Side-Channel Analysis for Malicious Activity Detection Using Deep Learning Techniques

Devajit Das[1][a], Manash Pratim Lahkar[1][b], Abhijit Gogor[2][c] and Debojit Boro[1][d]
[1]*Department of Computer Science and Engineering, Tezpur University, Tezpur, Assam, India*
[2]*Department of Computer Science and Engineering, Dibrugarh University, Assam, India*

Keywords: Malware Detection, Side-Channel Analysis, Machine Learning, Deep Learning.

Abstract: The continued existence of malicious software constitutes a significant threat to network systems. This necessitates the urgent need for the development of robust detection mechanisms. The rapid development of malware variants frequently makes it challenging for traditional signature-based detection techniques. It has been found that side-channel analysis of physical systems can disclose sensitive data, including the secret key used for encryption, software activities, cryptographic operations, time-based features of the system, and data processing patterns. The side-channel analysis involves obtaining data via unintentional leakage channels from the physical system. The leakage channel may contain electromagnetic radiation, power consumption, timing information, acoustic emissions, cache access patterns, and other side-channel leakage from hardware. In this work, we employ deep-learning techniques with side-channel leakage from the hardware for identifying malicious activities within a system. We demonstrate the effectiveness of our technique by deploying deep neural networks for feature extraction and to recognize complicated correlations within data, specifically using Bidirectional Long Short-Term Memory (BiLSTM) networks. The findings of our experiments demonstrate the accuracy with which recurrent neural networks classify malware instances. We achieved 95.97%, 95.08%, and 92.46% accuracy, recall, and precision, respectively. Furthermore, we carried out real-time malware detection experiments to test our strategy for protecting systems from cyber threats.

1 INTRODUCTION

The widespread existence of malicious software creates a serious risk to network security and integrity in today's digital world. Attackers can employ malware to gain unauthorized access, steal sensitive information, disrupt operations, extort money, and achieve various malicious goals (Tsalis et al., 2019; Dutta et al., 2022; Alsmadi and Alqudah, 2021). A key aspect of cybersecurity involves the detection of malware, as cyberattacks are becoming more frequent and advanced. Due to the limitations of traditional signature-based methods in identifying evolving threats, new and promising solutions for detecting malware have emerged (Islam and Shin, 2023; Abusitta et al., 2021). Advanced deep learning techniques combined with side-channel analysis could be a viable solution for improving security in a physical system (Yuan and Wu, 2021). Side-channel analy- sis is extracting sensitive information, such as secret keys utilized in the encryption process, passwords, and software activities, from unintended channels of physical systems (Hospodar et al., 2011; Xiao et al., 2019). These unintentional channels can comprise the power consumption of the physical system, electromagnetic radiation released by the device itself, and side-channel leakage from the hardware that occurs while the system is in use. Exploiting side channel leakage from the hardware might be a good way to sniff out malware. It allows harmful activities to be identified without needing access to the actual code (Maxwell et al., 2021). It is an unintentional means of leaking information from hardware peripherals such as fan speeds, core temperatures and memory usage. The processing and analyzing of large volumes of complex data has become more and deep learning is

[a] https://orcid.org/0000-0002-4051-6410
[b] https://orcid.org/0009-0008-7244-1653
[c] https://orcid.org/0009-0000-4849-8264
[d] https://orcid.org/0000-0001-5512-0913

Das, D., Lahkar, M. P., Gogor, A. and Boro, D.
Side-Channel Analysis for Malicious Activity Detection Using Deep Learning Techniques.
DOI: 10.5220/0013345500004646
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 1st International Conference on Cognitive & Cloud Computing (IC3Com 2024)*, pages 123-129
ISBN: 978-989-758-739-9
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

increasingly employed in artificial intelligence. Certain neurons in deep neural networks are good feature extractors and capable of grasping the relationships between complex variables. As a result, they might be applied to identify tiny, non-linear patterns in side-channel data that indicate malicious behavior (Perin et al., 2022). It could be possible to more accurately identify and stop harmful behaviour by leveraging deep learning techniques for side-channel analysis (Jin et al., 2020; Song et al., 2019). Consequently, deep learning algorithms with side-channel analysis can discern patterns and irregularities in the physical system which might suggest malware or other pernicious activities are present (Song et al., 2019). Deep learning in combination with feature selection techniques can differentiate malware from normal benign activity accurately (Alomari et al., 2023). The most popular features to consider when analyzing such data seems naturally fit for deep learning as it is able do feature extraction and complex pattern detection automatically.

Our primary goal in this research is to provide the detection of malicious activities on computer systems that are based on side-channel analysis and using advanced deep learning algorithms. In a brief, this work makes the following contributions:

- Preprocessing and training on public dataset with various side-channel data from the hardware to learn model weights for smooth learning.
- Integrating the learned model seamlessly into existing malware detection systems while ensuring compatibility and optimizing resource usage.
- Carried out extensive testing to evaluate the integrated model in the malware detection system, gaining valuable insights into its effectiveness.

The paper is arranged as follows: Section II pro- vides an overview of relevant work in side-channel analysis and malware detection. Section III presents our dataset and methodology detailing the integra- tion of side-channel analysis with deep learning tech- niques. We state the experimental setup in Section IV. In Section V, we describe the techniques and tests we conducted to evaluate our approach's effectiveness in identifying malicious activities. Section VI concludes by outlining possible fields of investigation, discussing future work, and offering conclusions.

This abbreviations in Table 1 provides a quick reference for understanding the symbols used throughout the paper.

2 RELATED WORK

The related work focuses on integrating deep learning techniques with side-channel analysis to infer different software activities and malicious behaviour from physical systems. The literature has a significant number of publications on side-channel analysis for inferring sensitive information that detects malware and other harmful activities utilizing deep learning algorithms.

Table 1: Abbreviations and their Meanings.

Abbreviation	**Description**
RNN	Recurrent Neural Network
CAN	Controller Area Network
CNN	Convolutional Neural Network
MLP	Multi-Layer Perceptron
IDS	Intrusion Detection System
CART	Classification and Regression Trees
F1 Score	Harmonic mean of precision and recall
HWiNFO64	System information capture tool
ED BiLSTM	Encoder Decoder Bidirectional LSTM
HFSS	High-Frequency Structure Simulator
EMSCA	Electromagnetic Side-channel Attack
D	Dataset
Preprocess(D)	Preprocessing of dataset
D_{train}	Training dataset
D_{test}	Testing dataset
M_i	Model with different parameters
L	Loss function
$Metrics(M_i, D_{test})$	Performance metrics
M_{best}	Best performing model
S	System
Inject(S, Malware)	Injection of malware into system
HWiNFO64	Application for information capture
Detect(M_{best}, HWiNFO64)	Detection of malware using best model
$Time_{detection}$ detection	Time from malware injection to

Rastogi et al. (Rastogi and Kavun, 2021) explored deep learning methods for side-channel analysis on AES datasets from both hardware and software platforms, including MLP, CNN, and RNN. Their study focused on evaluating the performance of these models for secret key recovery and optimizing attack performance when it comes to computation time and SCA efficiency. They found CNNs to be more effective for analyzing software implementations, while MLPs were better suited for hardware implementations of cryptographic algorithms. Haider et al. (Khan et al., 2019) proposed distinct electromagnetic side-channel signals monitoring solutions for crucial electronic and cyber-physical systems. They trained a neural network model using electromagnetic emissions from an uninfected device to produced normal activity patterns. They monitored the target device's electromagnetic emanations remotely to detect deviations indicating anomalous behavior. The system successfully detected DDoS, and ransomware attacks with high precision in practical evaluations on different devices, showcasing the adaptability of the framework. Xun et al. (Xun et al., 2023) created an IDS utilizing vehicle voltage signals to protect the security of the CAN bus in intelligent connected vehicles. They utilized a hybrid FeatureBagging-CNN model to detect malicious intrusion without developer documentation. Gao et al. (Gao et al., 2024) designed a novel hardware trojan detection model converting one-dimensional signals into two-dimensional data to utilize frequency information in time-series signals. They enhanced hardware trojan detection using an improved ConvNeXt network, effectively distinguishing and classifying different types of trojan while improving accuracy, recall, F1-score, and precision values. Maxwell et al. (Maxwell et al., 2021) demonstrated the effectiveness of side-channel information in detecting malware presence on computing platforms. They achieved malware detection accuracy exceeding 91% with a basic MLP model, 95.83% with a CNN model, and 90.91% with an RNN model. Stergiopoulos et al. (Stergiopoulos et al., 2018) introduced a machine learning-based network traffic monitoring system that analyses TCP/IP packet side-channel characteristics. Their solution achieves a true positive detection rate of about 94% for differentiating between harmful and regular traffic across a wide spectrum of threats, successfully separating legitimate traffic from various forms of malicious activities. It is noticeable that on full-scale mixed forms of harmful data, machine learning methods such as CART and KNN obtained detection rates of 94.2% and 93.4%, respectively. Sayakkara et al. (Sayakkara et al., 2020) presented a system for activity identification in forensic investigations that uses several filtering techniques and machine learning algorithms to locate leaky frequency channels from high-dimensional electromagnetic data. The method narrows 20,000 channels to 81 for real-time analysis. The accuracy of activity prediction ranged from 0.9047 to 0.9395. Ghosh et al. (Ghosh et al., 2022) proposed an An- sys HFSS-based simulation framework for EM anal- ysis of an integrated on-chip sensor to identify EM- SCA and FIA (fault injection attacks). A simple ap- proach for detecting incoming H-probes is also de- scribed, which involves taking the absolute average of the induced voltage for each encryption. Sayakkara et al. (Sayakkara et al., 2019) used deep neural network-based classifiers to classify cryptographic encryptions on a Raspberry Pi 4 with above 95% accuracy, utilizing electromagnetic side-channel analysis.

This related work demonstrates a clear emphasis on integrating side-channel analysis combined with deep learning techniques to identify various software activities and detect malicious activity, including malware. Several studies explore electromagnetic side-channel analysis, deep neural networks, corporating deep learning for side-channel analysis has several challenges, including data diversity, system integration, and adversarial robustness. In this work, we use multiple side-channel hardware sources to detect malware.

3 DATASET & METHODOLOGY

3.1 Dataset

The dataset utilized in this study was obtained from IEEEDataPort (Maxwell et al., 2020), comprising data collected by the HWiNFO real-time monitoring tool (hwi,). HWiNFO offers detailed hardware performance metrics, such as memory load, core frequencies, and usage percentages. Two samples were taken every second during the data recording and then saved in CSV format. This dataset includes 57 samples: 29 malware and 28 benign. Each sample consists of an eight-minute time-series output generated from HWiNFO captures. Samples were collected from six distinct hardware PCs running the Windows operating system on VMWare/VirtualBox virtual machines. To ensure diversity in the dataset, malware samples were collected randomly at intervals of 90, 120, or 150 seconds. The benign software operations simulated regular system behavior before executing the malware. Samples were labeled to denote machine status and indicate benign or malicious activity. Each file follows a naming convention facilitating future research such as investigating model transferability. Malware categories include ransomware, worms, trojan backdoors, spyware, viruses, and rootkits. Benign samples mimic routine software activities like web browsing, document editing, gaming, and system maintenance. The dataset comprises two folders: ”data_raw” containing original captures and ”data_blend” with standardized features derived from an inner join across all samples. The latter ensures uniform feature representation facilitating classification tasks with 122 common features and six label-type features.

3.2 Methodology

We initially obtained a dataset[1] called side-channel

[1]https://ieee-dataport.org/documents/side-channel-data-malware-intrusion-detection

data for malware detection from IEEEDataPort. Let this dataset be D and it is subjected to preprocessing techniques including data cleaning and column and other machine-learning algorithms to classify cryptographic encryptions, recover secret keys, and identify anomalies in system behavior. However, in-

removal denoted as Preprocess(D). Two sets of preprocessed data were created: training D_{train} and testing D_{test} sets. Various models M_i were trained using D_{train}, where i represents different architectures and parameters such as BiLSTM and CNN. Model parameters are optimized during training to minimize a loss function L, defined as $L(M_i, D_{\text{train}})$. After training, the models were tested on D_{test} to see how effectively they performed in terms of accuracy, precision, recall, and F1-score, which are expressed as Metrics(M_i, D_{test}). A comparative analysis was carried out to determine the model M_{best} with the highest performance metrics. Subsequently, a system S was developed based on the specifications consistent with the dataset creation. The system was then subjected to malware injection. We acquired the malware code from "theZoo repository" on GitHub[2](Nativ,). This is denoted as Inject(S, Malware), and dynamic information from the HWiNFO64 application was fed into the trained model M_{best} in real-time, represented as Detect(M_{best}, HWiNFO64), to detect malware presence. The time elapsed from malware injection to detection by the model M_{best} was recorded for further analysis. It is denoted as Time$_{\text{detection}}$. This methodology is depicted in Figure 1 and Algorithm 1.

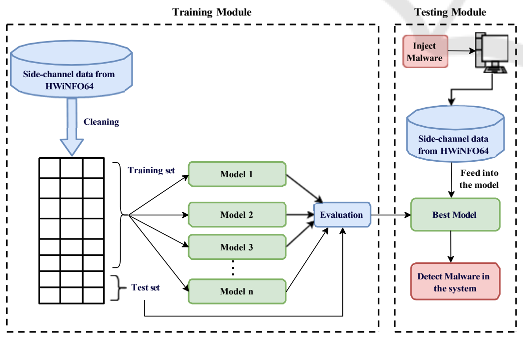

Figure 1: Methodology.

4 EXPERIMENTAL SETUP

The experimental setup consists of various hardware and software components to assess the efficacy of our suggested technique for identifying malicious activity in computer systems. Below are the details of the system that was employed in the experiments:

- **Operating System**: Windows

Algorithm 1 Malware Detection Methodology

```
1:  Step 1: Obtain dataset $D$ from IEEEDataPort
2:  Step 2: Preprocess dataset: $D_{\text{clean}}$ ←
      Preprocess($D$)
3:  Step 3: Divide the dataset into sets for testing
      and training:
      $D_{\text{train}}, D_{\text{test}} \leftarrow$ Split($D_{\text{clean}}$)
4:  Step 4: For each model $M_i$
5:      Train model: $M_i \leftarrow$ Train($M_i$,
$D_{\text{train}}$) 6:      Evaluate model:
Metrics($M_i$, $D_{\text{test}}$) 7:  Step 5: Select best
model:
      $M_{\text{best}} \leftarrow$ SelectBestModel($M_1, M_2, ..., M_n$)
8:  Step 6: Develop system: $S \leftarrow$ DevelopSystem()
9:  Step 7: Inject malware into system:
      Inject($S$, Malware)
10: Step 8: Monitor system using HWiNFO64:
      Monitor($S$, HWiNFO64)
11: Step 9: Detect malware using best model:
      Detect($M_{\text{best}}$, HWiNFO64)
12: Step 10: Record time to detection:
      Time$_{\text{detection}}$
```

- **Processor**: Intel Core i7-9700 @ 3.00GHz
- **Storage**: 512GB SSD
- **Memory**: 32GB DDR4 RAM
- **Graphics Processing Unit (GPU)**: NVIDIA GeForce GTX 1080

Table 2: Specifications of the Virtual Machine Setup

Component	Specifications
Virtualization Software	Oracle VM VirtualBox
VM Operating System	Ubuntu 20.04 LTS
VM Resources	**Processor**: 4 virtual CPUs **Memory**: 16GB **Storage**: 100GB virtual disk **GPU**: Integrated with host GPU for accelerated performance

We used a virtual machine (VM) configuration with specifications listed in Table 2 to ensure isolated experiments and realistic environment simulation. The development, training, and assessment of our deep learning models were facilitated by the following software tools and libraries, as indicated in Table 3:

[2] https://github.com/ytisf/theZoo

5 EXPERIMENTS AND RESULTS

This section outlines the experimental procedures used to assess the effectiveness of our proposed method for detecting malicious activity on computer systems. We present an extensive description of the model architecture and parameter setup that we employed in this research. The dataset utilized in our research was sourced from the IEEEDataPort repository ensuring accessibility and transparency. To prepare for testing and training, we divided the dataset into different subgroups. We used a stratified splitting strategy to maintain a balanced distribution of normal and anomalous instances. We conducted experiments using multiple models, such as BiLSTM, CNNs, and RNNs, to determine how effectively different deep learning algorithms performed in identifying malicious activities. Our main goal was to determine the model architecture that could best capture complex patterns indicative of malicious behavior.

Table 3: Software Tools and Libraries

Software Tools	Description
Python 3.8	Primary programming language used for model implementation. It is widely used in machine learning, data science, and general-purpose programming.
TensorFlow 2.4	Deep learning framework employed for constructing and training BiLSTM, CNN, and RNN models. It supports GPU acceleration for faster computation.
Keras	High-level neural network API running on TensorFlow, utilized for defining and training model architectures. It simplifies the model-building process.
NVIDIA CUDA 11.1	Parallel computing platform and programming model that leverages GPU capabilities for model training. It helps in utilizing hardware acceleration for deep learning models.
cuDNN 8.0	GPU-accelerated library for deep neural networks, integrated with TensorFlow to enhance performance. It accelerates deep learning operations.
HWiNFO	Real-time monitoring tool used to capture detailed system information during experiments. It provides insights into hardware performance and usage.

Among the architectures evaluated the BiLSTM model emerged as the most promising choice because of its inherent ability to capture temporal dependencies and bidirectional context information. This architecture proved particularly adept at discerning subtle anomalies within sequential data that make it appropriate for our detection task. The BiLSTM model was configured with 256 LSTM units to ensure sufficient representational capacity for capturing the intricacy of the input data. To improve generalised performance and reduce overfitting, 0.2 dropout layers were added. A total of 200 epochs and a batch size of 128 samples per iteration were used to train the BiLSTM model. This training regimen facilitated gradual learning and convergence toward an optimal solution.

Table 4: Comparison of Model Results with baseline

Model	Accuracy	Recall	Precision	F1 Score
BiLSTM	**95.97%**	**95.08%**	**92.46%**	**93.75%**
RNN	93.49%	93%	94%	93%
MLP	50.9%	51.2%	49.45%	50.17%
ED BiLSTM	93.78%	96.32%	91.47%	93.83%
Maxwell	95.83%	93.9%	91.84%	92.68%

Table 4 depicts the comparison of model results with different models, and it indicates that the BiL- STM model achieved the highest accuracy of 95.97%. The training and testing accuracy of the model is depicted in Figure 2. Having a recall rate of 95.08%, the model confirmed its ability to classify the positive cases properly. It means that among all real positive instances, the model was able to identify 95.08%. Moreover, the model had a precision of 92.46% and was able to identify 92.46% of all predicted positive instances. The F1 score of 93.75% shows a harmonic mean between precision and recall which gives reasonable performance for both of the indexes. In comparison, the RNN model had an accuracy value of 93.49% along with a recall rate 93% and precision of 94%.Even though this one is lower accurate that the BiLSTM model, the RNN model displayed more balanced performance in recall and precision and had an F1 score of 93%. However, the MLP shows much lower results compared to the recurrent models by all measures. As most of the instances were inaccurately classified, the accuracy percentage was 50.9%, and both recall and precision were compromised at 51.2% and 49.45%, respectively. This resulted in a low F1 score of 50.19%. Finally, the ED BiLSTM model performed at 93.78% accuracy, with an impressive recall of 96.32% and precision of 91.47%. The results more effectively represent the ability of the model to locate positive instances in the data set rather than minimize false positives. As a result, it obtained an F1 score of 93.83% maintaining a balanced performance between recall and precision. Overall, the evaluation results indicate the superiority of recurrent models particularly BiLSTM and ED BiLSTM in accurately classifying instances compared to the MLP model. These findings emphasize the potential benefits of RNNs.
Once our models were trained, and we identified the best performing model, BiLSTM, we proceeded to test its capabilities of real-time malware detec- tion. We wanted to do it under realistic conditions, so we

created a system exactly identical to the sys- tems which were used to create the dataset, including the non-malware system and running services. After that we mimicked the infection by running the mal- ware on the system and using the model towards it. We also obtained detailed information about the sys- tem, i.g. hardware and software components used, via HWiNFO64 and also fed this into our trained system to provide it with context. The model detected that a malware was run on the system in only 56 seconds. As a result, we would like to note that our trained model has practical use in protecting systems from malicious software threats. In addition, our model outperformed Maxwell's work (Maxwell et al., 2021), which stated an accuracy of 95.83%. In contrast, our model performed slightly better at identifying malicious activity, with an accuracy of 95.97%.

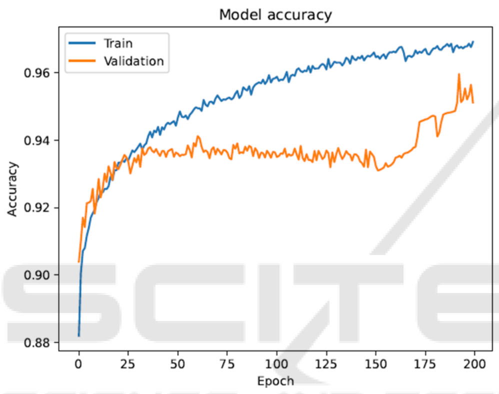

Figure 2: Training and testing accuracy.

6 CONCLUSIONS AND FUTURE WORK

In conclusion, our experimental methodology involved evaluating various deep learning architectures for the detection of malicious activity within computer systems. Through rigorous experimentation and model evaluation, we identified the BiLSTM model as the most effective in accurately classifying instances of malware with a high accuracy of 95.97%, recall rate of 95.08%, precision of 92.46%, and F1 score of 93.75%. The results obtained underscore the significance of utilizing recurrent neural networks particularly BiLSTM in handling sequential data and capturing complex patterns indicative of malicious behavior. Moreover, our real-time malware detection experiment further validates the practical utility of the BiLSTM model in safeguarding systems against cyber threats.

In the future, our goal is to investigate more characteristics and data sources to enhance the function- ality and resilience of our detection system. This in- cludes incorporating dynamic behavioral analysis and network traffic data to improve detection accuracy and resilience against evolving malware variants. Ad- ditionally, we plan to investigate techniques for enhancing the real-time detection speed and scalability of our model ensuring its effectiveness in large-scale deployment scenarios.

ACKNOWLEDGMENT

The authors are so grateful to the Department of Computer Science and Engineering at Tezpur University for providing us with the laboratory resources required to carry out our study.

REFERENCES

HWiNFO - free system information, monitoring and diagnostics. HWiNFO. April, 2020.

Abusitta, A., Li, M. Q., and Fung, B. C. (2021). Malware classification and composition analysis: A survey of recent developments. *Journal of Information Security and Applications*, 59:102828.

Alomari, E. S., Nuiaa, R. R., Alyasseri, Z. A. A., Mohammed, H. J., Sani, N. S., Esa, M. I., and Musawi, B. A. (2023). Malware detection using deep learn- ing and correlation-based feature selection. *Symmetry*, 15(1):123.

Alsmadi, T. and Alqudah, N. (2021). A survey on malware detection techniques. In *2021 international conference on information technology (ICIT)*, pages 371–376. IEEE.

Dutta, N., Jadav, N., Tanwar, S., Sarma, H. K. D., Pricop, E., Dutta, N., Jadav, N., Tanwar, S., Sarma, H. K. D., and Pricop, E. (2022). Introduction to malware analysis. *Cyber Security: Issues and Current Trends*, pages 129–141.

Gao, Y., Su, J., Li, J., Wang, S., and Li, C. (2024). A neural network framework based on convnext for side-channel hardware trojan detection. *ETRI Journal*.

Ghosh, A., Nath, M., Das, D., Ghosh, S., and Sen, S. (2022). Electromagnetic analysis of integrated on- chip sensing loop for side-channel and fault-injection attack detection. *IEEE Microwave and Wireless Com- ponents Letters*, 32(6):784–787.

Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., and Vandewalle, J. (2011). Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293–302.

Islam, N. and Shin, S. (2023). Review of deep learning-based malware detection for android and windows system. *arXiv preprint arXiv:2307.01494*.

Jin, S., Kim, S., Kim, H., and Hong, S. (2020). Recent advances in deep learning-based side-channel analysis. *Etri Journal*, 42(2):292–304.

Khan, H. A., Sehatbakhsh, N., Nguyen, L. N., Prvulovic, M., and Zajic´, A. (2019). Malware detection in embedded systems using neural network model for electromagnetic side-channel signals. *Journal of Hardware and Systems Security*, 3:305–318.

Maxwell, P., Niblick, D., and Ruiz, D. (2020). Side channel data for malware intrusion detection. IEEE DataPort. Nov. 25, 2020.

Maxwell, P., Niblick, D., and Ruiz, D. C. (2021). Using side channel information and artificial intelligence for malware detection. In *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 408–413. IEEE.

Nativ, Y. T. Thezoo. April 2024. Perin, G., Wu, L., and Picek, S. (2022). Explor- ing feature selection scenarios for deep learning- based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):828–861.

Rastogi, T. S. and Kavun, E. B. (2021). Deep learning techniques for side-channel analysis on aes datasets collected from hardware and software platforms. In *International Conference on Embedded Computer Systems*, pages 300–316. Springer.

Sayakkara, A., Le-Khac, N.-A., and Scanlon, M. (2019). Leveraging electromagnetic side-channel analysis for the investigation of IoT devices. *Digital Investigation*, 29:S94–S103.

Sayakkara, A., Miralles-Pechua´n, L., Le-Khac, N.-A., and Scanlon, M. (2020). Cutting through the emissions: feature selection from electromagnetic side-channel data for activity detection. *Forensic Science International: Digital Investigation*, 32:300927.

Song, S., Chen, K., and Zhang, Y. (2019). Overview of side channel cipher analysis based on deep learning. In *Journal of Physics: Conference Series*, volume 1213, page 022013. IOP Publishing.

Stergiopoulos, G., Talavari, A., Bitsikas, E., and Gritza- lis, D. (2018). Automatic detection of various mali- cious traffic using side channel features on tcp pack- ets. In *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*, pages 346–362. Springer.

Tsalis, N., Vasilellis, E., Mentzelioti, D., and Apostolopoulos, T. (2019). A taxonomy of side channel attacks on critical infrastructures and relevant systems. *Critical Infrastructure Security and Resilience: Theories, Methods, Tools and Technologies*, pages 283–313.

Xiao, Y., Jia, Y., Liu, C., Cheng, X., Yu, J., and Lv, W. (2019). Edge computing security: State of the art and challenges. *Proceedings of the IEEE*, 107(8):1608–1631.

Xun, Y., Deng, Z., Liu, J., and Zhao, Y. (2023). Side channel analysis: A novel intrusion detection system based on vehicle voltage signals. *IEEE Transactions on Vehicular Technology*, 72(6):7240–7250.

Yuan, S. and Wu, X. (2021). Deep learning for insider threat detection: Review, challenges and opportunities. *Computers & Security*, 104:102221.