Towards Efficient Malware Detection: A Lightweight Machine Learning Framework for File Analysis

Ethan Bryd
Department of Information Technology
Kennesaw State University
Kennesaw, USA
ebryd13@student.kennesaw.edu

Caleb Chitty
Department of Information Technology
Kennesaw State University
Kennesaw, USA
cchitty@student.kennesaw.edu

Kyle Loftus
Department of Information Technology
Kennesaw State University
Kennesaw, USA
kloftus@student.kennesaw.edu

Utsav Patel
Department of Information Technology
Kennesaw State University
Kennesaw, USA
upatel23@student.kennesaw.edu

Samuel Umar
Department of Information Technology
Kennesaw State University
Kennesaw, USA
sumar2@student.kennesaw.edu

***Abstract*— In our world today, the importance of security cannot be overstated, with the ever-recurring and advancing threat of vulnerabilities. Malware, being one of those issues, has continued to proliferate and fester in its volume and complexity, hence making regular or traditional signature-based defenses rather inadequate. In response to this dilemma, this study presents a lightweight machine learning solution that is designed to find malware by looking specifically at static-based file features instead of relying on the signature. For the purpose of this study, we made use of two unique datasets: the BODMAS dataset covering diverse malware families and a ransomware-focused Portable Executable (PE) dataset. We used these to train and assess some classification algorithms to ascertain which of them can make us achieve a high detection rating while also achieving low resource consumption. This solution can successfully identify malware but also classify it based on its signature family, such as trojan, worms, or ransomware. The results of the scans prove greatly that lightweight machine-driven scanners can highlight efficient and scalable protection against ever-evolving threats without the burden of heavy systems that may not work as well.**

Keywords*— *Malware Detection, Machine Learning, Portable Executable (PE) & Lightweight Scanner.

I. INTRODUCTION

Malware or malicious software is a software type that is designed with the deliberate purpose of damaging, disrupting, or gaining illegal access to a system or network. In today's ever-developing cyber world, malware is an issue that has grown at a rapid rate. Different malware families have evolved from basic viruses into very high-level software that is capable of attacking major infrastructure and stealing sensitive data, which can thereby cause massive financial losses. Based on industry reports, it is currently estimated that new types of malware samples are identified in the millions on a yearly basis, and thousands of variants are identified daily. While the attacks from malware can be rather complex, attackers have found ways to deploy advanced methods like obfuscation, fileless execution, and many other techniques that can evade traditional defense mechanisms. Famous attacks like the WannaCry ransomware attack showed how a single strain of malware can propagate worldwide within just hours of the start of the attack. The scale of a single attack just showed the importance and need for an adaptive solution that can match the trend of evolution of these issues.

A. Why Traditional Defenses Fail

When we bring up traditional defenses, we are focusing on solutions that work on detecting malware using signature-based antivirus programs. They form the basis of the backbone of most major security solutions. In most cases, how it works is the solution compares files to a database of known malware signatures that are compiled by the platform. While this works amazingly well when it comes to popular malware types and families, it is more of a reactive solution rather than a security-first focused solution. If an attack is a zero-day-based attack or has multiple modifications that make it complex, it would leave the system vulnerable to attack. Furthermore, the solutions must be constantly updated in order for them to be useful and keep up with the modern-day attacks. Because of this constraint, it would require an inflated cost of memory and the possibility of processing overhead issues, therefore making the solutions not always feasible depending on the systems that need to be protected. While scalability is a genuine issue, the main problem remains accuracy when it comes to real-time scenarios.

B. Motivation for Lightweight ML Scanners

Because of these limitations with traditional solutions, security professionals and experts are now turning their attention towards machine learning for malware detection. The solution, unlike the signature-based models, is based on file analytics. The machine learning model is focused on analyzing specific file characteristics like Portable Executable headers, API imports, and entropy levels to distinguish genuine files from malicious files. This approach of this solution is dynamic and hence allows the detection of variants of malware that may not have been previously seen. The detection is done by identifying the underlying operational and behavioral patterns of malware and using that

259-4-1967-7594-9/18/$50.00 ©2025 IEEE

information to flag and stop malware. Unfortunately, while this approach has high yields and accuracy, there is a systematic problem with the solution, which is that many machine-based solutions require a lot of processing power due to the deep computational requirements of most of them.

This creates a problem but also the need for a solution that is light-weight and can solve the problems of traditional defenses, which is: a solution that has a high level of accuracy and is not reactive; And finally, a solution that does not use overwhelming resources to detect malware of different types.

C. Research Question and Objectives

The goal of this study is to answer a research question that many professionals have asked to combat this problem: "How can we create a lightweight machine learning solution that accomplishes fast and precise malware detection, minimizing computational cost?" To answer this, we have created a set of objectives that would assist us in answering the question:

- Analyze publicly available datasets that have distinct types of malware: the Ransomware PE dataset and the BODMAS dataset.
- To apply and evaluate 8 machine learning algorithms. We would be using file features for classification.
- To compare algorithm performance, based on accuracy-based modifiers, to determine the most effective model.
- To demonstrate the possibility of creating a scanner that operates efficiently while maintaining its effectiveness.

By accomplishing these objectives, the goal of this research is to provide a practical solution that can help contribute towards the next generation of malware defense systems that would go beyond the traditional solutions.

II. LITERATURE REVIEW

Malware has become the newest tool and threat in the cyberworld. Still, malware itself has its own arsenal of different categories, which each affect the digital world differently but with only one end goal of damage to the cyberworld. There are so many types of malware that exist currently. There are viruses, worms, Trojan horses, and even more types of malware. Still, out of the many, there are three malware that have the biggest impact on cyberspace. The first one will be ransomware, which is a commonly known malware that, when infected in the system, will lock out all access till a ransom is given for a decryption key. Second would be worms, which are self-replicating malware that can infect systems without a human source to help create more. Lastly is a Trojan horse, which is a malware that can stay dormant once infected until it is needed for action. Many works have been created to both analyze and provide solutions to detect the malware that are going to be discussed in this section.

A. Ransomware

Ransomware has become one of the most well-known and popular malware. The malware can be used for any purpose, making it the perfect all-purpose tool for information. The malware uses multiple methods to infect a target system, from using other malware like phishing to a simple hacking of the system by the attacker. Once infected, the malware encrypts all data to be locked behind a password or key. A simple demand is made, mostly money, which, after being fulfilled, the attacker will present the decryption key to the target to recover the files. Criminal affiliations and cyber gangs capitalize on this malware by using ransomware-as-a-service offerings, which further enhances the danger they pose to the digital era [1]. Still, ransomware itself is not unique in pattern, as each attack can be considered a different pattern, as they exploit new weaknesses in the systems. Some known ransomware attacks are WannaCry, Defray, GandCrab, and nRansom, while there are even more attacks that might not be well known [2]. After COVID-19 began, the use of ransomware increased as the use of cryptocurrencies and people shifted to online work [3].

Vehabovic et al.'s paper uses federated learning with imbalanced ransomware datasets to showcase the application of the learning to ransomware analysis [1]. Determining if the learning is effective against a set of imbalanced data, a central server linked to multiple client servers to used as a way to compile the data. The literature determines that if the data is balanced and precise, then federated learning will give a proper output with efficient results [1]. The imbalance of data is not something new currently, as between changing algorithms and new attack patterns based on new technology, there will be gaps in data. The connection between the client to central servers has to be very precise and straightforward to help provide a balanced set of data to analyze for ransomware [1]. This paper will use different classifying methods to determine both the likelihood that ransomware is detected and which classifier can provide the highest accuracy with maximum precision. The datasets used in the paper contain both benign and malicious samples; hence, the paper will also present a contrasting viewpoint of data. Both papers contain the use of datasets, which will contain an imbalance of data, meaning the results might be on a similar level with similar conclusions.

Another paper created by Chen et al. uses more of a practical approach. The paper has an environment setup where seven different ransomware executables are injected into the environment [2]. Logs of uninfected systems and ransomware behavior are made to help analyze behavior in a normal environment versus a ransomware environment. The logs are created by the Cuckoo sandbox tool, after which the features of the logs are analyzed using three classification algorithms: TF-IDF, Fisher's LDA, and Extremely Randomized Trees [2]. This paper will use different classifying methods to determine both the likelihood that ransomware is detected and which classifier can provide the highest accuracy with maximum precision. This paper and Chen et al.'s paper are similar in the fact of both will use classification algorithms to analyze the datasets. Chen et al.'s paper creates the data by first running environmental tests, while this paper uses datasets previously compiled by other sources to analyze the features. This paper also uses more classification algorithms, hence the determination of a high accuracy of detection from another classification is presented. Chen et al.'s paper also created the environment, meaning the methodology might only stick to certain aspects instead of having a broad range of results.

B. Trojan Horses

Trojan horse malware can be considered the ninja of all malware. Malware can come in any form, from physical to digital. Once a Trojan horse has infected a target system, it can act like any legitimate software, deceiving users and gaining unauthorized access to multiple devices on the shared network [4]. AI has been mentioned to be a Trojan horse as it adapts with new data as new environments are created to analyze the new trends, removing the need for human decision factors [5]. AI has become a tool everyone uses to help them complete tasks, making it a very good tool for malicious acts to be committed at a faster speed. Hence, business leaders and the government view AI as an extinction-level threat to humans [5]. Trojan horses can also be physical, meaning a physical tool can be used as a way to deliver the malware. A simple motherboard chip can be a Trojan horse disguised as a normal chip to act as an undetected backdoor into the system, and then the network connected to that system, which provides even more systems. Hardware Trojan horses are designed to serve malicious purposes to be used for disclosing confidential information or allow a way to transmit unencrypted data on an unprotected channel [6]. Overall, Trojan horses can be dangerous as their ability to camouflage as a normal item can lead to many conflicts.

Mughaid et al.'s paper proposes the detection of Trojan horses on the Internet of Things through the comparison of machine learning methods. The paper dataset is first put through a series of preparation by deleting duplicates, checking for missing values which needs to be fixed, data normalization by ensuring the numbers are within a certain range, deciding which features to select to run the dataset inside, and splitting the data up in this case for the paper was 70% training and 30% testing [4]. After the data has been properly checked and formatted, the authors discuss them using Naïve Bayes, J48 tree, and IBK as the classification algorithms [4]. Mughaid et al.'s paper and this paper have similar approaches, as both will be using different classification algorithms to analyze the dataset. The difference in papers is the preparation of datasets and the variety of classification algorithms. This paper will use using few more classifications aside from Naïve Bayes, J48, and IBK to analyze detection accuracies. This paper might take some knowledge from Mughaid et al.'s paper to help refine any inaccuracies in the malware detection methods and classifications.

Barros et al.'s paper analyzes the effects of an approximate parallel prefix adder against hardware Trojan horse injection. The paper theorizes that a computer with ASIC or Application Specific integrated circuit designs becomes more complex, so will the risk of tampering through a hardware trojan horse [6]. Hence, the paper wants to introduce AxPPA as it can help exploit the error tolerance of certain applications to catch any hardware trojan horses integrated into the circuit [6]. While Barros et al.'s paper analyzed the hardware-based malware, this paper uses datasets already compiled from various sources to analyze the accuracy of identifying the malware. The dataset used in this paper might contain data on hardware trojan horses with their digital counterpart, but there is no way to see which one is which, as the dataset also contains other malware data. The classification algorithm might divide them into classes, but the focus of this paper is overall detection over detection based on subtypes of the malware.

C. Worms

Worms are malware that might be less known than ransomware or Trojan horses, or even other types of malware. Worms can be confused with viruses, as both have similar functions. A worm is a malicious software designed to replicate and spread randomly across computer networks [7]. This malware, unlike a virus, does not require a host as it will self-replicate to spread across system to systems [7]. There have been many types of worms that have existed in the cyberworld. Worms that were injected into a system or some previous program can be turned into a worm for the infection of systems. Wajam worm began with a software called Wajam, which was a search engine enhancer that helped search engines like Google have more results with the help of data extracted from social media connections [8]. The product eventually turned from having social connections to just pure ads/spyware. A Chinese company took over the product and exploited it as a program whose ads will generate on its own while gaining access to the systems. Morris Worm was a public event that shaped the structures and understanding of the internet itself [9]. The worm targeted the institutions that created the internet itself, causing a great amount of damage [9].

De Carnavalet & Mannan paper discusses the essence of the Wajam Worm from its history, concept, and practical testing of the worm. The authors compile 52 samples to use as testing matrices [8]. The author's paper also discusses the use of a VMware environment to test the injections of the Wajam worm in different formats [8]. The rest of the paper goes over the effects of the installer and how it can cause more issues. This paper only sticks to analyzing the dataset for malware through Machine Learning. Hence, both papers are on completely different sides of a research paper. This paper will discuss the accuracy of malware detection, which includes worms, using different classification algorithms, while the other paper is more of an analysis and information on the Wajam worm and its effects on the cyberworld.

Shemakov's paper analyzes the Morris Worm and the major incident that led to the emergence of digital insecurities. The author mentions how the Morris code disabled more than 6,000 government and university computers [9]. The worm was able to sneak past multiple security measures to gain access to different official business channels, which contained very confidential information [9]. The paper concludes on how the Morris Worm incident became the catalyst for multiple laws to be placed on the concepts of the internet before the internet itself gained recognition [9]. This paper is more of a machine learning analysis of Shemakov's paper on understanding the aftermath of the Morris worm. This paper will use classification algorithms to look over malware, which includes worms, to see how accurately worms can be detected based on the classification algorithm. The author's paper, being more of an analysis and history paper, does not show the full aspects of worms.

III. Methodology & Algorithm

This study followed a rigorous and structured methodology, which was designed to test the effectiveness of lightweight machine learning when it comes to static malware detection. This methodology consists of 4 main phases: abbreviations, selected datasets and pre-processing model training, and evaluation metrics.

A. Abbreviations

- PE: Portable Executable
- BODMAS: Blue hexagon Open Dataset for Malware Analysis
- WEKA: Waikato Environment for Knowledge Analysis
- RIPPER: Repeated Incremental Pruning to Produce Error Reduction

B. Selected Datasets

The two datasets that were chosen were publicly available and were chosen to ensure diverse data.

- Ransomware Portable Executable (PE) Dataset - A dataset focused on ransomware that contains benign and malicious PE files, which provides a well-balanced set for classifying binaries between safe and unsafe (malicious) files.
- BODMAS Dataset - A large dataset that covers multiple different malware families, going beyond that of just ransomware, allowing for multiclass classification across numerous malicious samples such as trojans, worms, ransomware, and even safe files.

C. Data Preprocessing

Before training, it was noted that the BODMAS dataset was quite large, and the Ransomware PE dataset was much smaller and precise. Therefore, changes were made to the BODMAS dataset to prevent any skewing or bias with a more focused dataset. The following update was applied to the BODMAS dataset:

- Dataset Reduction

  To manage and improve overall classification performance, a feature selection within WEKA 3.8.6 was applied to the BODMAS dataset using the *InfoGainAttributeEval* evaluator with the *Ranker* search method. This made it so that each attribute was given a score based on its information gain in relation to the class label, and then ranked in descending order of importance. The top-ranking features were then chosen to form a reduced dataset. This reduced dataset was chosen to train and test the machine learning model with the different algorithms. This approach made sure that within a large dataset, only the most informative attributes were kept to enhance classifiers, while also minimizing any skewed or redundant data.

D. Model Training

To ensure consistency and to support comparison of results, the model training process was conducted by two researchers independently. Both of the selected datasets (BODMAS & Ransomware PE) were tested against a wide variety of classifiers to ensure a wide and diverse coverage. All model training and testing were conducted using the WEKA 3.8.6 machine learning toolkit.

The classifiers that were applied to both datasets were chosen to reflect a wide range of learning strategies encompassing decision trees, probability models, and ensemble methods. By testing both datasets against the same classifiers, a direct comparison could be drawn regarding model effectiveness against both researchers.

- J48: A decision tree classifier that builds a tree structure by splitting the dataset into subsets, creating an interpretable model.
- PART: A rule-based classifier that categorizes parts into decision lists, combining aspects of decision trees and rules.
- JRip: An implementation of the RIPPER algorithm, making it a rule-based classifier that creates optimized rule sets.
- RandomForest: An ensemble method that forms multiple decision trees to help improve predictive accuracy and to reduce overfitting
- Logistic Regression: A probabilistic classifier that creates models based on the logistic function
- LogitBoost: An algorithm that combines numerous weaker learning models, such as decision stumps, into a strong classifier.
- OneR: A rule-based classifier that takes one feature with the lowest error rate to make predictions.
- Naive Bayes: A probabilistic classifier that applies Bayes' theorem, which assumes feature independence.

E. Evaluation Metric

When testing each classifier, they were evaluated and ranked using their correctly classified instances (accuracy) as the main metric. These datasets and WEKA define accuracy as the number of correctly classified instances compared to the total number of instances it was run against, providing a direct measure of how each classifier performed. Since the objective of this experiment was to compare the overall ability of different algorithms on the two selected datasets, accuracy was chosen as a solid benchmark across each classifier and dataset.

IV. Experimental Setup

All testing and training were conducted by both researchers using the WEKA machine learning toolkit (version 3.8.6). As previously mentioned in the methodologies, both data sets were trained using the same 8 classifiers. A standard 10-fold cross-validation was applied to each classifier, allowing for a cohesive and equal assessment of the classifier's performance.

The first research workspace was equipped with an AMD Ryzen 9 5900x processor, 32 GB of DDR4 memory operating at 3200 MHz, an NVIDIA RTX 3080 GPU, and storage was a NVMe M.2 Solid State Drive. This build was used to ensure fast responses from WEKA. The second researcher's workstation was equipped with an Intel Core i9-11900K processor, 32 GB of DDR4 memory operating at 3600 MHz, an NVIDIA RTX 3070 Ti GPU, with storage being provided by a 1 TB NVMe M.2 solid state drive. WEKA primarily only utilizes CPU resources, but details of the hardware equipment for both researchers are provided to ensure transparency. To ensure full utilization of the aforementioned workstations, the researchers modified WEKA settings to increase RAM usage to prevent any form of bottlenecking, allowing the datasets to be run and tested without memory errors or long runtimes.

V. RESULTS

The ransomware and BODMAS malware datasets were used to train eight different algorithms to measure performance after applying top-rated features for more accurate detection of potential ransomware and malware. Each result has been rounded to the nearest tenth for easier comparison.

For the ransomware dataset, the two algorithms that stood out the most had the lowest accuracy in correctly classifying instances. Logistic scored 86.4% accuracy, while OneR achieved 89%. The remaining algorithms performed at a much higher level, with LogitBoost at 97.2%, NaiveBayes at 98.1%, PART at 98.6%, J48 at 98.8%, JRip at 99.5%, and RandomForest achieving the highest accuracy overall with 99.7%.

Algorithm	Accuracy(%)
RandomForest	99.7
JRip	99.5
J48	98.8
PART	98.6
NaiveBayes	98.1
LogitBoost	97.3
OneR	89.0
Logistic	86.4

Table 1.1: Ransomware Algorithm Accuracy

When training the algorithms on the BODMAS malware dataset, the overall scores and rankings shifted slightly. The lowest result came from NaiveBayes at 89.7%, followed by OneR at 91.3% and LogitBoost at 92.4%. The remaining algorithms performed at a similarly high level. JRip reached 98%, J48 slightly higher at 98.1%, and both RandomForest and Logistic tied with 98%. The highest overall result came from PART, which achieved 98.3%.

Algorithm	Accuracy(%)
PART	98.3
J48	98.1
JRip	98
RandomForest	98
Logistic	98
LogitBoost	92.4
OneR	91.3
NaiveBayes	89.7

Table 1.2: BODMAS Algorithm Accuracy

VI. DISCUSSION

This section interprets the results of our experiments, showing how different machine learning classifiers performed on the ransomware and BODMAS datasets and what these outcomes mean for lightweight malware detection. On the ransomware dataset, Random Forest stood out with the highest accuracy at 99.7%. JRip and J48 followed closely with 99.5% and 98.8%, and PART also stayed competitive at 98.6%. These small differences demonstrate that tree-based and rule-based approaches are very strong when working with static Portable Executable (PE) features. While Random Forest gave the best results overall, the fact that JRip and J48 came so close is important because not only did they achieve high accuracy, but they also provided interpretable results. Furthermore, they may have a lighter computational cost. These models produced outputs that are easier to interpret, which makes them practical in situations where understanding the reasoning behind detection matters.

The simpler models performed less consistently. Logistic Regression scored 98% accuracy on the BODMAS dataset but had a significant drop to 86.4% accuracy on the ransomware dataset. Naïve Bayes showed a similar performance issue, scoring 98.1% accuracy on BODMAS but only 89.7% on the ransomware dataset. While LogitBoost performed better than Logistic and OneR, its results were more inconsistent between the datasets. The accuracy dropped sharply from 97.3% on ransomware to 92.4% on BODMAS, which suggests that it may only excel when the dataset has patterns that can be easier for the model to understand. The dependence on dataset quality and composition limits its reliability and makes it less practical than the stronger classifiers. OneR was the weakest performer overall, with 89% on ransomware and 91.3% on BODMAS. These results highlight the limits of models that assume most features are independent of each other, when in reality, many file attributes are interconnected. In this case, the simpler models missed patterns that the tree-based and rule-based methods captured. OneR, while not practical for real-world detection, provides a useful baseline to show why more advanced classifiers are needed.

The datasets themselves also had an impact on the outcomes. The ransomware dataset created a clearer separation between malicious and benign samples, which allowed the stronger classifiers to show their advantage while exposing the weaknesses of Logistic Regression and Naïve Bayes. The BODMAS dataset was more challenging because it included a wider variety of malware families, but even with that variety, the top models still achieved close to 98% accuracy. This suggests that the stronger classifiers can handle a broad range of malware types without losing much effectiveness.

When comparing results across both datasets, the results showcase an important trade-off between accuracy, computational efficiency, and interpretability. Random Forest provides the highest accuracy but may require more computing resources, which may not be ideal for lightweight scanners. J48 and JRip performed nearly as well while offering easier explanations of their decisions, making them more practical in environments where human analysts need to validate or trust the outputs. Logistic Regression and Naïve Bayes, while generally less accurate, may still be useful in resource-constrained systems where speed or efficiency matters more than absolute accuracy.

While the results are strong, there are still important limitations. Our approach only used static PE file features, which means the models cannot detect malware that hides its behavior until execution or that changes after it runs. The datasets, although diverse, also do not represent every malware family or account for completely new, zero-day threats. These constraints mean that while the classifiers performed well in testing, their effectiveness in real-world situations may be lower. These points make clear that while the results are strong, there is still room for improvement in future work.

VII. CONCLUSION

The project presented a fast and practical approach to malware detection using machine learning frameworks. We were able to leverage static file features and use two unique but complementary datasets. Our experiments revealed that Random Forest and J48 presented impressive performance and achieved high detection accuracy while being efficient in their results and analysis. These results were able to establish and accomplish the initial goals of our objective for this study and provide support for the feasibility of machine learning based scanners as possible alternatives to the signature-based solutions if applied properly.

As technology improves, more research is done in this field of work. We propose that for future work, we could expand this method by using more dynamic features to improve the accuracy of the tool to detect more obscure types of malware. We also believe that as computing power issues become a thing of the past, with the large strides we are seeing in computational growth with the surge of artificial intelligence, we will be able to deploy even better and more robust models that can help keep our systems safe from the threat of malware.

REFERENCES

[1] A. Vehabovic *et al*., "Ransomware Detection Using Federated Learning with Imbalanced Datasets," *2023 IEEE 20th International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT (HONET)*, Boca Raton, FL, USA, 2023, pp. 255-260, doi: 10.1109/HONET59747.2023.10375023.

[2] Chen, Q., Islam, S.R., Haswell, H., Bridges, R.A. (2019). Automated Ransomware Behavior Analysis: Pattern Extraction and Early Detection. In: Liu, F., Xu, J., Xu, S., Yung, M. (eds) Science of Cyber Security. SciSec 2019. Lecture Notes in Computer Science(), vol 11933. Springer, Cham. https://doi.org/10.1007/978-3-030-34637-9_15

[3] J. Ispahany, M. R. Islam, M. Z. Islam and M. A. Khan, "Ransomware Detection Using Machine Learning: A Review, Research Limitations and Future Directions," in *IEEE Access*, vol. 12, pp. 68785-68813, 2024, doi: 10.1109/ACCESS.2024.3397921

[4] A. Mughaid, R. Ibrahim, M. AlJamal and I. Al-Aiash, "Detection of Trojan Horse in the Internet of Things: Comparative Evaluation of Machine Learning Approaches," *2024 International Conference on Multimedia Computing, Networking and Applications (MCNA)*, Valencia, Spain, 2024, pp. 35-41, doi: 10.1109/MCNA63144.2024.10703911.

[5] D. Schlienger, "AI is a Trojan Horse – A Rationale," *2024 IEEE International Symposium on Technology and Society (ISTAS)*, Puebla, Mexico, 2024, pp. 1-6, doi: 10.1109/ISTAS61960.2024.10732771.

[6] E. Barros, M. M. A. Rosa, R. Lopes, L. Antonietti, E. Costa and R. Soares, "Evaluating the Resilience of the Approximate Parallel Prefix Adder (AxPPA) Against Hardware Trojan Horse Injection," *2024 37th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI)*, Joao Pessoa, Brazil, 2024, pp. 1-5, doi: 10.1109/SBCCI62366.2024.10704005.

[7] Kukutla, Tejonath Reddy. (2023). A Comprehensive Guide to Computer Worms and Defense Strategies.

[8] de Carnavalet, X. D. C., & Mannan, M. (2019). Privacy and Security Risks of" Not-a-Virus" Bundled Adware: The Wajam Case. arXiv preprint arXiv:1905.05224.

[9] Shemakov, R. (2019). The Morris Worm: Cyber Security, Viral Contagions, and National Sovereignty. Swarthmore College. https://scholarship.tricolib.brynmawr.edu/handle/10066/22427