A Deep Learning Approach for Malware and Software Piracy Threat Detection

Khalid Aldriwish
Department of Computer Science
College of Science and Humanities
Majmaah University
Majmaah, Saudi Arabia
k.aldriwish@mu.edu.sa

Abstract-Internet of Things (IoT) -based systems need to be up to date on cybersecurity threats. The security of IoT networks is challenged by software piracy and malware attacks, and much important information can be stolen and used for cybercrimes. This paper attempts to improve IoT cybersecurity by proposing a combined model based on deep learning to detect malware and software piracy across the IoT network. The malware's model is based on Deep Convolutional Neural Networks (DCNNs). Apart from this, TensorFlow Deep Neural Networks (TFDNNs) are introduced to detect software piracy threats according to source code plagiarism. The investigation is conducted on the Google Code Jam (GCJ) dataset. The conducted experiments prove that the classification performance achieves high accuracy of about 98%.

Keywords-cybersecurity; malware; software piracy; deep learning; Internet of Things

I. INTRODUCTION

Artificial intelligence (AI) approaches are overgrowing through machine learning and deep learning technologies. Using AI in applications improves accuracy and efficiency. The AI approach supports innovations in various fields [1-10]. The Internet of Things (IoT), as an interconnection of device-based sensors through the Internet, requests a safety mechanism based on AI methods to prevent attacks and intrusion. IoT devices are defined by unique Radio Frequency Identifier (RFID) tags and are connected via nodes. The IoT interconnection mechanism ensures distant monitoring and controlling [9]. The IoT connectivity is a universal mechanism to support cloud computing, service industries, and innovative applications. With the number of connected devices via the Internet exceeding 50 billion by 2021 [10], data security becomes a significant challenge. The IoT technology faces a massive amount of data due to the growth of communication networks. Attackers benefit from the IoT architecture to handle attacks through IoT devices. Pirated software and malware infection have been used to affect the security of the industrial IoT cloud [11]. These methods attempt to reuse source code illegally and to use the system as a regular user. The attacker writes a malware code based on reverse engineering through the logic of the original code [12]. This kind of attack is a severe threat because it allows unlimited downloads of pirated software. This issue is solved by using an intelligent software plagiarism technique that finds the stolen source code in the illegal software. Intelligent software plagiarisms are based mainly on test-based analysis and structure. The proposed techniques use many methods: similarity identification, clone detection, software birthmark investigation, and software bug analysis [12]. Software plagiarism based on structure technique focuses on the basic structure of the source code, graph behavior, function call graph, and syntax trees. These methods do not catch the attack if another type of programming can preserve the same behavior as the original software.

Providing secure IoT networks is the purpose of many malware detection and intelligent software plagiarism techniques. Infecting the privacy of IoT nodes, smartphones, and computer systems is the goal of malware attacks. The different ways to detect malware are: Statistic identification analysis and dynamic identification analysis. The second one learns malware patterns when the code is executed in real-time. The malware is detected considering function parameters' exploration, function calls, visual investigation of codes, dataflow, and instruction traces. Some detection tools based on the dynamic behavior of malicious codes such as Anubis, TT analyzer, and CW Sandbox [13] are provided online. These tools, characterized by the monitoring of every dynamic behavior, suffer from the time-consuming issue. Statistic methods attempt to capture the layout information without real-time execution. As a statistic method, the signature identification technique detects windows-based malware via specification signatures as opcode frequency, string signature, and control flowgraph. Statistic methods are supported by disassembling tools to extract the hidden patterns from binary executables [14]. Byte sequence technique is considered a statistic method and removes n-byte sequences from patterns.

A. Software Piracy Detection

In most software plagiarism cases, the software is written on a single programming language and crackers change the control flow using a similar programming language. Authors in [15] proposed a method-based software benchmark to detect threats in java source code. The authors retrieved structural features by extracting the control flow of source codes. Then the similarity is computed between benchmarks of two source

Corresponding author: Khalid Aldriwish

codes. Authors in [16] attempted to acquire the similarity between codes through a hybrid approach. The compiler level features technique was considered. The authors performed an unsupervised learning approach to detect plagiarism. The proposed method computes the similar functionalities of different sources codes. Authors in [17] introduced an approach to achieve the similarity features between C++ and C source codes. The proposed method was based on a source forager search engine to extract features of every code. The control flow of the source code is identified based on the shape functionalities of the code. Authors in [18] computed the difference between two source codes using a logic-based approach. The semantics for differences were captured using symbolic execution and preconditions techniques. Authors in [19] tried to detect plagiarism related to student's assignments using the Latent Semantic Analysis (LSA) method. The authors aimed to compare source codes with regard to syntactic structures. This objective was achieved by combining LSA with PlaGate to identify the similarity. Then, the syntax tree detected the syntactic view and the abstract of the source code. Authors in [20] attempted to detect similar source code fragments using the parse tree kernel. The authors focused on Java files and proved that the achieved results were inaccurate. Therefore, a fingerprinting method was proposed instead of the parse tree kernel. Authors in [21] introduced a behavioral approach called BPlag to detect source code plagiarism. The behavior was extracted using symbolic execution. Then, the code was assimilated to a novel graph-based format. The plagiarism was computed according to these graphs. The authors proved that their approach was more accurate and more robust to plagiarism-hiding than 5 source code plagiarism tools. Authors in [22] presented a combined approach, using a Greedy String Tiling and Explicit Semantic Analysis method named EsaGst. The proposed method supported the detection of source code plagiarism independently from the programming language. The evaluation was conducted using different languages, including Java, C++, Python, PHP, and Java-Script. The results proved the good performance of the EsaGst approach.

B. Malware Detection

Malware detection is an open scientific topic. Authors in [23] used a machine learning approach to classify worms from the binaries of benign files based on a sequence of variable length instructions. The authors built a dataset including 1330 benign files and 1444 worms. The experimental results achieved a classification accuracy of around 96%. Authors in [24] combined the SVM-based machine learning and N-opcode sequences to detect malware. The detection process included the critical instruction sequence and cosine similarity. Findings demonstrated that similar malware possessed common core signatures. The proposed malware detection method achieved an accuracy around 98%. Authors in [25] introduced a method based on the dynamic analysis approach to highlight the limits of the static analysis approach. The author proposed an obfuscation model to execute binary samples and identify significant behavioral features within a virtual machine. The evaluation proved the insufficiency of the static analysis approach in the case the malware is obfuscated. Authors in [26] attempted to automate malware detection by identifying abnormal behavior within the program. The proposed idea provides little information about malicious behavior. Authors in [27] applied a classification approach based on the clustering method. The aim was to classify malware samples based on behavioral features. The proposed method was added to the Anubis system to track malware samples. The tracking report detailed the in-depth activities of the malware samples. Authors in [28] aggregated between statistic and dynamic analysis to detect malware accurately. The statistical analysis managed the operational codes based on frequency occurrences. The dynamical analysis executed traces of system calls and executable files. Tacking the advantages of each approach, the authors achieved a better result than statistic or dynamic case separately. Authors in [29] utilized Convolutional Neural Networks for malware detection. The announced purpose was to reduce time, size, and resource overhead. The proposed method used image-based malware for classification with 98.5% accuracy. Authors in [30] tried to detect malware using the image similarity technique. The authors employed benign and vision research lab datasets to evaluate their method. Samples from executable files were converted to binary code. In the testing phase, the accuracy reached 98%. Authors in [31] established an enhanced method to detect malware variants based on the deep learning approach. The authors intended to obtain high accuracy at a low time cost. The suggested process transformed the malicious code into a grayscale image, then classified samples using the CNN based on significant features. The authors overcame the imbalance of data by applying the bat algorithm. According to the experimental results based on the research lab dataset, the computed accuracy and speed were sufficient. Authors in [32] applied a machine learning method with processor core events to detect malware with a hardware event counter to ensure the detection. The purpose was to detect the SPECTRE using on-chip hardware on time. The proposed hardware architecture was based on software agents. To predict malicious activity, the authors used several machine learning classifiers. The predictive results achieved an accurate detection.

The current paper's contributions can be summarized as follows:

- A Deep Learning approach based on the TensorFlow Deep Neural Network is proposed to detect software piracy through source code plagiarism.
- A Deep Convolutional Neural Network (DCNN) is proposed to detect malware through binary visualization.
- The two models are combined into the same architecture for the IoT case.
- The proposed architecture is evaluated according to adequate datasets.

II. THE PROPOSED ARCHITECTURE

Figure 1 describes the proposed architecture, which is composed of malware and software piracy detection. Cloud data storage faces many kinds of attacks. Cyber security threat samples are stored in 4 databases. Database 1 contains network traffic data. Database 2 stores a list of earlier known malware data. Database 3 comprises new signatures and features of

detected malware. Database 4 includes pirated software via IoT devices. These databases contain a massive amount of data and request a lot of computational time and cost. Firstly, the network traffic data database (D1) transmits raw data to the detection module, which identifies the kind of attack (malware or software piracy). The classification is ensured based on learning signatures stored in databases D2 and D4.

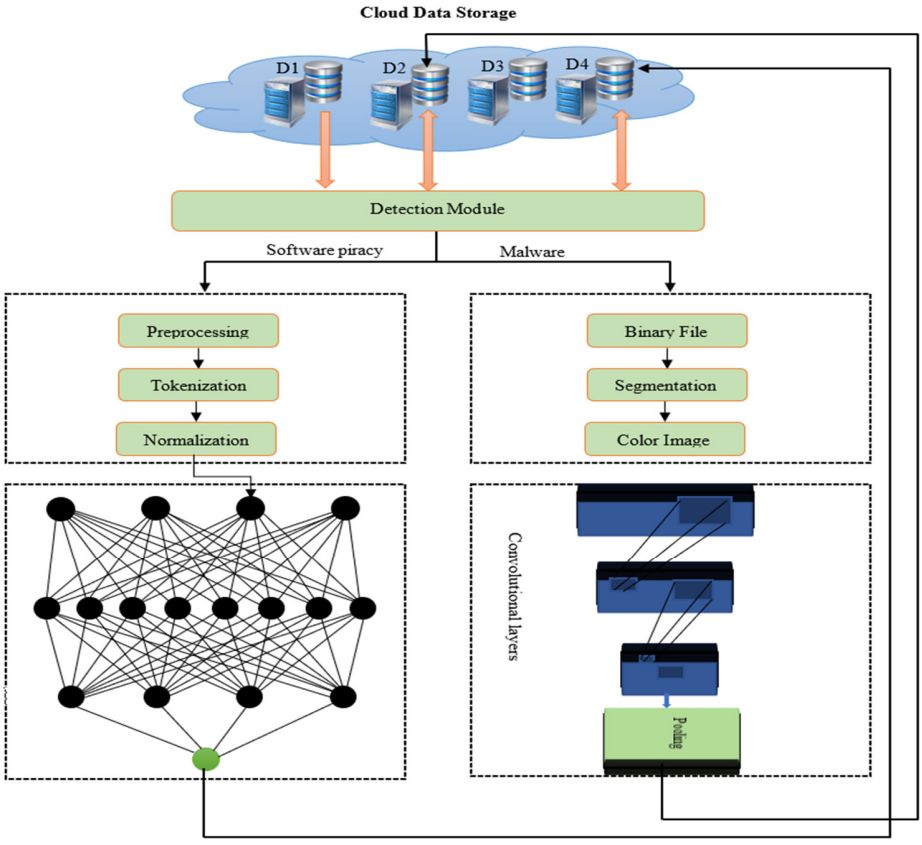

Fig. 1. The proposed architecture for malware and software piracy.

A. Software Piracy Threat Detection Model

The proposed software piracy attack detection is based on deep learning. The detection based on plagiarism methodology is captured through different types of source codes. The pirated version uses the same logic as the original software. Once the traffic data are classified as software piracy, the source codes are tokenized to decrease the dimensions of the data. In that step, significant features are extracted using the TensorFlow framework. The Keras API for deep learning is applied to capture source code plagiarism. The first database (D1) includes the network traffic data collected from Google Code Jam (GCJ) database. The D1 data are built from 100 programmers and contain about 400 source code documents. Software piracy threat detection is preceded by a preprocessing step. The purpose of this step is to divide the source code into small pieces. Then, the semi-code is converted into useful information, and noise is removed. Meaningful tokens are obtained during the tokenization step. Finally, the contribution of each token is zoomed through a weighting mechanism, see (1), based on the Logarithm of Term Frequency algorithm [33] and the Term Frequency and Inverses Document Frequency (TFIDF).

$$W(t, Doc, DS) = TF(t, Doc) \times IDF\ (t, DS) \quad (1)$$

where t defines the token, Doc defines a document, DS represent all the documents used in the dataset, TF defines the Term Frequency function, and IDF defines the Inverse Document Frequency function.

Deep learning is conducted by Tensorflow, which is used for high-level computations. The pirated software is identified through extracted similar codes. A fully connected network with dense layers is utilized for input and output data. The first layer, which contains 100 neurons, receives the data. The second layer is composed of 50 neurons. The third layer consists of 30 neurons. The output variable uses the fourth dense layer to identify the target of the plagiarized code. Deep learning is able to solve the overfitting problem using the drop out layer. The pattern is computed based on the rectifier (ReLU) activation method [34]:

$$f(x) = x^{+} = \max(0, x) \quad (2)$$

where x defines the input of the equivalent neurons.

The multi-class problem is conducted using the sigmoid method defined by:

$$S(x) = \frac{1}{1+e^{-x}} \quad (3)$$

The deep learning approach used to detect software piracy provides the following benefits: (1) the model is trained automatically, (2) the design supports various computational types, (3) services are reliable while updating and extending the model, and (4) the proposed model supports big networks.

B. Malware Threat Detection Model

The proposed malware threat detection model consists of two steps: preprocessing and deep convolutional neural network. Raw binary files generate the color images and the problem is becoming an image classification problem. The adopted color system is grayscale, and features are extracted from the color image. A feature reduction method is used to enhance the classification performance. It aims to reduce the feature set. The generation of the color image from a binary file proceeds as follows: (1) generate the hexadecimal strings, (2) divide the hexadecimal strings into a chunk of 8-bit vector, (3) convert each 8-bit vector to a two-dimensional matrix, and (4) plot the two-dimentional space. Then, the Deep Convolutional Neural Network (DCNN) is utilized to identify the malware. The DCNN receives training images. The Convolution layer's purpose is to reduce noise and enhance signal features. It reduces the over-fitting problem. The convolutional layer performs the computations using (4):

$$x_j^n = f(\sum_{i \in M_j} x_j^{n-1} * k_{ij}^n + b_j^n \quad (4)$$

where f defines the activation function, M is the cluster of given maps, b_j^n presents the bias consistent, and k_{ij}^n denotes the convolution kernel.

The accuracy of the proposed DCNN is improved through the convolutional kernel width. The pooling layer ensures the reduction of the data overhead and selects useful information. It minimizes the consequence of image distortion using (5):

$$x_j^n = f(Pool(x_j^{n-1}) + b_j^n) \quad (5)$$

where *Pool*() ensures the pooling task.

The classification of the output of the pooling layer is performed at the fully connected layer. It aims to enhance the model by reducing the over-fitting issue. The noise is removed using filters. Then, the training of the proposed DCNN is performed using Softmax-Cross-Entropy loss [35], as defined by:

$$L = -\log(\frac{e^{r_k}}{\sum_k e^{r_k}}) \quad (6)$$

where r_k denotes the rank of the k class. The learning of the parameters attempted to minimize the loss is conducted with the use of the Adam optimizer.

III. EXPERIMENTAL PART

A. Software Piracy Detection Performance Evaluation

The evaluation is based on the code similarity between the pirated software and the source software using the GCJ dataset [36]. The similarity is checked using Codeleaks plagiarism tool [12]. The dataset is proceeded by the preprocessing step to provide the valuable tokens of each source code as root word, stemming, token's length, and token's frequency. Then, the TFIDF and LogTF algorithms are applied to conduct token weighting. The accuracy of the classification is improved according to the number of neurons. The evaluation is shown in Figure 2 based on validation accuracy, validation loss, and loss.

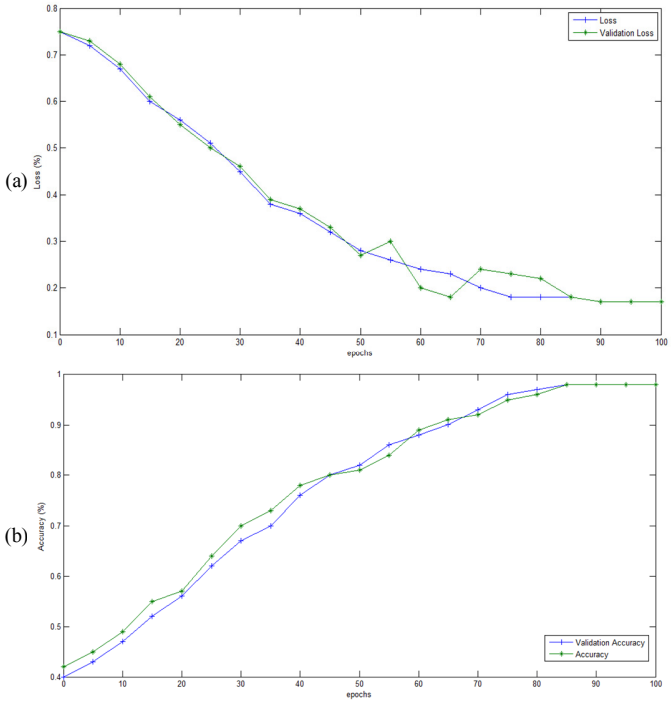

Fig. 2. (a) Loss and (b) accuracy results for source codes.

The loss curves (Figure 2) start from 0.75 and follow the same trend until 0.3. A fluctuation can be seen in the loss curve, but both curves are reduced in a similar way. From the accuracy curves, we can see that the proposed software piracy threat detection model achieved an accuracy of about 98%.

B. Malware Detection Performance Evaluation

The proposed model measures the effect of malware image ratios. The image size is taken as 180×180 and 196×196. We used the Leopard Mobile dataset [37], which is composed of 2486 benign and 14733 malware samples for evaluation. The training phase uses 15219 samples, and the testing phase employs 2000 samples. According to the experiments, the 196×196 dimension reached better accuracy than the 180×180 dimension. Therefore, the 196×196 ratio is more suitable for the proposed model. Table I highlights the comparison between the different dimensions based on the classification results. The image size of 196×196 achieves 98.12% testing accuracy. The computation time is about 18s. Figure 3 presents the performance evaluation related to the 196×196 image size based on training accuracy, training loss, test accuracy, and test loss metrics. A comparison with previous works is presented in Table II.

TABLE I. CLASSIFICATION RESULTS

Ratio	Precision (%)	Recall (%)	Accuracy (%)	Time (s)
180×180	94.15	94.68	94.47	16s
196×196	98.02	97.88	98.12	18s

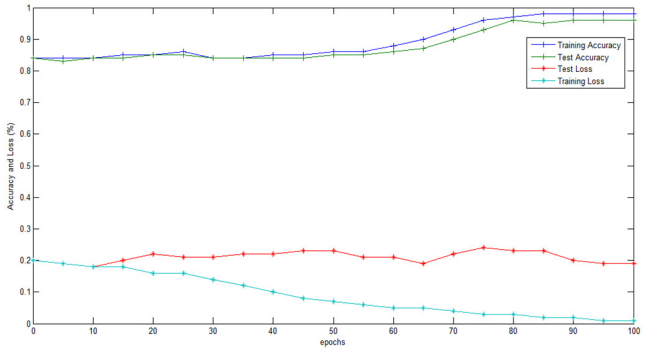

Fig. 3. Loss and accuracy results for malware detection.

TABLE II. MALWARE DETECTION APPROACHES COMPARISON

Ref	Method	Year	Approach	Accuracy (%)	Time (s)
[38]	SVM+GIST	2017	ML	86.1	46
[39]	SVM+LBP	2018	ML	78.05	27
[40]	SVM+CLGM	2019	ML	92.06	21
Proposed	DCNN	2021	DL	98	18

The comparison in Table II proves that the proposed malware threat detection model is more accurate than previous studies that are based only on the machine learning approach. The proposed method requires only 18s of computation time.

IV. CONCLUSION

Recent industrial systems migrate to industrial-based IoT to support new network services. Many security issues are related to IoT networks, especially malware threats and software piracy. Accurate cyber security defending IoT big data is needed. In this paper, a new security architecture based on the deep learning approach is proposed. The attempt aims to detect malware attacks and pirated software. The proposed approach is a combined methodology to detect threats. A Deep Learning approach based on the TensorFlow Deep Neural Network is introduced to detect software piracy through source code plagiarism. Then, a DCNN is utilized to detect malware through binary visualization. The findings prove that the proposed combined approach of DCNN and TFDNN achieves a classification accuracy of about 98%. The results of the proposed approach are better than the results obtained by related works. Speeding up the computation time to support real-time systems can be the purpose of future work. This target could be reached by proposing a high-security level hardware accelerator.

REFERENCES

[1] J. Huang, J. Chai, and S. Cho, "Deep learning in finance and banking: A literature review and classification," *Frontiers of Business Research in China*, vol. 14, no. 1, Jun. 2020, Art. no. 13, https://doi.org/10.1186/s11782-020-00082-6.

[2] M. B. Ayed, "Balanced Communication-Avoiding Support Vector Machine when Detecting Epilepsy based on EEG Signals," *Engineering, Technology & Applied Science Research*, vol. 10, no. 6, pp. 6462–6468, Dec. 2020, https://doi.org/10.48084/etasr.3878.

[3] M. Ben Ayed, A. Massaoudi, and S. A. Alshaya, "Smart Recognition COVID-19 System to Predict Suspicious Persons Based on Face Features," *Journal of Electrical Engineering & Technology*, vol. 16, no. 3, pp. 1601–1606, May 2021, https://doi.org/10.1007/s42835-021-00671-2.

[4] M. Ramzan, M. S. Farooq, A. Zamir, W. Akhtar, M. Ilyas, and H. U. Khan, "An Analysis of Issues for Adoption of Cloud Computing in Telecom Industries," *Engineering, Technology & Applied Science Research*, vol. 8, no. 4, pp. 3157–3161, Aug. 2018, https://doi.org/10.48084/etasr.2101.

[5] H. E. Fazazi, M. Elgarej, M. Qbadou, and K. Mansouri, "Design of an Adaptive e-Learning System based on Multi-Agent Approach and Reinforcement Learning," *Engineering, Technology & Applied Science Research*, vol. 11, no. 1, pp. 6637–6644, Feb. 2021, https://doi.org/10.48084/etasr.3905.

[6] S. S. T. Alatawi *et al.*, "A New Model for Enhancing Student Portal Usage in Saudi Arabia Universities," *Engineering, Technology & Applied Science Research*, vol. 11, no. 3, pp. 7158–7171, Jun. 2021, https://doi.org/10.48084/etasr.4132.

[7] T. Brito, J. Queiroz, L. Piardi, L. A. Fernandes, J. Lima, and P. Leitão, "A Machine Learning Approach for Collaborative Robot Smart Manufacturing Inspection for Quality Control Systems," *Procedia Manufacturing*, vol. 51, pp. 11–18, Jan. 2020, https://doi.org/10.1016/j.promfg.2020.10.003.

[8] F. Musumeci *et al.*, "An Overview on Application of Machine Learning Techniques in Optical Networks," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1383–1408, 2019, https://doi.org/10.1109/COMST.2018.2880039.

[9] C. R. Srinivasan, B. Rajesh, P. Saikalyan, K. Premsagar, and E. S. Yadav, "A Review on the Different Types of Internet of Things (IoT)," *Journal of Advanced Research in Dynamic and Control Systems*, vol. Volume 11, no. 1, pp. 154–158, 2019.

[10] Y. B. Zikria, R. Ali, M. K. Afzal, and S. W. Kim, "Next-Generation Internet of Things (IoT): Opportunities, Challenges, and Solutions," *Sensors*, vol. 21, no. 4, Jan. 2021, Art. no. 1174, https://doi.org/10.3390/s21041174.

[11] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Android Malware Detection using Deep Learning on API Method Sequences," *arXiv:1712.08996 [cs]*, Dec. 2017, Accessed: Oct. 07, 2021. [Online]. Available: http://arxiv.org/abs/1712.08996.

[12] F. Ullah, J. Wang, M. Farhan, M. Habib, and S. Khalid, "Software plagiarism detection in multiprogramming languages using machine learning approach," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 4, 2021, Art. no. e5000, https://doi.org/10.1002/cpe.5000.

[13] A. Orgah, A. Case, and G. Richard, "MemForC: Memory Forensics Corpus Creation for Malware Analysis," in *Proceedings of the 16th International Conference on Cyber Warfare and Security*, Jan. 2021.

[14] V. Raja, "Introduction to Reverse Engineering," in *Reverse Engineering: An Industrial Perspective*, V. Raja and K. J. Fernandes, Eds. London, UK: Springer, 2008, pp. 1–9.

[15] H. Lim, H. Park, S. Choi, and T. Han, "A method for detecting the theft of Java programs through analysis of the control flow information," *Information and Software Technology*, vol. 51, no. 9, pp. 1338–1350, Sep. 2009, https://doi.org/10.1016/j.infsof.2009.04.011.

[16] J. Yasaswi, S. Kailash, A. Chilupuri, S. Purini, and C. V. Jawahar, "Unsupervised Learning Based Approach for Plagiarism Detection in Programming Assignments," in *Proceedings of the 10th Innovations in Software Engineering Conference*, Feb. 2017, pp. 117–121, https://doi.org/10.1145/3021460.3021473.

[17] V. Kashyap, D. B. Brown, B. Liblit, D. Melski, and T. Reps, "Source Forager: A Search Engine for Similar Source Code," *arXiv:1706.02769 [cs]*, Jun. 2017, Accessed: Oct. 07, 2021. [Online]. Available: http://arxiv.org/abs/1706.02769.

[18] F. Zhang, D. Wu, P. Liu, and S. Zhu, "Program Logic Based Software Plagiarism Detection," in *2014 IEEE 25th International Symposium on*

Software Reliability Engineering, Naples, Italy, Nov. 2014, pp. 66–77, https://doi.org/10.1109/ISSRE.2014.18.

[19] G. Cosma and M. Joy, "An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 379–394, Mar. 2012, https://doi.org/10.1109/TC.2011.223.

[20] J.-W. Son, T.-G. Noh, H.-J. Song, and S.-B. Park, "An application for plagiarized source code detection based on a parse tree kernel," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1911–1918, Sep. 2013, https://doi.org/10.1016/j.engappai.2013.06.007.

[21] H. Cheers, Y. Lin, and S. P. Smith, "Academic Source Code Plagiarism Detection by Measuring Program Behavioral Similarity," *IEEE Access*, vol. 9, pp. 50391–50412, 2021, https://doi.org/10.1109/ACCESS.2021.3069367.

[22] T. Foltýnek, R. Všianský, N. Meuschke, D. Dlabolová, and B. Gipp, "Cross-Language Source Code Plagiarism Detection using Explicit Semantic Analysis and Scored Greedy String Tilling," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, New York, NY, USA, Aug. 2020, pp. 523–524, https://doi.org/10.1145/3383583.3398594.

[23] M. Siddiqui and M. C. Wang, "Detecting Internet Worms Using Data Mining Techniques," *Journal of Systemics, Cybernetics and Informatics*, vol. 6, no. 6, pp. 48–53, 2008.

[24] B. Kang, S. Y. Yerima, K. Mclaughlin, and S. Sezer, "N-opcode analysis for android malware classification and categorization," in *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)*, London, UK, Jun. 2016, https://doi.org/10.1109/CyberSecPODS.2016.7502343.

[25] A. Moser, C. Kruegel, and E. Kirda, "Limits of Static Analysis for Malware Detection," in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, FL, USA, Dec. 2007, pp. 421–430, https://doi.org/10.1109/ACSAC.2007.21.

[26] M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior," in *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, New York, NY, USA, Sep. 2007, pp. 5–14, https://doi.org/10.1145/1287624.1287628.

[27] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, Behavior-Based Malware Clustering," 2009.

[28] I. Santos, J. Nieves, and P. G. Bringas, "Semi-supervised Learning for Unknown Malware Detection," in *International Symposium on Distributed Computing and Artificial Intelligence*, 2011, pp. 415–422, https://doi.org/10.1007/978-3-642-19934-9_53.

[29] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, France, Feb. 2018, https://doi.org/10.1109/NTMS.2018.8328749.

[30] R. Kumar, Z. Xiaosong, R. U. Khan, I. Ahad, and J. Kumar, "Malicious Code Detection based on Image Processing Using Deep Learning," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, New York, NY, USA, Mar. 2018, pp. 81–85, https://doi.org/10.1145/3194452.3194459.

[31] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang, and J. Chen, "Detection of Malicious Code Variants Based on Deep Learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018, https://doi.org/10.1109/TII.2018.2822680.

[32] R. Oshana, M. A. Thornton, E. C. Larson, and X. Roumegue, "Real-Time Edge Processing Detection of Malicious Attacks Using Machine Learning and Processor Core Events," in *2021 IEEE International Systems Conference (SysCon)*, Vancouver, Canada, Apr. 2021, https://doi.org/10.1109/SysCon48628.2021.9447078.

[33] J. H. Paik, "A novel TF-IDF weighting scheme for effective ranking," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, Jul. 2013, pp. 343–352, https://doi.org/10.1145/2484028.2484070.

[34] T. Georgiou, Y. Liu, W. Chen, and M. Lew, "A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision," *International Journal of Multimedia Information Retrieval*, vol. 9, no. 3, pp. 135–170, Sep. 2020, https://doi.org/10.1007/s13735-019-00183-w.

[35] M. Ben Ayed, S. A. Alshaya, and A. Alshammari, "Enhanced heart rate estimation based on face features," in *2021 18th International Multi-Conference on Systems, Signals Devices (SSD)*, Monastir, Tunisia, Mar. 2021, pp. 840–844, https://doi.org/10.1109/SSD52085.2021.9429508.

[36] A. Back and E. Westman, "Comparing programming languages in google code jam," Chalmers University of Technology, University of Gothenburg, Gothenburg, Sweden, 2017.

[37] T. H.-D. Huang and H.-Y. Kao, "R2-D2: ColoR-inspired Convolutional NeuRal Network (CNN)-based AndroiD Malware Detections," in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, Dec. 2018, pp. 2633–2642, https://doi.org/10.1109/BigData.2018.8622324.

[38] A. D. Moore, *Intellectual Property and Information Control: Philosophic Foundations and Contemporary Issues*. New Brunswick, NJ, USA: Routledge, 2004.

[39] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, Nov. 2018, https://doi.org/10.1016/j.neunet.2017.12.012.

[40] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, "Malicious code detection based on CNNs and multi-objective algorithm," *Journal of Parallel and Distributed Computing*, vol. 129, pp. 50–58, Jul. 2019, https://doi.org/10.1016/j.jpdc.2019.03.010.