A COMPREHENSIVE FRAMEWORK FOR REAL-TIME MALWARE DETECTION AND MONITORING IN PRODUCTION

Elshan Baghirov *

Institute of Information Technology of The Ministry of Science and Education of the Azerbaijan Republic, Baku;
Azerbaijan

* Corresponding Author, e-mail: elsenbagirov1995@gmail.com

Abstract: This paper proposes a comprehensive framework for real-time malware detection and monitoring tailored to operational systems. Leveraging advanced machine learning algorithms, our framework integrates continuous monitoring mechanisms to ensure timely detection and response to emerging threats. The framework emphasizes regular assessment of model performance using metrics such as the Population Stability Index (PSI), ensuring models remain effective and adaptive to evolving malware patterns. By deploying models within the production environment, the framework enables regular evaluation and adaptation, enhancing the robustness and reliability of the detection system. Our results demonstrate the framework's efficacy in providing a scalable and efficient solution for real-time malware detection and monitoring, contributing to improved cybersecurity posture in dynamic and high-risk environments.

Key words: malware detection, malware analysis, machine learning, monitoring, real-time detection

1. INTRODUCTION

With the continuous growth of technology, contemporary attacks increasingly aim to impact availability, integrity, and confidentiality [1]. Malware poses a significant threat to the security and integrity of information systems, especially in production environments where downtime and breaches can have severe consequences.These applications can harm users by leaking sensitive data such as bank account details, blocking access to information and demanding monetary compensation (ransomware), or leveraging social engineering scams. The increasing sophistication of malware necessitates the development of advanced detection frameworks [2]. Traditional malware detection methods, including signature-based and heuristic approaches, are increasingly inadequate in keeping pace with the ever-evolving landscape of malware. These conventional techniques often fail to detect new, unknown malware variants, particularly zero-day attacks, which exploit vulnerabilities that have not yet been discovered or patched. The limitations of static and dynamic detection methods underscore the necessity for advanced, real-time detection frameworks capable of swiftly identifying and mitigating threats.

Recent years have witnessed the application of machine learning [2-5] and deep learning [6-13] techniques to enhance malware detection capabilities. Although numerous studies have advocated for the use of ML and DL in malware detection, there is a conspicuous lack of comprehensive monitoring processes. Continuous monitoring is vital to ensure the real-time adaptability and effectiveness of these detection models in production environments. Moreover, many researchers rely on commonly balanced datasets for training, often achieving nearly 99% accuracy. However, this does not reflect real-world scenarios where analyzed files within organizations are not evenly distributed between malware and benign files. This imbalance significantly challenges the models' performance and reliability in actual production settings. Upon analyzing existing works, it becomes evident that these studies typically encompass dataset formalization, preprocessing, training, testing, and model evaluation. However, they often neglect subsequent steps required for production deployment. Critical considerations such as overfitting, underfitting, and the practical performance of models in production environments are frequently overlooked. Given that these models are intended for operational use, it is imperative to address these aspects to ensure their robustness and adaptability in real-world applications. Concept drift, if unaddressed, can significantly degrade the performance of model, rendering it practically unusable. To tackle this, our framework employs continuous monitoring and adjustment mechanisms, ensuring the robustness and adaptability of models over time.

This work presents a comprehensive framework for real-time malware detection and monitoring in production environments, addressing several critical gaps identified in existing research. The key contributions are as follows:

- **Detailed Framework**: This work proposes a comprehensive framework for real-time malware detection and monitoring, tailored for production environments to ensure robustness and practical applicability.
- **Real-Time Monitoring, PSI Assessment, and Feature Drift Analysis:** The framework integrates real-time monitoring capabilities using the PSI to detect model drift, ensures model stability and adaptability over time, and monitors feature importance drift to maintain model accuracy and reliability in detecting new and emerging malware threats.

The rest of this article is organized as follows: Section II reviews previous studies on malware detection. Section III outlines the proposed real-time malware detection framework, detailing dataset, training, PSI and feature importance drift. The Section IV describes the experimental setup, datasets, evaluation metrics, and performance outcomes. Finally, Section V summarizes the findings, discusses the implications, and suggests future research directions.

2. LITERATURE REVIEW

The HEAVEN approach, presented in [13], combines hardware and software components for real-time, signature-based malware detection. It leverages Intel's x86/x86-64 branch predictors for rapid hardware-assisted signature matching, triggering software-based AV for in-depth analysis if suspicious patterns are found. This method reduces system load by avoiding constant monitoring. However, it relies heavily on specific hardware capabilities, limiting its applicability, and its signature-based nature may not be effective against unknown malware variants. Additionally, integrating these components adds complexity to deployment, maintenance, and updates.

The Defendroid approach enhances Android app security by detecting vulnerabilities during the source code writing process, as presented in [14]. It employs a blockchain-based federated neural network with Explainable AI (XAI), achieving high accuracy in identifying vulnerabilities. Defendroid facilitates real-time vulnerability detection in Android Studio but poses challenges in deployment and maintenance due to its complex integration and reliance on active participation for continuous updates. Scalability to handle large client numbers and data volumes is also a potential limitation.

The MeMalDet technique, a memory analysis-based malware detection method utilizing deep autoencoders and stacked ensemble learning, is presented in [15]. This approach addresses the evolving nature of malware attacks by introducing an improved dataset with temporal attributes for realistic evaluations. It extracts optimal features from memory dumps using deep autoencoders and employs a stacked ensemble of classifiers for accurate detection. Experiments demonstrate MeMalDet's high performance under temporal splits, achieving up to 98.82% accuracy and a 98.72% F1-score. Despite these advancements, challenges include implementation complexity, dependency on dataset quality, scalability issues, and computational resource requirements.

An adaptive real-time malware detection approach using API call sequences is presented in [16]. The method transforms API sequences into graphs and uses a Maximum Entropy Model for weighing pairs as malicious or benign. An optimized cluster algorithm and improved LSTM structure enhance detection accuracy, while results show high accuracy and robustness against adversarial attacks. However, real-time capability is only simulated, and balanced input data is required. Future work will focus on practical implementation and defense enhancements.

The effectiveness of Quantum Neural Networks (QNNs) for malware detection is presented in [17]. The proposed QNN framework, which integrates quantum feature extraction, classification, and real-time analysis, demonstrates a high accuracy of 0.95, outperforming both traditional machine learning models and other quantum models. While QNNs show promising results in malware detection, limitations include the need to enhance quantum-safe cryptography for securing quantum-enhanced systems. Integrating quantum technologies with classical methods poses challenges, but this convergence could lead to more resilient cybersecurity measures. Further exploration is essential to fully exploit QNNs' potential, laying the groundwork for a quantum-secure future in cybersecurity.

3. METHODS AND METHODOLOGY

3.1. Suggested framework

In the Figure 1, the flowchart depicts a typical workflow for malware detection using machine learning (ML) and deep learning (DL) methods, specifically focusing on Portable Executable (PE) files. The generalization presented in Figure 1 is prepared based on a synthesis of existing works. This workflow includes several key steps. The initial step involves gathering a comprehensive set of PE files, including both benign and malicious samples. These files are sourced from various datasets to ensure diversity and robustness. Common datasets used in this step include Drebin, CICMalDroid, Microsoft Malware Challenge Dataset, and VirusShare. Once the PE files are collected, they undergo disassembling to convert the executable binaries into human-readable code. This process helps in analyzing the underlying behavior and structure of the files. Popular tools for disassembling include PE-view, IDA-Pro, and Cuckoo Sandbox. In the feature extracting step, relevant features are extracted from the disassembled code. These features are critical for training the machine learning models, as they represent the characteristics that distinguish benign files from malicious ones. Examples of features include API calls, control flow graphs, opcode sequences, and other behavioral indicators. After feature extraction, the data is organized into a structured dataset. This dataset is divided into training and testing subsets to facilitate the training and evaluation of the models. A portion of the dataset is used to train the machine learning models, while another portion is set aside to evaluate the performance of the trained models. Using the training dataset, machine learning and deep learning models are trained. These models learn to identify patterns and characteristics associated with malware. Various algorithms can be used, including supervised learning, unsupervised learning, and ensemble methods. The trained models are then tested using the test dataset. This step evaluates how well the models can generalize to new, unseen data. Performance metrics such as accuracy, precision, recall, and F1-score are used to assess the models.

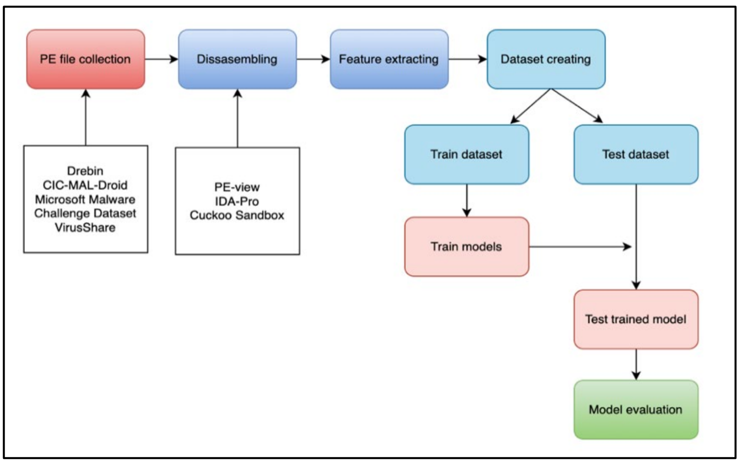

Figure 1. General workflow of current ML and DL malware detection methods

While the workflow presented in Figure 1 is a standardized approach for malware detection using machine learning and deep learning methods, it has several limitations when applied to real-world scenarios. The workflow does not account for critical factors such as PSI, and continuous monitoring and adaptation. Additional steps and considerations are necessary to ensure its effectiveness and robustness in real-world production environments.

Figure 2 illustrates the detailed workflow of our methodology. This flowchart demonstrates the steps from data collection and preprocessing to model deployment and continuous monitoring. Unlike typical workflows, which often end after model evaluation, our approach includes mechanisms for real-time monitoring and adjustments, making it suitable for practical applications in production environments.

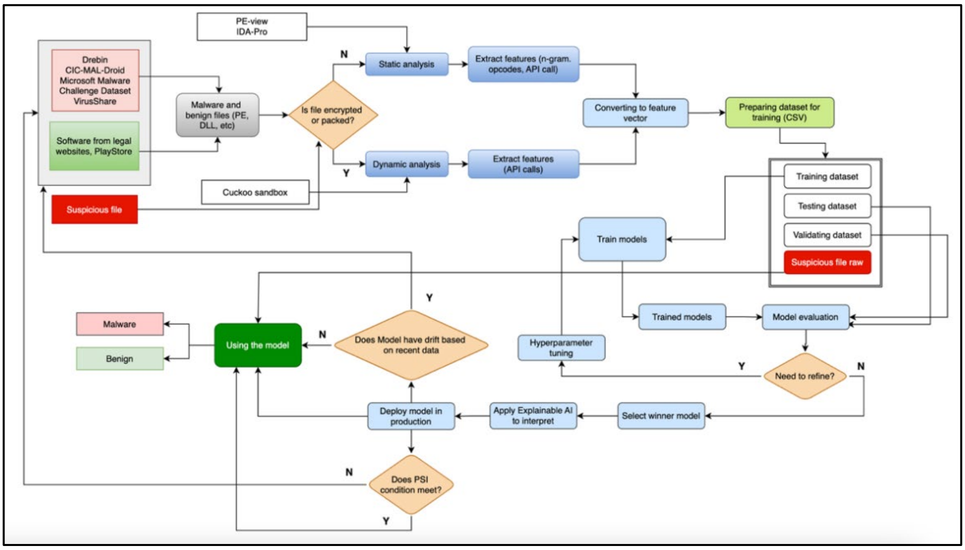

Figure. 2. Suggested real-time detection workflow for malware detection

This comprehensive methodology ensures that our models are not only accurate but also robust and adaptive to evolving threats, addressing common issues such as overfitting and feature importance changes.

The preprocessed dataset is split into training, testing, validation, and recent data subsets. The training and testing dataset is used to develop and evaluate the models. The validation dataset helps tune model parameters and prevent overfitting, ensuring the models generalize well to unseen data. Each model's performance is evaluated using the testing dataset, employing metrics such as training time, accuracy, precision, recall, F1-score, and ROC-AUC. To further ensure robustness, recent dataset is used to test the models, providing insight into how well they might perform on completely new data.

Additionally, the recent dataset plays a crucial role in monitoring model performance over time. This subset is used to check the PSI, which measures whether the distribution of current data significantly differs from the data used to develop the model. This step is essential for detecting model drift, ensuring that the model remains accurate and reliable as the underlying data changes over time. Feature importance drift is also assessed, evaluating changes in the significance of features used by the models, which could indicate shifts in the underlying data patterns.

3.2. The Population Stability Index

PSI is a statistical measure used to evaluate the stability of a predictive model's performance over time. It is particularly useful in monitoring the performance of models in real-world production environments, where changes in the population's characteristics can affect the model's accuracy and reliability. PSI helps to detect shifts in the distribution of features or the target variable, indicating whether the model needs recalibration or retraining [18]. The PSI is calculated by comparing the distribution of a feature or score in the current dataset (D_c) to a reference dataset (D_r). The PSI calculation process is as follows:

The data is divided into a series of bins. The same binning strategy should be applied to both the reference and current datasets. For each bin, the proportions of observations are calculated. Then, the PSI for each bin is computed using the formula given in Equation 1 [18]:

$$PSI = \sum \left((D_c - D_r) * \ln \frac{D_c}{D_r} \right) \tag{1}$$

The calculated PSI values can be interpreted as follows: a PSI less than 0.1 indicates a minimal shift in the distribution, suggesting model stability. A PSI between 0.1 and 0.25 indicates a moderate shift, suggesting the model may require attention. A PSI of 0.25 or greater indicates a significant shift, suggesting the model likely needs to be recalibrated or retrained.

PSI is essential as it quantitatively measures model performance stability over time, aids in the early detection of model drift by identifying shifts in data distribution, and informs decisions on whether model recalibration, retraining, or feature engineering adjustments are necessary.

3.3. Feature importance drift

Feature importance drift refers to the changes in the importance or influence of features used by a machine learning model over time. This drift can occur due to various reasons such as changes in the underlying data distribution, evolving patterns in the data, or shifts in user behavior. Monitoring feature importance drift is crucial for maintaining the accuracy and reliability of the model. To quantitatively measure feature importance drift, the Kolmogorov-Smirnov (KS) test can be used. The KS test is a non-parametric test that compares the distributions of a feature's importance scores between two different time periods [19]. This helps in identifying whether there is a significant change in the feature importance distribution, indicating potential drift. The formula for the KS statistic is given by Equation 2 [19].

$$D = max|F_1(x) - F_2(x)| \tag{2}$$

Here, $F_1(x)$ is the empirical distribution function of the feature importance scores in the reference period, $F_2(x)$ is the empirical distribution function of the feature importance scores in the current period.

The KS statistic D measures the maximum difference between the two empirical distribution functions. A higher D value indicates a more significant drift in feature importance.

4. ANALYSIS AND RESULTS

4.1. Dataset description

In this work, the Drebin dataset [20] was utilized to demonstrate the entire process. Drebin dataset contains 5,560 Android malware applications from 179 different families, collected between August 2010 and October 2012. Each application in the dataset is described by 182 features, encompassing a wide range of permissions, API calls, and hardware interactions. This dataset provides a comprehensive basis for evaluating malware detection techniques. The dataset was divided into several subsets: training, testing, validation, and recent data. The training dataset was used to train multiple machine learning models, including Random Forest, Gradient Boosting Tree, Logistic Regression, LightGBM, XGBoost, Decision Tree, and Support Vector Machine (SVM).

4.2. Results of trained models

The evaluation of multiple machine learning models aimed to determine their efficacy in detecting malware. Models were assessed based on several performance metrics, including training time, accuracy, precision, recall, F1-score, and ROC-AUC. To prevent overfitting, k-fold cross-validation was applied, and hyperparameter tuning was conducted using GridSearch. Table 1 presents the results of these evaluations, showcasing the performance of each model. As shown in Table 1, the LightGBM model outperformed other models with an accuracy of 0.98, precision of 0.99, recall of 0.97, and an F1-score of 0.98. The ROC AUC score for LightGBM was 1.00, indicating excellent performance in distinguishing between malware and benign files. Other models such as XGBoost, Random Forest, and SVM also demonstrated strong performance, with accuracy and precision values close to those of LightGBM.

4.3. Result of PSI

PSI for the top 20 variables within the dataset was assessed. The PSI for all selected variables was found to be zero. This result suggests that there has been no significant shift in the distribution of these variables, indicating that the model remains stable and reliable over time without any substantial changes in data distribution.

Table 1. Performance results of trained models.

Name	***Train Time***	***Accuracy***	***Precision***	***Recall***	***F1-score***	***ROC AUC***
XGBoost	*12s*	*0.96*	*0.96*	*0.93*	*0.95*	*0.99*
LightGBM	***11s***	***0.98***	***0.99***	***0.97***	***0.98***	***1.00***
Decision Tree	*7s*	*0.93*	*0.88*	*0.93*	*0.90*	*0.95*
Gradient Boosted Trees	*11s*	*0.96*	*0.97*	*0.93*	*0.95*	*1.00*
SVM	*1m 25s*	*0.98*	*0.98*	*0.96*	*0.97*	*1.00*
Logistic Regression	*11s*	*0.97*	*0.95*	*0.98*	*0.96*	*0.99*
Random Forest	*19s*	*0.98*	*0.98*	*0.97*	*0.98*	*1.00*

4.4. Result of feature importance drift

To further understand the stability of the model's variable importance, the importance scores of the top 10 variables were compared between the original and recent datasets. The mean absolute difference was calculated to quantify the changes in feature importance over time. The Table 2 presents the results. These results demonstrate that the model's feature importances remained largely stable, with minimal variation between the original and recent data. This stability indicates that the model has maintained its predictive power over time, with no significant feature drift observed.

Table 2. Model feature importance drift

#	Feature	Drift
1.	Landroid.content.Context.unregisterReceiver	0.02
2.	Ljava.lang.Class.getResource	0.01
3.	Ljava.net.URLDecoder	0.01
4.	WRITE_HISTORY_BOOKMARKS	0.01
5.	transact	0.01
6.	TelephonyManager.getLine1Number	0.01
7.	SEND_SMS	0.01
8.	onServiceConnected	0.01
9.	Ljava.lang.Class.getMethods	0.01
10.	RECEIVE_SMS	0.01

5. CONCLUSIONS AND FUTURE WORKS

In this study, we proposed a comprehensive framework for real-time malware detection and monitoring tailored to operational systems. Our framework integrates continuous monitoring mechanisms, employing advanced machine learning algorithms to ensure timely detection and response to emerging threats. The evaluation demonstrated the efficacy of the framework, with models like LightGBM achieving high accuracy, precision, recall, and F1-scores, indicating robust performance in real-world production environments. PSI and feature importance drift analysis were crucial in ensuring the stability and adaptability of the models over time. The PSI values for the top 20 variables remained zero, indicating no significant shift in data distribution. Similarly, the feature importance drift analysis showed minimal variation, confirming the models' predictive power remained stable.

This research contributes significantly to the field of cybersecurity, providing a scalable and efficient solution for real-time malware detection and monitoring, essential for maintaining robust defenses in dynamic and high-risk environments.

Future research will focus on implementing more sophisticated real-time adaptation mechanisms to further improve the models' responsiveness to evolving threats. Additionally, a deep analysis of the concept of drift will be conducted, splitting it into several categories such as dataset drift and model drift, to better understand and mitigate their impacts on model performance. Future work will also prioritize using more recent datasets to address contemporary malware threats more effectively, selecting data from diverse years to observe concept drift on a year-by-year basis.

REFERENCES

[1] Shahapurkar, A., Rodd, S. F. Efficient feature aware machine learning model for detecting fraudulent transaction in streaming environment. *International Journal on Information Technologies and Security*, vol.14, no.3, 2022, pp. 3-14.

[2] Costa, L. d., & Moia, V. A Lightweight and Multi-Stage Approach for Android Malware Detection Using Non-Invasive Machine Learning Techniques. *IEEE Access*, vol.11, 2023, pp. 73127-73144. DOI: 10.1109/ACCESS.2023.3296606

[3] Odat, E., & Yaseen, Q. M. A Novel Machine Learning Approach for Android Malware Detection Based on the Co-Existence of Features. *IEEE Access*, vol.11, 2023, pp. 15471-15484. DOI: 10.1109/ACCESS.2023.3244656

[4] Li, C., Chen, X., Wang, D., Wen, S., Ahmed, M. E., Camtepe, S., & Xiang, Y. Backdoor Attack on Machine Learning Based Android Malware Detectors. *IEEE Transactions on Dependable and Secure Computing*, vol.19, no.5, 2023, pp. 3357-3370.

[5] Almarshad, F. A., Zakariah, M., Gashgari, G. A., Aldakheel, E. A., & Alzahrani, A. I. A. Detection of Android Malware Using Machine Learning and Siamese Shot Learning Technique for Security. *IEEE Access*, vol.11, 2023, pp. 127697-127714. DOI: 10.1109/ACCESS.2023.3331739

[6] Al-Fawa'reh, M., Abu-Khalaf, J., Szewczyk, P., & Kang, J. J. MalBoT-DRL: Malware Botnet Detection Using Deep Reinforcement Learning in IoT Networks. *IEEE Internet of Things Journal*, vol.11, no.6, 2024, pp. 9610-9629. DOI: 10.1109/JIOT.2023.3324053

[7] Dong, S., Shu, L., & Nie, S. Android Malware Detection Method Based on CNN and DNN Hybrid Mechanism. *IEEE Transactions on Industrial Informatics*, vol.20, no.5, 2024, pp. 7744-7753. DOI: 10.1109/TII.2024.3363016

[8] Gibert, D., Zizzo, G., Le, Q., & Planes, J. Adversarial Robustness of Deep Learning-Based Malware Detectors via (De)Randomized Smoothing. *IEEE Access*, vol.12, 2024, pp. 61152-61162. DOI: 10.1109/ACCESS.2024.3392391

[9] Puneeth, S., Lal, S., Pratap Singh, M., & Raghavendra, B. S. RMDNet-Deep Learning Paradigms for Effective Malware Detection and Classification. *IEEE Access*, vol.12, 2024, pp. 82622-82635. DOI: 10.1109/ACCESS.2024.3403458

[10] Lu, T., & Wang, J. DOMR: Toward Deep Open-World Malware Recognition. *IEEE Transactions on Information Forensics and Sec*urity, vol.19, 2024, pp. 1455-1468. DOI: 10.1109/TIFS.2023.3338469

[11] Maray, M., Maashi, M., Alshahrani, H. M., Aljameel, S. S., Abdelbagi, S., & Salama, A. S. Intelligent Pattern Recognition Using Equilibrium Optimizer with Deep Learning Model for Android Malware Detection. *IEEE Access*, vol.12, 2024, pp. 24516-24524. DOI: 10.1109/ACCESS.2024.3357944

[12] Habib, F., Shirazi, S. H., Aurangzeb, K., Khan, A., Bhushan, B., & Alhussein, M. Deep Neural Networks for Enhanced Security: Detecting Metamorphic Malware in IoT Devices. *IEEE Access*, vol.12, 2024, pp. 48570-48582. DOI: 10.1109/ACCESS.2024.3383831

[13] Botacin, M., Alves, M. Z., Oliveira, D., & Grégio, A. HEAVEN: A Hardware-Enhanced AntiVirus ENgine to accelerate real-time, signature-based malware detection. *Expert Systems with Applications*, 2022, 117083. DOI: 10.1016/j.eswa.2022.117083

[14]Senanayake, J., Kalutarage, H., Petrovski, A., Piras, L., & Al-Kadri, M. O. Defendroid: Real-time Android code vulnerability detection via blockchain federated neural network with XAI. *Journal of Information Security and Applications*, vol.82, 2024, 103741. ISSN 2214-2126. DOI: 10.1016/j.jisa.2024.103741

[15]Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. MeMalDet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations. *Computers & Security*, vol.142, 2024, 103864. ISSN 0167-4048. DOI: 10.1016/j.cose.2024.103864

[16]Yang, S., Li, S., Chen, W., & Liu, Y. A Real-Time and Adaptive-Learning Malware Detection Method Based on API-Pair Graph. *IEEE Access*, vol. 8, 2020, pp. 208120-208135. DOI: 10.1109/ACCESS.2020.3038453

[17]Bikku, T., Chandolu, S., Praveen, S., Tirumalasetti, N., Swathi, K., & Sirisha, U. Enhancing Real-Time Malware Analysis with Quantum Neural Networks. *Journal of Intelligent Systems and Internet of Things*, vol. 12, 2024, pp. 57-69. DOI: 10.54216/JISIoT.120105

[18]Yurdakul, B. Statistical Properties of Population Stability Index. *Dissertations*. 3208. https://scholarworks.wmich.edu/dissertations/3208, 2018.

[19]Lopes, R.H.C. Kolmogorov-Smirnov Test. In: Lovric, M. (eds) International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg, 2011. https://doi.org/10.1007/978-3-642-04898-2_326

[20]Daniel, A., Michael, S., Malt, H. et al. Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket, 21st Annual Network and Distributed System Security Symposium (NDSS), February, San Diego, California, 2014.

Information about the authors:

Elshan Baghirov – PhD student at the Institute of Information Technology of The Ministry of Science and Education of the Azerbaijan Republic, simultaneously working as a senior data scientist at Kapital Bank. Research interests are malware analysis, data science, and information security incident management. The number of publications — 20.

Manuscript received on 30 Jun 2024