A Visual Analytics Framework for Explainable Malware Detection in Edge Computing Networks

Dilara T. Uysal
Birkbeck College, University of London, UK
duysal01@student.bbk.ac.uk

Shimaa Naser
Khalifa University, Abu Dhabi, UAE
shimaa.naser@ku.ac.ae

Zaid Almahmoud
Birkbeck College, University of London, UK
z.almahmoud@bbk.ac.uk

Sami Muhaidat
Khalifa University, Abu Dhabi, UAE
sami.muhaidat@ku.ac.ae

Paul D. Yoo*
Birkbeck College, University of London, UK
p.yoo@bbk.ac.uk

Abstract—**The emergence of new technologies for the fifth/sixth generation (5G/6G) wireless networks has led to the development of new services, resulting in an increase in malicious activities and cyber-attacks targeting various networklayers. Edge computing, a crucial technology enabler for 6G, is expected to facilitate traffic optimisation and support new ultra-low latency services. By integrating computing power from supercomputing servers into devices at the network edge in a distributed manner, edge computing can provide consistentquality-of-service, even in remote areas, which will drive the growth of associated applications. However, the complex environment created by edge computing also poses challenges for detecting malware. Therefore, this paper proposes a novel approach to malware detection using explainability via visualization and a multi-labelling technique. An object detection algorithm is used to identify malware families within the dataset which is created by emphasizing key regions. Using features from different malware categories in an image, this model displays a thorough malware recipe. Our experiments using real malware data demonstrate that identifying malware by its visible characteristics can significantly improve the interpretability of the detection process, enhancing transparency and trustworthiness.**

Keywords—6G, edge computing, crowd sensing/sourcing, cloud computing, machine learning, malware detection, explainability.

I. INTRODUCTION

The increasing connectivity of the world has given rise to a new interconnected world of people and digitalised things. In this environment, sensors are deployed to collect real-time data that is integrated into the physical environment and monitored, analysed, and assessed. However, this new environment will present brand-new challenges, including an explosive growth of data volume and complexity from new sources, as well as security concerns for connected fifth generation (5G) physical things. These concerns have motivated the initiation of recent exploratory studies of the sixth generation (6G) networks. As the evolution of service-based architectures into 'service everywhere' with 6G dramatically increases the number of associated applications, malware is also expected to become more sophisticated [1]. Attacks of various dimensions and network technologies necessitate more sophisticated and trustworthy defence mechanisms.

Recent advancements in analytical security and defence solutions for 6G networks have enabled progress in managing, operating, and optimising the underlying IoT services. However, cloud/edge-based services that require a third party to process 6G-enabled IoT data pose serious security and privacy issues, as data may be released or altered by malware when the cloud/edge is infected (i.e., data poisoning) [2]. To address these challenges, a new architecture for 6G is needed that could be developed through digital twins (DTs). DTs make use of a virtual representation of the 6G physical system, including the algorithms, computing, communication, and security technologists. Then, automated feedback loops are provided between physical things and their digital equivalents such that the behaviour of DT-enabled 6G could be affected, altering their specified states [3]. The authors in [2] proposed an integrated framework utilising blockchain and deep learning for securing data integrity and authenticity. It was demonstrated that the proposed blockchain-based data transmission system, employing smart contracts, can improve detection performance and make it easier to build a virtual environment to simulate and duplicate security-critical IoT activities. Additionally, in [4], the authors introduced a novel depth-wise efficient attention module (DEAM) in combination with Densenet for identifying malware samples and improving detection performance.

The available mechanisms for defending against malware operate in a narrow manner, identifying the attacking goals of the malware developers, such as Adware, Spyware, or Ransomware [5]-[7]. However, categorising malware is not as straightforward as distinguishing between black and white colours due to the varied paths and attacking chains employed to achieve these goals. In this regard, exploring the development of a multi-label solution using machine learning to identify malware based on their unique characteristics or "recipes" is an unexplored area. Each class of malware has

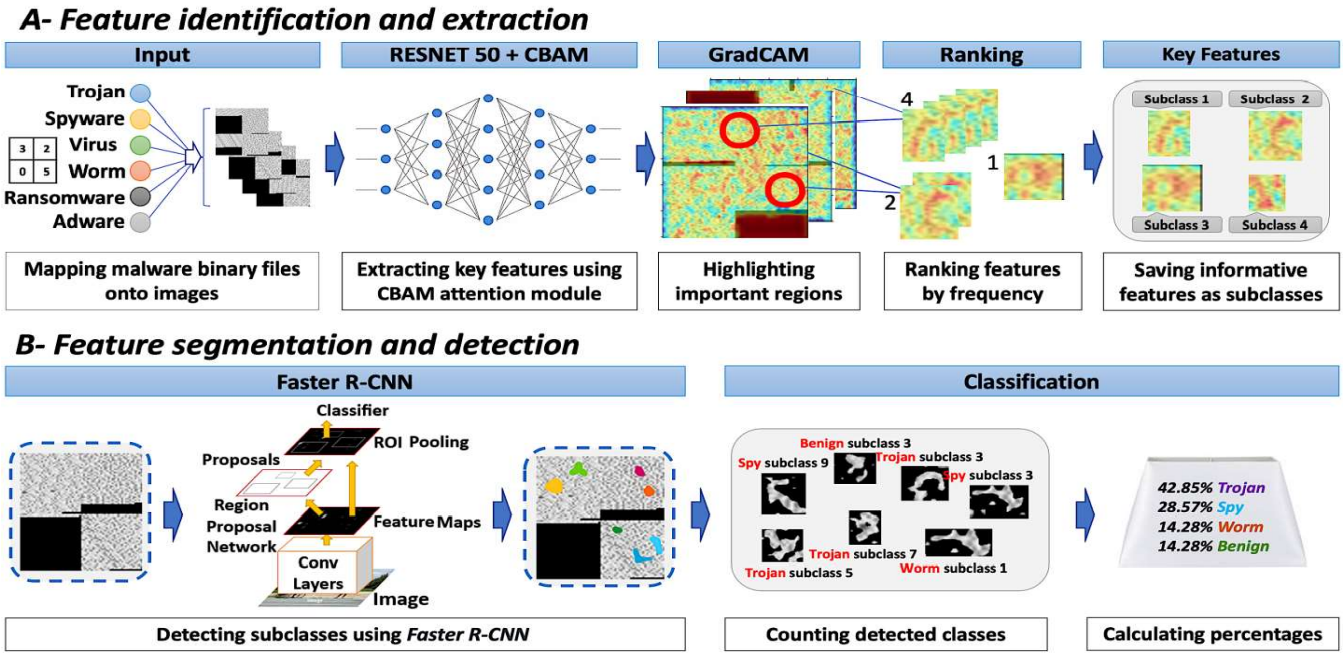

Fig. 1 The architecture of our framework including feature identification, extraction, segmentation and detection.

specific distinguishing features, and if these regions could be segmented in a malware image, samples from the same class should exhibit common patterns. By using a generalisation of the class activation mapping technique to extract the significant regions for each type of malware image, common features can be identified and fed into a model that learns to recognise the malware's "recipes" or DNA. This approach provides a better degree of transparency in the decision-making process. Thus, the aim of this work is to propose a novel *'explainability via visualization'* concept utilising a multi-labelling approach. Our experimental results on real malware data demonstrate that identifying malware by its visible recipes can significantly improve the interpretability of the malware decision process.

The rest of this paper is organised as follows. Section II proposes a novel *'explainability via visualization'* concept utilising a multi-labelling approach that aims to address the gap in the literature and discusses the construction of the framework. Section III summarises the experimental results obtained in the frame of research concerning the improvement of transparency in the decision process. In Section IV, a brief conclusion is provided with suggestions for future research direction.

II. METHODOLOGY

The proposed framework incorporates various components that are essential to explore the potential of the explainability concept in the malware detection process. The architectural design combines image generation from raw malware data, object identification and generated dataset for the extracted objects, and object detection and classification into a single system. This integration enables directtranslation from the generated output images. Fig. 1 provides an overview of the entire framework's architecture.

A. Malware Visualisation

In the context of malware detection, feature extraction iscommonly performed using static and dynamic analysis techniques. While static analysis can extract features without executing the malware, dynamic analysis is more resistant to obfuscation techniques and can provide insights into the goals of the malware. Table I shows the features that can be extracted using both methods. For Static analysis, the two extraction methods listed are assembly code-based and PE structure-based. The extracted features for assembly code- based analysis are opcodes, while for PE structure-based analysis, the extracted features include imports, exports, strings, DLLs, CFG, and API. For dynamic analysis, the extraction methods include system resource-based, machineactivity-based, and function call-based. The extractedfeatures for system resource-based analysis are CPU and memory usage. For machine activity-based analysis, theextracted features include File, registry, network, and process. Finally, for function call-based analysis, the extracted features include system calls and API calls.

Recently, image-based malware detection has gained much popularity. Fig. 2 shows an image of the Dontovo. A, which downloads and executes arbitrary files. It is noted thatthe malware image exhibits distinctive features in different sections of its binary fragments. Thus, in our proposed framework, a malware analysis report file is converted into agrayscale image in the range [0, 255], with each string in thereport mapped to an integer. By converting API call features into images, the resulting visual patterns provide a more effective way of managing and analysing malware behaviour,as similar examples of the same type have similar patterns that can be easily identified.

TABLE I

FEATURE EXTRACTION METHODS

Analysis Type	Extraction Methods	Extracted Features
Static	Assembly code-based	Opcodes
	PE structure-based	Imports, exports, strings, DLLs, CFG, API
Dynamic	System resource-based	CPU, memory usage
	Machine activity based	File, registry, network, process
	Function call-based	System calls, API calls

It is worth mentioning that our approach involves a complete visualisation method where numerical inputs are transformed into images in order to ensure transparency in both learning and decision-making processes. As shown in Fig. 1.A, the input phase of our framework generates a dataset that is based on images, which is further illustrated in Fig. 3. The availability of a sufficient quantity of up-to-date samples for any category of malware is of utmost importance. The utilisation of data-intensive machine learning techniques enables the detection and identification of recently launched malware attacks. MalwareBazaar and VirusTotal are regularly updated databases that offer labelled samples of malicious software. Initially, we execute malware executable files extracted from reputable repositories like VirusTotal [8] (which contains around 200 million binaries) and MalwareBazaar [9] (with over 280,000 samples) on CapeSandbox, which facilitates the extraction and analysis of suspicious files, including network, registry, file system and process features. The generated log file contains a record of the malware analysis execution within the guest environment. Subsequently, by utilising various VirusTotal labelling engines such as Kaspersky, Avira, and ESET, we conduct voting for malware labels. The malware samples are then classified into six categories based on the type of malware they contain, namely spyware, trojans, adware, ransomware, worms, and viruses.

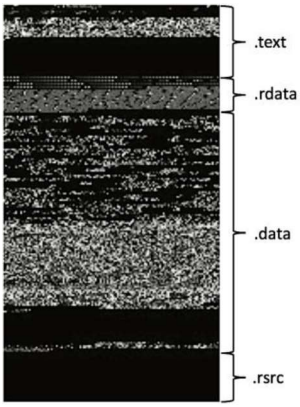

Fig. 2. Malware visualisation in grayscale

It is important to note that some API names are duplicated intentionally to deceive analysts. To address this concern, API names are grouped into 20 categories for each process and recurring API names are removed by including them only once in each group. Next, the features are converted into integers scaled between 0-255 from their corresponding strings. Each image is generated by combining individual images created for each feature, maintaining the same order. To accommodate varying file sizes, the image's width remains constant while the height is adjusted accordingly, as suggested by Nataraj et al. [10]. The merged image takes the form of 256 by 256, as depicted in Fig. 3. Any black areas within the image denote empty spaces, which can occur if the total number of pixels exceeds the overall size of the feature pixels.

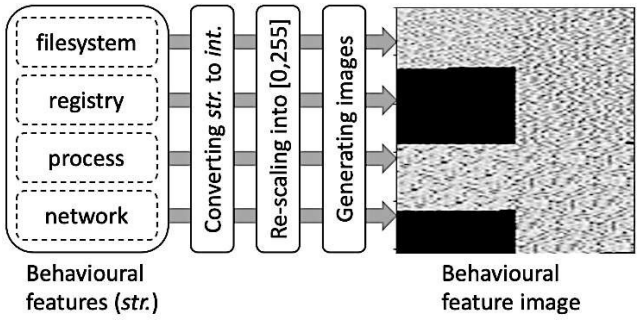

Fig. 3 Behavioral feature image generation.

B. *Object Detection and Malware Labelling*

When it comes to CNN training, attention modules are the superstars that selectively concentrate on informative features, leaving the irrelevant ones in the dust. But we didn't settle for just any attention module. We chose the convolutional block attention module (CBAM) [11], which is like a superhero with both spatial and channel-wise attention mechanisms, making it stand out from other similar modules. This makes it much better than the boring old channel-wise attention mechanism. It improves the model's ability to distinguish complex patterns and features throughout the input data by highlighting both of these features at the same time.

The focus of channel attention is on what is meaningful in the context of an input image. A multi-scale feature is used as the input, and average-pooled features and max-pooled features are first generated in each channel. The output features are then combined using element-wise summation after being sent to shared fully connected layers with both features. As a last step, the activation function is used to derive the channel attention weight. The channel attention process can be summarised as follows:

$$\mathbf{M_c}(\mathbf{F}) = \sigma(\text{MLP}\ (\text{AvgPool}\ (\mathbf{F})) + MLP(\text{MaxPool}\ (\mathbf{F})))$$
$$= \sigma\left(\mathbf{W_1}\left(\mathbf{W_0}\left(\mathbf{F^c_{avg}}\right)\right) + \mathbf{W_1}\left(\mathbf{W_0}\left(\mathbf{F^c_{max}}\right)\right)\right),$$

In contrast to channel attention, spatial attention focuses on where as an informative component that complements channel attention. A convolution layer is applied to the concatenated feature description to build a spatial attention map. Two pooling processes are used to aggregate the channel information of a feature map, producing two 2D maps (and). Both represent the channel's max- pooled and average-pooled features, respectively. A typical convolution layer concatenates and connects them to create the 2D spatial attention map. The following is an explanation of the spatial attention process:

$$\mathbf{M_s}(\mathbf{F}) = \sigma\left(f^{7\times7}([\text{AvgPool}(\mathbf{F})\,;\text{Max Pool}(\mathbf{F})])\right)$$
$$= \sigma < \left(f^{7\times7}\left(\left[\mathbf{F^s_{avg}};\mathbf{F^s_{max}}\right]\right)\right)$$

σ is sigmoid function and signifies a convolution operation with a 7×7 filter size. The input feature map is element-wise multiplied by the channel attention map and the spatial attention map, and the resulting feature maps are then concatenated to produce the output feature map.

The representations of the channel attention map and the spatial attention map are, respectively, Mc ∈ RC×1×1 and Ms ∈ R1×H×W. Following is an explanation of the entire attention process:

$$F' = Mc(F) \otimes F,$$

$$F'' = Ms(F') \otimes F',$$

where F stands for the intermediate feature map of size RC×H×W.

We used the ResNet50-CBAM backbone on our dataset, with each ResBlock containing CBAM integration. Our choice of ResNet-50 was motivated by the need to learn complex patterns and features from our dataset for pattern-based categorization problem. In the ResNet50 architecture, the CBAM module is inserted after each residual block. Through the use of skip connections, the residual blocks in ResNet50 enable the network to learn residual mappings, which aid in resolving the vanishing gradient issue.

To improve the discriminative ability of our deeply learned features, we turned to Wen et al. [12], who proposed a joint loss function that uses both softmax loss and center loss supervision. This loss function was our go-to error for the learning process. The following is the formulation of the softmax loss function:

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}$$

$xi \in \mathbb{R}d$ signifies the i^{th} deep characteristic of the class y_i and is the feature dimension. $Wj \in \mathbb{R}d$ represents the j^{th} column of weights $W \in \mathbb{R}d \times n$ in the final fully connected layer, while the bias term is denoted $b \in \mathbb{R}n$. The size of the mini batch is m and the number of classes is n. The softmax loss simply separates the underlying characteristics of distinct classes.

It is essential for the centre loss to efficiently concentrate all of the deep features of the same class into the centre. The idea, intuitively, is to minimise intra-class variations while maintaining the properties of various classes separately. The centres are calculated after each iteration by averaging the features of the relevant classes. Second, to avoid massive perturbations produced by a few mislabelled data, the learning rate of the centres is controlled by utilising a scalar factor α. The following equation formulates the centre loss function, which is defined as follows:

$$\mathcal{L}c = \frac{1}{2}\sum_{i=1}^{m} \left\| x_i - c_{y_i} \right\|_2^2$$

Here, c_{y_i} is the centre of the y_i-th class, and x_i is the i^{th} deep feature.

With joint supervision, not only are inter-class feature differences increased, but also intra-class feature variations are reduced. The equation below provides the joint supervision formula:

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_c$$

$$= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} + \frac{\lambda}{2}\sum_{i=1}^{m} \left\| x_i - c_{y_i} \right\|_2^2$$

For balancing the two loss functions, a scalar λ is employed.

According to Fig. 1.A, the output of Grad-CAM [13] came along and showed us the model prediction biases. With the help of Grad-CAM, we were able to identify "where" and "what" the learned model was focusing on. And guess what? The models we learned using Grad-CAM have shown great capabilities of highlighting critical areas, which we then used as potential objects of classes.

Grad-CAM assigns relevance values to each neuron for a specific decision of interest using the gradient information streaming into the final convolutional layer of the CNN. The gradient of the score for class c, y^c (before the softmax),is first computed with respect to the feature map activations A^k of a convolutional layer, represented as $\partial y^c / \partial A^k$ in order to obtain the class-discriminative localisation map Grad-CAM $L^c_{\text{Grad-CAM}} \in \mathbb{R}^{u \times v}$ for any class c with dimensions u and v.

The neuron importance weight α_k^c are obtained by global-average-pooling these gradients flowing back over the width and height dimension as follow:

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

The deep network downstream from A has been partially linearized by the weight α_k^c which evaluates the significance of a feature map k for a target class c. A ReLU is attained by performing the weighted combination of forward activation maps given below.

$$L^c_{\text{Grad-CAM}} = \text{Re}LU \underbrace{\left(\sum_k \alpha_k^c A^k \right)}_{\text{linear combination}}$$

Fig. 4 provides a visual representation of our feature extraction process, highlighting the critical feature regions that the model utilises. We then employ sliding windows to compare these extracted regions with those of other images of the same malware type. By grouping similar items together, we can identify the top four subclasses for each malware category based on the highest number of elements. Using these results, we have created our own object detection dataset

in COCO format [14] for malware detection, focusing solely on essential features. It is worth noting that currently, no malware dataset exists for object detection.

Object detection is the primary method used to locate objects within an image, allowing us to search for specific extracted features that are characteristic of various types of malware. Our multi-labelling process employs Faster RCNN [15] with DarkNet to identify regions within an image. By analysing the scoring labels of these regions, we can discover the recipe for a given malware type, as shown in Fig. 5.

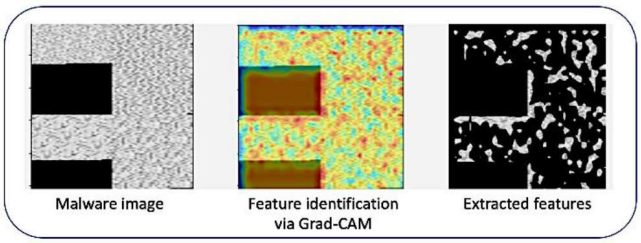

Fig. 4 Malware feature identification and extraction

III. Results and Discussions

Malware detection has long relied on signature-based analysis, which involves maintaining a database of known malware signatures. While this approach has been effective, recent research has introduced machine learning models for image-based detection, which can lead to decisions that are too complex for human comprehension. Unfortunately, this lacks of transparency in machine learning-based methods can impact the trustworthiness of the model and its decision-making process. To address this issue, we propose a new approach that focuses on exposing details of the decision process to users. This approach reduces the psychological distance between users and the model and highlights the main features of a task, which can be used to improve performance. We believe that our approach can significantly improve the interpretability and trustworthiness of machine learning-based malware detection systems.

Our approach to detecting malware families involves leveraging an object detection model on our generated dataset. This model is able to effectively and precisely detect relevant feature regions present in malware images, allowing us to discover the most informative and frequently occurring features for each malware type. Fig. 5 illustrates the results of this process, where the predicted version of an adware image includes detections of worm, ransomware, and spyware features, while the predicted version of a spyware image includes detections of worm and adware features. By using the proportion of detected features for each malware type, we are able to classify the malware images and record the final percentage of malware characteristics at the end of the detection process. This methodology not only provides accurate classification, but also provides a deeper understanding of the features that are characteristic of each malware type.

Consequently, the object detection model can uncover a malware recipe that consists of features from multiple types of malware in a given image. Through deductive reasoning, these extracted feature regions can be linked to the corresponding binary codes, offering insights into the structure and behaviour of each malware type. By reconstructing the source code and studying its functionality, this "*explainability via visualization*" method can enhance future detection techniques.

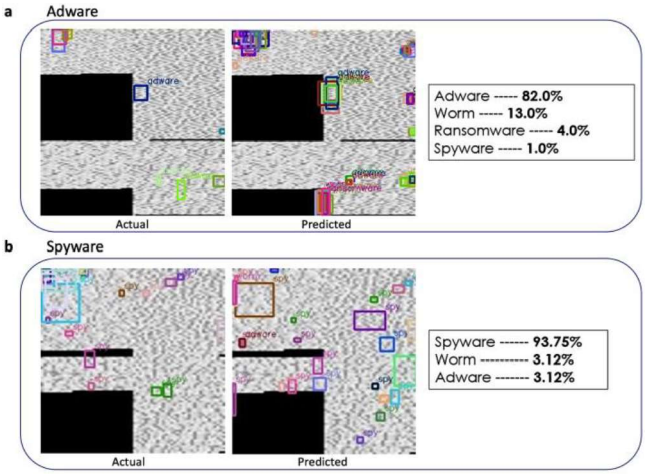

Fig. 5 Malware detection by recipes and classification results.

Furthermore, this approach has broader applications beyond malware detection. Unlike current machine learning tools that are highly specialised, our approach can facilitate the development of artificial general intelligence by promoting reasoning, building common sense knowledge, and planning. It has the potential to unlock new pathways for solving complex problems and advancing research in various domains.

IV. Concluding Remarks

Malware is expected to become more sophisticated and harder to detect in the upcoming 6G wireless networks with edge computing capabilities. To address this issue, a novel approach of "explainability via visualization" has been proposed in this paper, along with a comprehensive framework comprising various data-analytic components to achieve transparency in the malware detection process. This approach not only overcomes the limitations of traditional signature-based methods but also enhances the reliability of recent machine-learning-based techniques by providing transparency.

Furthermore, we have shown how the current monocular approach to malware detection can be extended to achieve a higher level of transparency through the effective segmentation of specific features of individual malware types. The class activation mapping technique, combined with an object detection model and a region proposal network, has proved to be effective in fine-tuning the modeland identifying the recipes in malware images. The deductivemapping of the identified regions to the corresponding binaries has provided valuable insights into how each malware is formed. It is essential to note that this approach isnot restricted to malware detection tasks, and the concept can be applied to various intelligent domains.

Acknowledgment

The authors are grateful to Dr Hussam Al-Hammadi at the Centre for Cyber-Physical Systems (C2PS) for his invaluable discussions and feedback, and special thanks to the EBTIC, British Telecom's (BT) security team in London and Abu Dhabi for their constructive criticism on this work.

Special thanks to Angelina McDonald who proof-read the manuscript.

REFERENCES

[1] D. T. Uysal, P. D. Yoo and K. Taha, "Data-driven malware detection for 6G networks: A survey from the perspective of continuous learning and explainability via visualisation," in *IEEE Open Journal of Vehicular Technology*, 2022, doi: 10.1109/OJVT.2022.3219898.

[2] P. Kumar, R. Kumar, A. Kumar, A. A. Franklin, S. Garg and S. Singh, "Blockchain and Deep Learning for Secure Communication in Digital Twin Empowered Industrial IoT Network," in *IEEE Transactions on Network Science and Engineering*, 2022, doi: 10.1109/TNSE.2022.3191601.

[3] S. Suhail, Z. Sherali, J. Raja, H. Rasheed, M. Raimundas, and S. Davor, "Security attacks and solutions for digital twins," 2022, *arXiv preprint arXiv:2202.12501.*

[4] C. Wang, Z. Zhao, F. Wang, and Q. Li, "A Novel Malware Detection and Family Classification Scheme for IoT Based on DEAM and DenseNet," *Security and Communication Networks*, Jan, 2021.

[5] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," *Proceedings of the sixth ACM conference on data and application security and privacy,* 2016.

[6] M. Kalash, et al. "Malware classification with deep convolutional neural networks," 2018 9th IFIP international conference on new technologies, mobility and security (NTMS). IEEE, 2018.

[7] B. Yuan, J. Wang, D. Liu, W. Guo, P. Wu, and X. Bao, "Byte-level malware classification based on markov images and deep learning," *Computers & Security*, vol. 92, p. 101740, May 2020, doi: 10.1016/j.cose.2020.101740.

[8] "VirusTotal" Virustotal.com, 2019. https://www.virustotal.com/gui/home/upload

[9] "MalwareBazaar | Malware sample exchange," bazaar.abuse.ch. https://bazaar.abuse.ch/

[10] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*, 2011, pp. 1–7.