Comparative study of malware detection techniques for industrial control systems.

REID, D., HARRIS, I. and PETROVSKI, A.

2021

Comparative Study of Malware Detection Techniques for Industrial Control Systems

Deborah Reid
School of Computing
Robert Gordon University
Aberdeen, Scotland
d.a.reid@rgu.ac.uk

Ian Harris
School of Computing
Robert Gordon University
Aberdeen, Scotland
i.s.harris1@rgu.ac.uk

Andrei Petrovski
School of Computing
Robert Gordon University
Aberdeen, Scotland
a.petrovski@rgu.ac.uk
ORCID: 0000-0002-0987-2791

***Abstract*—Industrial Control Systems are essential to managing national critical infrastructure, yet the security of these systems historically relies on isolation. The adoption of modern software solutions, and the unique challenges presented by legacy systems, has made securing industrial networks increasingly difficult. With malware identified as the leading cause of cyber incident in industrial systems, this work presents a comparative study of existing malware detection techniques, to compare both accuracy and suitability for use in the defence of industrial systems.**

Keywords—industrial control systems, malware detection, machine learning, cybersecurity

I. Introduction

Industrial Control Systems (ICS) are responsible for controlling national critical infrastructure such as power stations, water treatment centres, and manufacturing plants. However, the security of these systems has long been based on isolation and proprietary software. With cyber-attacks becoming more sophisticated and the introduction of Enterprise solutions and the Internet of Things (IoT) in industrial networks, protecting these critical systems has become even more important.

There is a long history of cyber-attacks targeting industrial systems using malware, from the Trojan inserted into the Trans-Siberian pipeline by the CIA, to the recent attacks on the Ukrainian power grid in 2015 and 2016 [1]. Defending complex legacy systems from malware requires resilient detection solutions capable of protecting these critical systems without interference to their operation.

While there is an abundance of research related to malware detection, the discussion surrounding security and malware detection in industrial control systems is limited. A report by the US Army Research Laboratory [2] found that malware attacks were the most common cause of incidents in ICS systems, and this threat continues to rise. As ICS networks begin to adopt modern solutions, the importance of securing these systems also increases.

II. ICS Security Landscape

Traditionally deployed on isolated networks, ICS are known for using proprietary control protocols and specialised system components to ensure high availability and real-time response [3]. However, there is now a rise in ICS adopting IT solutions to replace and upgrade propriety components, introducing remote access capabilities and allowing ICS to communicate with the wider business network. While the transition away from isolated, propriety networks support new capabilities, it also increases the risk of introducing new vulnerabilities.

A. Industrial Internet of Things

The Industrial Internet of Things (IIoT) refers to the use of IoT devices in industrial sectors, bringing together traditional automated ICS systems and advanced Internet-based capabilities to create "cyber-physical" systems. With modern ICS solutions turning away from isolated networks in favour of more connected systems, IIoT provides a novel approach to implementing additional features and capabilities to ICS [4].

Despite the new technologies IIoT can introduce to ICS, the security and hardening of these connected devices may be unknown, with the potential to introduce new threat surfaces. Generally, IIoT devices are resource-constrained in terms of processing power, memory, and energy consumption. This means that much like traditional ICS components, there is little scope for additional security modules within IIoT devices [5].

B. ICS Security Threats

Threats to ICS can take numerous forms, ranging from structural threats, such as equipment failure, to accidental damage, including human error, to adversarial threats from individuals, groups, or nation-states [6]. The Kaspersky ICS CERT [7] report on the "Threat Landscape of Industrial Automation Systems" states that the main threat sources in the ICS environment as Internet, removeable media, and email, with the last two commonly used as carriers for malware.

Historically, the main cause of ICS cyber-incidents is malicious software that infects and damages hosts within a

system by deploying a malicious payload or overloading server central processing units (CPU) with can harm the performance of the system [2]. Detecting and defending against the malware threat is an important step to securing industrial systems, using security mechanisms to detect known and unknown malware variants, and protect against advanced malware techniques [8].

C. Malware in Industrial Systems

Historic cyber-attacks against ICS, such as Stuxnet in 2010 and Shamoon in 2012, have shown that despite air-gapped networks, the use of obscure protocols, and the deployment of specialised components, ICS are still vulnerable to cyber threats [2].

Modern malware threats can employ advanced techniques to hide their true intentions and avoid detection from anti-virus programs and intrusion detection systems. Commonly seen techniques such as polymorphism, the ability to appear in different forms, and obfuscation, the ability to hide true intentions behind a benign appearance, drive the development of more advanced detection techniques [19].

Research by Dragos Inc. [10] recently published details regarding ICS-specific ransomware, capable of targeting ICS operations and forcible stop processes. Their report on the EKANS and MEGACORTEX malware families identifies the first known ICS-specific ransomware variants. ICS-specific malware is currently limited, but these new creations show an evolution of tactics of threat actors turning towards targeting ICS environments.

III. Related Work

Detecting malware in inbound traffic and within systems can be made difficult with the increased use of advanced obfuscation techniques. Implementing a robust detection system can help protect a network from successful attack. This section discusses existing research in the field of malware detection and the different approaches to defending against this threat.

A. Static Analysis Techniques

Sufficient monitoring and logging within a computer network are the least technical methods of malware detection, yet it provides system engineers and administrators with an overview of the system and a valid way to detect changes or abnormal activity.

Commercial antivirus products typically employ signature-based malware detection, where signatures from malware code, for example a unique sequence of bytes, are used to determine if suspicious programs are malicious in nature by comparing the program to malware signatures. However, this method of detection cannot identify malware which can mutate during deployment, as each mutation generates a new signature [8].

An approach presented by [11] uses data-clustering based anomaly detection, designed for ICS using data from a range of sources such as industrial processes and network traffic. While this approach provides hardening to a system, the required storage space and performance drain could negatively affect ICS availability [12].

B. Behaviour-based Analysis Techniques

Behaviour-based malware detection looks to overcome the vulnerabilities of static analysis, which can be vulnerable to code obfuscation techniques [9]. Dynamic analysis of malware overcomes this issue by monitoring the behaviour of files during execution, usually in a virtual or sandbox environment.

One behaviour-based approach suggested by [13] employs virtual environments to monitor sample API calls, as certain malware families rely on libraries provided by the host operating system. The results of this research showed a 97% detection accuracy when using a Decision Tree for classification.

However, [9] suggests API calls are vulnerable to manipulation, resulting in misclassification. Their own solution focuses on using short snapshots of behavioural data from samples, suggesting a 'sliding-window' approach over full behavioural analysis, which they argue can be time consuming and runs the risk of malicious samples executing their payloads before detection.

C. Deep Learning Techniques

Deep Learning and unsupervised learning algorithms hold promise for automating malware detection in systems and developing generalisations for identifying benign and malicious files [14]. The process of Deep Learning involves training a network of artificial neurons to recognise complex patterns and classify new samples [15].

Research by Saxe and Berlin [14] and Dahl et al. [16] both present Deep Learning models for malware classification using neural networks. Saxe and Berlin achieved an accuracy of 95% with a false positive rate of 0.1% using a deep neural network with three layers, with a dataset of 431,926 benign and malicious samples.

In contrast, Dahl et al. achieved an accuracy of ~90% with a false positive rate of 0.83% and a false negative rate of 0.35%, using an ensemble of shallow, one-layer neural networks and a large dataset of 2.6 million files. Both approaches show that deep learning can be used effectively to provide accurate malware detection.

D. Malware Detection in ICS

As shown, research into malware detection and classification is widely conducted, with researchers continuing to search for more advanced and accurate techniques. However, there is little attention on this field within industrial systems and the use of modern technology to defend legacy systems.

The study conducted in this paper looks to provide a comparison of detection techniques with a focus on their application in an industrial environment, as the unique requirements of ICS demand a malware detection technique that works with the system, not against it.

IV. Comparative Study

The comparative study presented in this paper looks to compare the accuracy of three different malware detection approaches: signature-based, behaviour-based, and deep learning. The goal of the study is to identify the ability of each technique to identify known and unknown malicious samples.

A. Tools and Architecture

Development of a robust IDS tool requires extensive use of advanced topics such as machine learning and network traffic monitoring. In place of developing custom modules, that may be insecure or inaccurate, the decision was made to use the Python language to complete the implementation of the study experiments, which benefits from a large selection of libraries designed for these topics [17].

Due to limitations with physical hardware, virtualisation will be used to deploy virtual machines for the purposes of malware analysis. Use of virtual machines provides an additional layer of security while working with malicious files, keeping potentially damaging software in an isolated environment. Another benefit to using virtual machines is the ability to create new machines when required, without needing additional physical hardware for hosting.

As part of this project, a storage solution is required to store the project dataset, solution results, and provide a temporary storage for network traffic for the detection solution. SQLite was chosen as the storage solution, as it provides fast data access and a small code footprint. Additionally, SQLite databases are self-contained and can be embedded into applications, without the need for a standalone server to host the database.

B. Dataset

Research to identify existing malware research datasets found that few repositories of samples exist, with even fewer containing data related to industrial systems. Two research datasets containing malicious and benign samples were identified: EMBER (Elastic Malware Benchmark for Empowering Researchers) and SOREL-20M (Sophos-ReversingLabs – 20 Million) [18], both of which are publicly available for use in malware detection research.

Following evaluation of these two datasets, SOREL-20M was chosen to provide malicious samples for the project dataset. Since the dataset contains classified samples, features, and the raw binaries, there is scope for use of the samples for each detection technique in the comparative study.

1) Populating Study Dataset

The SOREL-20M dataset contains samples, features, and raw binaries for approximately 10 million malware samples and is available from an Amazon Web Services (AWS) S3 Bucket cloud storage.

The original dataset contains approximately 8 Terabytes of data, with the raw binaries contributing the most to the extremely large size. As the storage available for this work was limited, there was no possibility of storing the whole dataset. Therefore, 1000 samples were chosen at random from the SOREL-20M dataset using SQLite's built in *random()* SQL function to select samples from the dataset `meta.db` file, which contains the binary hash, class, and malware tags for every sample.

The malicious samples contained in the SOREL-20M dataset are all Portable Executable files. Portable Executable (PE) is the standard file format for executable files under the Windows operating system, with the name of the format a reference to the fact these files are not architecture specific.

As the SOREL-20M dataset does not contain the raw binaries for any benign samples, these files were sourced from a Windows 7 virtual machines, taking copies of benign Windows executables and other PE files, such as Dynamic-Link Libraries (DLL), screensavers, and system files.

2) Class Imbalance

The final project dataset following analysis and processing contains a total 1949 samples, with 1011 benign samples and 938 malicious samples. For this work, class distribution is slightly imbalanced, with Benign being the majority class. However, there is no extreme disproportion between classes which lowers the risk of the training model being overwhelmed and increases the chance of successfully detecting Malicious samples.

3) Dataset Similarity

As this project focuses on the protection of industrial and legacy systems, it is important to show the link between the dataset malware samples and live malware known to target industrial systems and networks.

Known industrial malware samples were download from `theZoo` [19], a malware binary collection hosted on GitHub, and analysed using Cuckoo Sandbox [20] to provide static and behavioural data. These live samples were then compared to the project dataset, to determine the level of similarity with the project malware samples, calculated using Cosine Similarity of each sample's DLL imports, supplied by the Cuckoo Sandbox analysis.

Cosine Similarity is used to determine the similarity between two documents or vectors, irrespective of size, having the advantage over Euclidean Distance, another common similarity metric which calculates similarity by counting common values in documents. The Cosine Similarity approach offers an alternative to as document size is not considered [21]. The equation below outlines the calculation for Cosine Similarity.

$$Similarity(a,b) = \frac{a \cdot b}{\|a\|\|b\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_1^2}\sqrt{\sum_1^n b_1^2}}$$

The similarity of the study dataset samples and the live malware samples was based on the DLL imports called by each sample. Two known ICS malware were used to perform the similarity calculations. Fig. 2 and Fig. 3 show the results of this experiment and display the percentage similarity for both malware samples against the malicious samples included in the project dataset.

Shamoon, the malware used in the 2012 Saudi Aramco attack [2], and Triton, a malware known to target industrial safety controllers [22], both share similarities with samples in the study dataset, with some samples showing 50% and above similarity. The rise in similar samples around 40-60% indicates shared DLL imports, common files used by both the malware and the samples. As the malicious nature and type of the dataset samples is unknown, the higher similarity suggests the sample shares the same malware family or behaviour as the industrial malware samples.

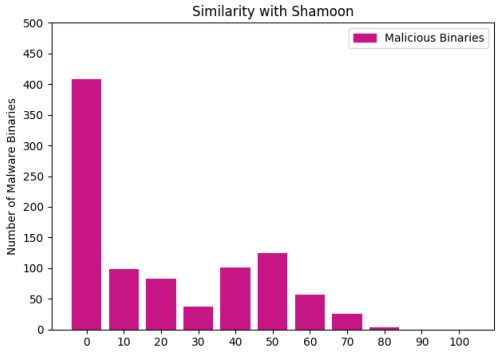

Fig. 3. Dataset Similarity with Shamoon

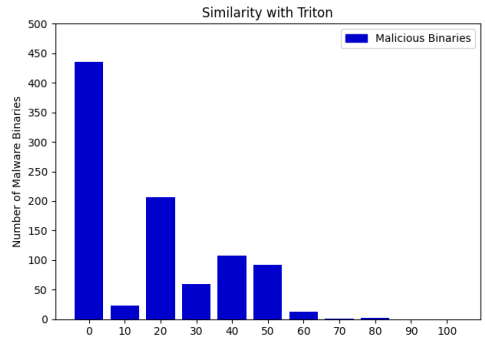

Fig. 4. Dataset Similarity with Triton

Although not exact due to the random nature of the project malware samples, this basic test of similarity suggests that samples in the dataset could affect industrial systems, lending support to the developed detection techniques and solution capabilities of defending legacy systems from malware.

C. *Dynamic Analysis*

To perform behaviour-based analysis for the comparative study, behavioural data was required. Although the SOREL-20M dataset contains the results of dynamic analysis and feature extraction for each sample, this information is stored in large files that cannot be accessed directly in the AWS S3 storage and are too large to download. Therefore, to provide the required behavioural information, dynamic analysis of samples was completed using Cuckoo Sandbox, which was deployed and configured for the project.

Due to a lack of additional physical machines, Cuckoo Sandbox was installed using virtual machines, hosted using VMware Workstation Pro 15. Cuckoo Sandbox consists of two main processes: the Cuckoo Host, which controls the analysis, configuration, and stores the results; and the Cuckoo Guests, which perform the analysis.

D. *Feature Extraction*

While the analysis reports from Cuckoo Sandbox provide extensive details about the execution of samples, additional information was required for the Deep Learning analysis. The `pefile` Python package, used in part by Cuckoo Sandbox, was used to perform feature extraction on the samples to provide static analysis for the Deep Learning detection, a technique performed by [14] in their similar malware detection research.

PE feature extraction was performed on each sample, with the information retrieved from the sample Image Optional Header recorded for use in the Deep Learning detection. This feature extraction returns useful information about the sample and its contents provide enough information for the Deep Learning technique to accurately distinguish malicious samples from benign.

V. Implementation of Techniques

To conduct a comparative study of the three chosen malware detection techniques and determine the most accurate, the study will record accuracy, false positive rate, and false negative rate for each technique.

Additionally, to provide standard sets of samples for both training and testing, a subset of the project dataset is chosen when running the comparative study. This subset initially contains 1500 samples chosen at random from the project database, containing both malicious and benign samples. Using the `train_test_split()` function available from the Scikit Learn library [23], the subset of samples is split into a training set and a testing set, with no overlap of samples.

The training set contains 70% of the original 1500 samples and will be used only for training the models for each detection technique. The test set contains the remaining 30% of the samples and will be used to test each technique. For a fair comparison of techniques, the same training and test sets was used for each technique.

A. *Signature-based Detection*

For the purposes of this project, a signature will be generated for each sample by taking the hash of the sample contents and comparing the hash to a repository of known malicious signatures. File hashes were chosen over byte strings from file contents due to the lack of information surrounding each malicious sample, as each malware has unique indicators of compromise and without knowing what malware family each sample belongs to, it is difficult to identify the signature byte strings that identify samples as malicious.

To demonstrate signature-based detection, SHA256 hashes were generated for the malicious and benign samples in the project dataset.

For this technique within the comparative study, the training set is used to populate the hash repository, with each malicious hash in the training set added to the repository. To test this technique, each hash in the test set is compared to the hash repository to identify if the sample is malicious. Predictions recorded by the method are then compared to the actual results, to provide an accuracy for the technique.

B. *Behaviour-based Detection*

Behaviour-based detection is conducted using the DLL import data for each sample, which was generated by the Cuckoo Sandbox analysis. The classification algorithm chosen for this technique is a Linear Support Vector Machine (SVM), a model used for binary classification problems, built using the Scikit Learn Python library.

As the DLL imports are stored as a string value in the project database, the values for each sample are first transforms into a JSON array using the built-in `json` Python standard package [17]. The class and DLL imports array for each sample in the training or test set is then passed to the Behaviour class, which performs normalisation and scaling on the data to format the DLL import array into integer vector features, as the Linear SVM model cannot perform classification on text.

The behaviour-based technique uses integer feature vectors to perform classification, the length of which equals the number of features set by the model, which for this project is set to 50, an arbitrary value that could be adjusted following future investigations into which parameter values provide the highest accuracy.

For each sample, an empty feature array is created with a length equalling the number of features set by the model. The normalisation of the DLL import data begins by looping through each DLL in the array and generating a hash value for each DLL, calculated using the Python built-in `hash()` function and returning the remainder when divided by the number of features. The returned value will be between 0 and the number of features. Using this value as an index for the sample feature vector, the value in that position is incremented by one. Upon completing the hash function for each DLL string, the feature vector containing integer values representing each DLL will be returned for classification.

C. Deep Learning-based Detection

The Deep Learning technique uses a neural network with three hidden layers to perform detection, created using the `MLPClassifier()` function from Scikit Learn [23]. This technique uses the Image Optional Header information extracted from each sample to perform the classification.

The MLPClassifer provides a multi-layer perceptron, a class of feedforward Artificial Neural Network (ANN). For this experiment, three hidden layers were used with 30, 15, and 10 neurons respectively, using the ReLU activation function and a stochastic gradient descent solver.

As the values extracted from the header of each sample are between very different value ranges, to perform accurate detection the values must be standardised using a scalar function. This will ensure all values are scaled to the same value range and is a common requirement for many machine learning algorithms. The chosen scalar for this technique is the `MinMaxScalar`, provided by the Scikit Learn library, which will transform each feature into a value between 0 and 1.

Much like the behaviour-based technique, the training set is used to train the Deep Learning model, with the training set used to fit the model scaling when used with the test set. Once training is complete the Deep Learning model is tested using the test set, with predicted classes recorded alongside the actual class to allow for accuracy calculations.

VI. Results

Table I shows the results of the comparative study, outlining the recorded accuracy, false positive rate, and false negative rate of each technique. The experiments to test each technique were completed ten times to produce an average result.

TABLE I. Results of Comparative Study

Technique	Accuracy, %	False Positive Rate	False Negative Rate
Signature-based	52.22	0.0 (known)	1.0 (unknown)
Behaviour-based	83.24	0.19	0.15
Deep Learning	**95.53**	0.07	0.015

Signature-based detection provided the highest efficiency for identifying known threats but proved weak when detecting unknown samples. This method of detection relies heavily on existing knowledge to build a repository of known threats, and with limited ability to detect unknown threats, this method is less suitable for critical industrial systems.

Behaviour-based detection provided high accuracy, but also recorded the highest false negative rate, making it less successful at detecting malicious samples. With high false negative rates, this technique would potentially allow malicious threats into a system.

Additionally, this technique may not detect malware samples designed to evade sandbox environments and would require more resources to conduct dynamic analysis, which may not be available within legacy systems. This technique would be best utilised as part of a wider security approach, working in tandem with other monitoring tools.

The Deep Learning approach achieved the highest accuracy, with an increase of 10% over the behaviour-based approach, with low false positive and false negative rates, confirming this technique can detect unknown malicious samples with a high degree of success. In comparison with the results achieved by [14] and [16], this approach achieved similar values for accuracy. Optimising the hyperparameters such as the batch size, number of epochs, and number of hidden layers for the neural network may produce higher accuracy results or lower the false positive and negative rates.

VII. Limitations

The most significant limitation of this work is the reduced size of the project dataset, as only a limited number of samples were used to train and test the techniques in the comparative study, and to test the detection solution. This was due to a lack of storage space on the available physical hardware used to conduct the project, as well as the time required to analyse a greater number of samples.

Furthermore, only Portable Executable files were used to provide samples for the project dataset, due to the limited availability of research-level malware datasets to source malicious samples. Therefore, the developed malware detection solution is only capable of analysing PE files due to their unique file structure.

Resources for the study were limited due to a lack of additional storage and physical hardware, resulting in the use of

virtual machines for both the Cuckoo Sandbox malware analysis and to simulate a network for the detection solution.

While this work focused only on the accuracy of each detection technique, in a real-world deployment, analysis time would be an important factor and critical for identifying threats quickly in an industrial or legacy network. Therefore, further research would be required to identify the time taken to analyse samples for each technique.

VIII. Future Work

There is a large scope for future work expanding on the foundation presented by the results of the conducted comparative study.

With the availability of increased storage and processing power, a larger dataset could be used to further test the techniques chosen for the comparative study. Additionally, the possibility of future research using a greater variety of files and samples would provide a more accurate comparison of detection technique in different environments.

As the project focused on accuracy statistics to provide comparisons and results, future research could investigate other statistics such as resource use, processing power, and analysis time. Additional work using the compared detection techniques on physical hardware or simulated industrial systems would also provide a greater insight into the ability of malware detection techniques to detect threats in a live system.

IX. Conclusion

The results of the comparative study show that each detection technique can provide accurate results, dependent on the scenario, similar to those presented in [23, 24]. While the study proved the Deep Learning technique can achieve the highest accuracy for the chosen dataset, this technique may not be the best fit for another dataset or file type. Future work exploring the use of Deep Learning in malware detection could provide further insight into the application of this technique in the cyber security domain.

Despite limitations, the work itself contributes to existing research in the field of malware detection, providing further insight into the use of machine learning in network security and for the protection of legacy systems.

References

[1] S. McLaughlin et al., "The cybersecurity landscape in industrial control systems", in *Proceedings of the IEEE*, vol. 105, issue 5, pp. 1039-1057, 2016.

[2] D. T. Sullivan. (2015). Survey of malware threats and recommendations to improve cybersecurity for industrial control systems version 1.0 (Online). Available: https://apps.dtic.mil/sti/pdfs/ADA617910.pdf

[3] D. Timpson and E. Moradian, "A methodology to enhance industrial control system security", in *Procedia Computer Science*, vol. 126, issue 2018, pp. 2117-2126, 2018.

[4] P. McLaughlin and R. McAdam. (2016). *The undiscovered country: the future of industrial automation* (Online). Available: https://www.honeywellprocess.com/en-US/online_campaigns/IIOT/Documents/The-Undiscovered-Country.pdf

[5] E. Sisinni et al., "Industrial internet of things: challenges, opportunities, and directions", in *IEEE Transactions on Industrial Informatics*, vol. 14, issue 11, pp. 4724-4734, 2018.

[6] K. Stouffer et al. (2014). *Guide to industrial control systems (ICS) security* (Online). Available: http://www.gocs.com.de/pages/fachberichte/archiv/164-sp800_82_r2_draft.pdf

[7] Kaspersky ICS CERT. (2020). *Threat landscape for industrial automation systems: H1 2020* (Online). Available: https://ics-cert.kaspersky.com/media/KASPERSKY_H1_2020_ICS_REPORT_EN.pdf

[8] P. Vinod, V. Laxmi and M. Gaur, "Survey on malware detection methods", in *Hack.in 2019: 3rd Hackers' Workshop on Computer Science and Internet Security*, Kanpur, India, 2009, pp.74-79.

[9] M. Rhode, P. Burnap and K. Jones, "Early-stage malware prediction using recurrent neural networks", in *Computers & Security*, vol. 7, pp. 578-594, 2018.

[10] Dragos, Inc. (2020). *EKANS ransomware and ICS operations* [Online]. Available: https://www.dragos.com/blog/industry-news/ekans-ransomware-and-ics-operations/

[11] I. Kiss et al., "Data clustering-based anomaly detection in industrial control systems", in *IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2014, pp. 275-281.

[12] E. Kabanga and C. H. Kim, "Malware images classification using convolutional neural network", in *Journal of Computer and Communications*, vol. 6(1), pp. 153-158, 2018.

[13] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behaviour-based features model for malware detection", in *Journal of Computer Virology and Hacking Techniques*, vol. 12(2016), pp. 59-67, 2016.

[14] J. Saxe and K. Berlin. (2015). *Deep neural network based malware detection using two-dimensional binary program features* (Online). Available: https://arxiv.org/pdf/1508.03096v2.pdf

[15] W. Knight. (2015). *Antivirus that mimics the brain could catch more malware* (Online). Available: https://www.technologyreview.com/2015/10/29/165374/antivirus-that-mimics-the-brain-could-catch-more-malware/

[16] G. Dahl et al., "Large-scale malware classification using random projections and neural networks", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3422-3426, 2013.

[17] G. Van Rossum, *The python library reference, release 3.9.4* [Online]. Available: https://www.python.org/downloads/release/python-394/

[18] R. Harang and E. M. Rudd, "SOREL-20M: A large scale benchmark dataset for malicious PE detection", Sophos, [Online], 2020. Available: https://arxiv.org/abs/2012.07634

[19] Y. Nativ, L. Ludar and 5fingers. (2015). *theZoo – a live malware repository* (Online). Available: https://github.com/ytisf/theZoo

[20] Cuckoo Sandbox. (2019). *Cuckoo Sandbox – automated malware analysis* (Online). Available: https://cuckoosandbox.org/

[21] S. Prabhakaran. (2020). *Cosine similarity – understanding the math and how it works* (Online). Available: https://www.machinelearningplus.com/nlp/cosine-similarity/

[22] National Cyber Security Centre. (2018). *TRITON malware targeting safety controllers* (Online). Available: https://www.ncsc.gov.uk/information/triton-malware-targeting-safety-controllers

[23] F. Pedregosa, "Scikit-learn: Machine learning in python", *Journal of Machine Learning*, vol 12, pp. 2825-2830, 2011.

[24] U. Otokwala, A., Petrovski., H. Kalutarage,."Effective Detection of Cyber-Attack in a Cyber-Physical Power Grid System*"*, in *Proceedings of the Future of Information and Communications conference*, 2021, vol. **1**, pp. 812-829.

[25] F. Majdani, et. Al. "Detecting Malicious Signal Manipulation in Smart Grids Using Intelligent Analysis of Contextual Data*"*, in *Proceedings of the 13th International Conference on Security of Information and Networks*, 2020, article No: **4**, pp. 1-8.