	The size of the stack to reserve.
DllCharacteristics	Characteristics of the DLL.
ResourceSize	Size of the resources.
BitcoinAddresses	Presence of Bitcoin addresses within the file (usually indicator of ransomware).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

- Precision : calculates the ratio of true positive instances to the total number of positives (true and false), as expressed by Equation 6.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

- Recall : calculates the ratio of true positive instances to the total number of true positives and false negatives, as expressed by Equation 7.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

- F1-Score : is the balanced harmonic mean of *Precision* and *Recall* as expressed by Equation 8.

$$\text{F1-Score} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{8}$$

- AUC : depicts the area under the curve, which reflects the ability of the model to distinguish between classes.

In all metrics, *TP* denotes the true positives, *TN* denotes the true negatives, *FP* denotes the false positives and *FN* denotes the false negatives.

IV. Results

Table of figures III shows the evolution of the used metrics over epochs at the training time (200 epochs used). Fig 4 presents the confusion matrix. Fig. 5 presents the performance metrics of our model at test time reaching an Accuracy of 96.67%, a Precision of 97.01%, a Recall of 96.37%, an F1-Score of 96.69% and an AUC of 96.67%.

V. Discussion and comparison with related work

Recent advancements in ransomware detection have introduced many interesting and innovative approaches and datasets to tackle evolving threats. Matthew et al. [27] proposed Pulse, a method utilizing Transformer models to classify functions in assembly language, offering a robust solution for detecting zero-day ransomware. Complementing this, Manabu et al. introduced RanSMAP [28], an open dataset capturing ransomware storage and memory access patterns, enabling the development of deep learning-based detectors. On mobile platforms, Alamgir et al. [29] demonstrated the effectiveness of ensemble machine learning for

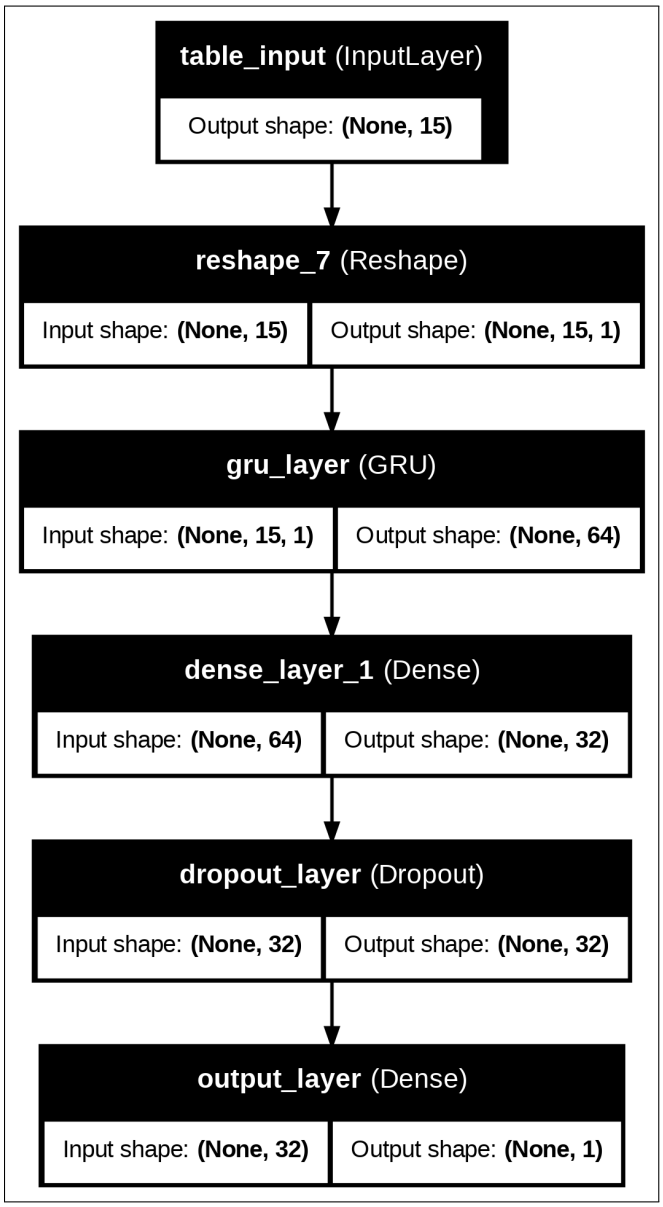
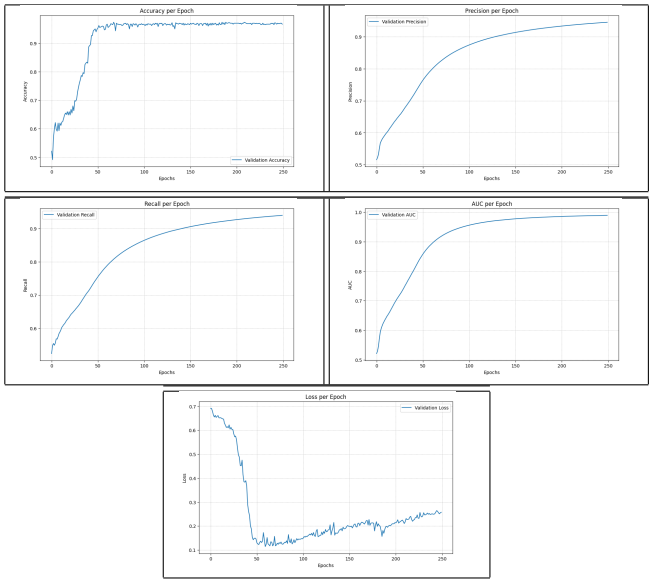


Fig. 3: Model architecture

TABLE III: Model training and metric evolution over epochs



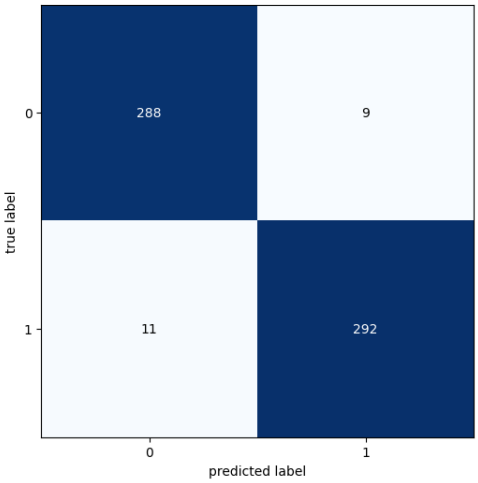


Fig. 4: Confusion matrix (600 entries)

superior Android ransomware detection. Lastly, Jennie et al. [30] explored the use of hardware performance counters on non-virtualized systems, presenting an efficient hardware-level approach for classifying ransomware activity. Our GRU-based tool stays among the competitive models for ransomware detection, distinguished by its simplicity, high performance, and reduced training time. Its streamlined architecture ensures efficient processing while maintaining robust detection capabilities, making it a practical and effective solution compared to more complex alternatives. Our tool, RansFighter, is not designed to rely on a single model. Instead, it is designed to integrate other deep learning models that demonstrate effectiveness in ransomware detection, which are currently being explored and are intended to work together.

VI. Conclusion

This study showcases the effectiveness of our tool based on gated recurrent unit models in detecting ransomware with excellent results observed in test time. The GRUs capability to capture ransomware activities positions it as a good candidate for real-time ransomware detection. However, due to the dynamic nature of ransomware threats and their evolving patterns over time, cybersecurity experts are called to stay vigilant and make perpetual adjustments in their strategies, coupled with a constant state of alertness to fend against potential vulnerabilities in systems across various platforms. In this context, future efforts will focus on enhancing the detection ability to react to threats while integrating data sources such as logs and network traffic to reach greater resilience.

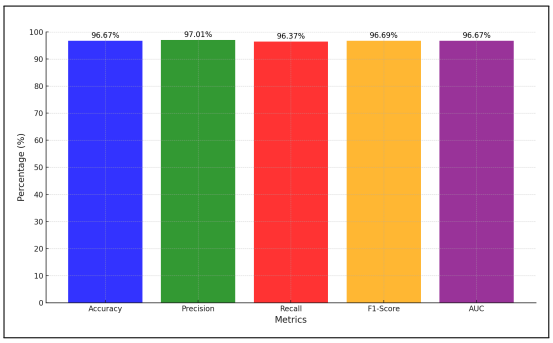


Fig. 5: Model performance

REFERENCES

[1] M. Ryan, *Ransomware Revolution: The Rise of a Prodigious Cyber Threat*, vol. 85 of *Advances in Information Security*. Springer, 2021.

[2] M. U. Rana, M. A. Shah, M. A. A. Naeem, and C. Maple, "Ransomware attacks in cyber-physical systems: Countermeasure of attack vectors through automated web defenses," *IEEE Access*, vol. 12, pp. 149722–149739, 2024.

[3] J. Ferdous, M. R. Islam, A. Mahboubi, and M. Z. Islam, "Ai-based ransomware detection: A comprehensive review," *IEEE Access*, vol. 12, pp. 136666–136695, 2024.

[4] H. L. Bouder, *Symmetric cryptography applied in different contexts: physical attacks and ransomware*. 2023.

[5] M. G. Gaber, M. Ahmed, and H. Janicke, "Zero day ransomware detection with pulse: Function classification with transformer models and assembly language," *Comput. Secur.*, vol. 148, p. 104167, 2025.

[6] G. Kim, S. Kang, S. Baek, K. Kim, and J. Kim, "How to decrypt files encrypted by rhysida ransomware without the attacker's private key," *Comput. Secur.*, vol. 151, p. 104340, 2025.

[7] S. Ali, J. Wang, V. C. M. Leung, and A. Ali, "Decentralized ransomware recovery network: Enhancing resilience and security through secret sharing schemes," in *Proceedings of the 9th International Conference on Internet of Things, Big Data and Security, IoTBDS 2024, Angers, France, April 28-30, 2024* (A. Kobusinska, A. Jacobsson, and V. Chang, eds.), pp. 294–301, SCITEPRESS, 2024.

[8] M. Cen, F. Jiang, X. Qin, Q. Jiang, and R. Doss, "Ransomware early detection: A survey," *Comput. Networks*, vol. 239, p. 110138, 2024.

[9] J. Fattahi, "Machine Learning and Deep Learning Techniques used in Cybersecurity and Digital Forensics: a Review," *arXiv e-prints*, p. arXiv:2501.03250, Dec. 2024. https://ui.adsabs.harvard.edu/abs/2025arXiv250103250F.

[10] J. Fattahi, M. Mejri, and M. Ziadia, "Extreme gradient boosting for cyberpropaganda detection," in *New Trends in Intelligent Software Methodologies, Tools and Techniques - Proceedings of the 20th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques, SoMeT 202, Cancun, Mexico, 21-23 September, 2021* (H. Fujita and H. Pérez-Meana, eds.), vol. 337 of *Frontiers in Artificial Intelligence and Applications*, pp. 99–112, IOS Press, 2021.

[11] J. Fattahi, F. Sghaier, M. Mejri, R. Ghayoula, E. Pricop, and B. E. Lakdher, "Handwritten signature recognition using parallel cnns and transfer learning for forensics," in *10th International Conference on Control, Decision and Information Technologies, CoDIT 2024, Vallette, Malta, July 1-4, 2024*, pp. 1697–1702, IEEE, 2024.

[12] J. Fattahi, B. E. Lakdher, M. Mejri, R. Ghayoula, E. Manai, and M. Ziadia, "Fingfor: a deep learning tool for biometric forensics," in *10th International Conference on Control, Decision and Information Technologies, CoDIT 2024, Vallette, Malta, July 1-4, 2024*, pp. 1667–1672, IEEE, 2024.

[13] E. Manai, M. Mejri, and J. Fattahi, "Helping cnas generate cvss scores faster and more confidently using xai," *Applied Sciences*, vol. 14, no. 20, 2024.

[14] J. Jagadeesan, S. Nandhini, and B. Sathiyaprasad, "Classification of malware for security improvement in iot using heuristic aided adaptive multi-scale and dilated resnext with gated recurrent unit," *Appl. Soft Comput.*, vol. 163, p. 111838, 2024.

[15] W. Feng, Y. Wu, and Y. Fan, "A new method for the prediction of network security situations based on recurrent neural network with gated recurrent unit," *Int. J. Intell. Comput. Cybern.*, vol. 13, no. 1, pp. 25–39, 2020.

[16] J. Fattahi, M. Ziadia, and M. Mejri, "Cyber Racism Detection Using Bidirectional Gated Recurrent Units and Word Embeddings," in *New Trends in Intelligent Software Methodologies, Tools and Techniques - Proceedings of the 20th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques, SoMeT 202, Cancun, Mexico, 21-23 September, 2021* (H. Fujita and H. Pérez-Meana, eds.), vol. 337 of *Frontiers in Artificial Intelligence and Applications*, pp. 155–165, IOS Press, 2021.

[17] L. Arras, J. A. Arjona-Medina, M. Widrich, G. Montavon, M. Gillhofer, K. Müller, S. Hochreiter, and W. Samek, "Explaining and interpreting lstms," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K. Müller, eds.), vol. 11700 of *Lecture Notes in Computer Science*, pp. 211–238, Springer, 2019.

[18] G. V. Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5929–5955, 2020.

[19] J. Fattahi, M. Mejri, M. Ziadia, and R. Ghayoula, "Spamdl: A high performance deep learning spam detector using stanford global vectors and bidirectional long short-term memory neural networks," in *New Trends in Intelligent Software Methodologies, Tools and Techniques - Proceedings of the 21st International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques, SoMeT 2022, Kitakyushu, Japan, 20-22 September, 2022* (H. Fujita, Y. Watanobe, and T. Azumi, eds.), vol. 355 of *Frontiers in Artificial Intelligence and Applications*, pp. 143–162, IOS Press, 2022.

[20] J. Fattahi and M. Mejri, "Damaged fingerprint recognition by convolutional long short-term memory networks for forensic purposes," in *5th IEEE International Conference on Cryptography, Security and Privacy, CSP 2021, Zhuhai, China, January 8-10, 2021*, pp. 193–199, IEEE, 2021.

[21] M. Ibrahim and R. Elhafiz, "Modeling an intrusion detection using recurrent neural networks," *Journal of Engineering Research*, vol. 11, no. 1, p. 100013, 2023.

[22] P. Zavoral, D. Varis, and O. Bojar, "Adversarial testing as a tool for interpretability: Length-based overfitting of elementary functions in transformers," *CoRR*, vol. abs/2410.13802, 2024.

[23] C. Kuo and G. Chen, "Automatic sleep staging based on a hybrid stacked LSTM neural network: Verification using large-scale dataset," *IEEE Access*, vol. 8, pp. 111837–111849, 2020.

[24] R. J. M. Veiga and J. M. F. Rodrigues, "Fine-grained fish classification from small to large datasets with vision transformers," *IEEE Access*, vol. 12, pp. 113642–113660, 2024.

[25] S. Noh, "Analysis of gradient vanishing of rnns and performance comparison," *Inf.*, vol. 12, no. 11, p. 442, 2021.

[26] A. Bensalah, "Ransomware Detection Dataset," Online. https://www.kaggle.com/datasets/amdj3dax/ransomware-detection-data-set. Last accessed: January 6, 2025.

[27] M. G. Gaber, M. Ahmed, and H. Janicke, "Zero day ransomware detection with pulse: Function classification with transformer models and assembly language," *Comput. Secur.*, vol. 148, p. 104167, 2025.

[28] M. Hirano and R. Kobayashi, "Ransmap: Open dataset of ransomware storage and memory access patterns for creating deep learning based ransomware detectors," *Comput. Secur.*, vol. 150, p. 104202, 2025.

[29] M. A. Hossain, T. Hasan, F. Ahmed, S. H. Cheragee, M. H. Kanchan, and M. A. Haque, "Towards superior android ransomware detection: An ensemble machine learning perspective," *Cyber Secur. Appl.*, vol. 3, p. 100076, 2025.

[30] J. Hill, T. O. Walker, J. A. Blanco, R. W. Ives, R. N. Rakvic, and B. Jacob, "Ransomware classification using hardware performance counters on a non-virtualized system," *IEEE Access*, vol. 12, pp. 63865–63884, 2024.