Journal Homepage: - www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI: 10.21474/IJAR01/19953
DOI URL: http://dx.doi.org/10.21474/IJAR01/19953

RESEARCH ARTICLE

INNOVATION IN CYBER THREAT DETECTION: TRANSFORMER-BASED APPROACH

Djè Bi Djè Guy Gabin[1,3], Dr. Diako Doffou Jérôme[1,2], Dr. Kanga Koffi[1,2] and Prof. Oumtanaga Souleymane[1,3]

1. Computer Science and Telecommunications Research Laboratory (LARIT).
2. African Higher School of ICT (ESATIC).
3. Félix Houphouët-Boigny National Polytechnic Institute, Polytechnic Doctoral School, Engineering Science and Technology, Mathematics and Digital Science, Yamoussoukro. (INPHB, EDP, STI, MSN)

Manuscript Info

Manuscript History
Received: 17 September 2024
Final Accepted: 27 October 2024
Published: November 2024

Key words:-
Artificial Intelligence, Cyber-Security, Malware, Transformer

Abstract

Malware poses a major threat to cyber security. In fact, its increasing sophistication and rapid spread over the internet poses increasingly complex challenges. Modern malware uses advanced evasion strategies, often rendering traditional detection systems ineffective, especially against zero-day attacks. These challenges are amplified by complex obfuscation techniques, as well as the diversity of malicious behaviors, fueled by the daily creation of new malware. In the face of these threats, our study proposes an innovative approach using BERT and GPT-2 to improve malware detection. The main innovation of our method lies in the application of Transformers to analyze and identify complex behavioral signatures of malware, which improves the detection capability, particularly in terms of accuracy and generalization to new threats. The evaluation of our model on the CICMalDroid2020 dataset, as well as the comparison of the results obtained with similar works, demonstrate that BERT and GPT-2 offer significant advantages in terms of accuracy, robustness and generalization capacity against modern threats.

Introduction:-

Cybersecurity remains a critical issue in the digital age, with the increasing sophistication of malware that poses a constant threat (Aboaoja et al. 2022). Attack methods are evolving rapidly, making traditional detection systems increasingly vulnerable. In particular, modern malware is able to bypass these security measures, causing potentially devastating impacts (Alomari et al. 2023;Djenna et al. 2023).

Faced with these challenges, early and autonomous malware detection has become imperative. Our research is part of this dynamic by exploring the application of Artificial Intelligence (AI) to cyber-security. More precisely, we propose an innovative approach exploiting the capabilities of Transformer-type language models, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT-2 (Generative Pre-trained Transformer 2), for malware detection. These models, pre-trained on large text corpora, have already distinguished themselves by their effectiveness in the semantic understanding of natural language (Alam, Khan, and Alam 2020; Qin et al. 2023), but their potential for malware detection remains little explored.

Corresponding Author:- Dje Bi Dje Guy Gabin
Address:-Research Scholar, Félix Houphouët-Boigny National Polytechnic Institute, EDP, STI, MSN Doctoral School, Engineering Science and Technology, Mathematics and Digital Science, Yamoussoukro.

The novelty of our method lies in the joint use of BERT and GPT-2 to capture complex behavioral and semantic patterns characteristic of malware. Unlike traditional approaches that mainly focus on malware signatures or static analysis, our method stands out for its ability to detect emerging threats even in the presence of obfuscation and evasion techniques. This hybrid approach not only allows for better accuracy, but also better generalization to new malware variants, thus providing increased robustness to security systems.

The remainder of our paper is organized as follows:

1. Section 2:We will discuss related works and existing methods in the field of malware detection to our knowledge, highlighting their strengths and limitations.
2. Section 3:This section is devoted to our contribution. In it, we will present the methodology used, detailing the data preprocessing steps, the construction of the BERT and GPT-2 models, as well as the training strategies used.
3. Section 4:Here, we will evaluate the performance of our model using metrics such as accuracy, recall, precision, and F-score, and compare our results with those of other approaches.
4. Section 5:We will discuss the implications of our results, challenges encountered, and perspectives for future research.
5. Section 6:Finally, we will conclude with a summary of our contributions and recommendations for the development of more robust cyber-security systems.

State of the art:-

Malware detection remains a critical concern in the cyber-security field, and many approaches have been developed to address this challenge. In this state-of-the-art review, we will review current malware detection methods, highlighting recent advances and persistent challenges.

Malware Signatures

Traditionally, malware signatures have been widely used to identify specific patterns of malicious code. However, this approach is limited by its reactivity, being able to detect only already known malware.

Authors Karnik, Goswami, and Guha (2007)proposed a technique for detecting obfuscated malware based on signatures and cosine similarity measure. With a threshold of 0.97, the cosine similarity measure was calculated to cope with obfuscation techniques. Microsoft's .Net Framework using C++ allowed the experiments to be performed with obfuscated versions of Win32.Evol, Win32.Oroch and Win32.Evul. viruses and a variety of benign software versions.

Zolkipli and Jantan (2010)also proposed a framework for malware detection consisting of three main components, namely signature-based detection, genetic algorithm detection and signature generator. The model is capable of automatically generating signatures in order to enrich the malware detection signature base. The study focused on viruses, worms and Trojans.

Similarly, HuijuanZhu and al. (2020)presented SEDMDroid, a static malware detection framework. SEDMDroid has a two-tier architecture (the set of base learners with multi-layer perception (MLP) and the fusion of the base learners results by the support vector machine (SVM)). The experiments were performed on two separate datasets collected by static analysis. SEDMDroid achieves an accuracy of 89.07% for the first dataset. With the second dataset the average accuracy is 94.92%.

DroidAnalytics is a similar signature-based malware analysis system for Android, implemented by Zheng, Sun, and Lui (2013)The tool can automatically collect malware, generate signatures, identify malicious code segments, and associate the studied malware with various malware in the database. Extensive experiments using 150,368 Android apps correctly classified 2494 Android malware from 102 different families, with 342 zero-day malware samples from six different families.

Heuristic Analysis

Heuristics scans suspicious program behavior to detect potential malicious activity. While more proactive than signatures, they can generate false positives and are not always effective against polymorphic attacks.

Nguyen et al.(2018)have, in this context, proposed an automatic detection system of malware intrusions specifically backdoors. It is based on the modeling of user behaviors as well as a heuristic analysis method of mobile logs generated during the operation of mobile devices. This collected raw information is classified and processed using techniques such as feature selection, term weighting. Subsequently, machine learning algorithms such as support vector machine (SVM), logistic regression (LR) and artificial neural network (RNN) are applied to these data.

The comparative evaluation of the above models shows that the SVM obtains the highest values of Accuracy, Precision, Recall and F1 respectively 99.8%, 90.0%, 97.6% and 93.6%.

Authors (Alrammal, Naveed, and Rihawi 2018)also used a heuristic approach to create an anti-malware that is able to adapt to new malware definitions and detect them in real time. Indeed, they train the machine learning algorithm, called RBCM, based on a Monte Carlo simulation model named MOCART, to detect malicious activity on a mobile device. The dataset is automatically generated from malware they developed and existing malware. The experiments were performed on three dedicated Android devices. The results reveal that malware significantly influences the resource usage of a device when it infects it, thus the malicious behavior can be deduced. Moreover, they explain that the different definitions of malicious code work according to the same indicator models and that there is a consistency between the behaviors of malware.

Behavior Based Detection

This approach analyzes the real-time behavior of applications, enabling malware detection based on abnormal actions. Behavioral detection systems are effective against unknown attacks but can be circumvented by sophisticated evasion techniques.

Kim et al. (2022)proposed a malware detection system called MAPAS. Indeed, through the API call graphs, it analyzes the behaviors of malicious applications. They use convolutional neural networks (CNNs) to discover the common features of these malware API call graphs and the Jaccard similarity algorithm to detect malware with an accuracy of 91.27%.

Razeghi, Abadi, and Haniye(2013)presented MalHunter, a malware detection system based on the behavior generated automatically from different malware samples. The authors demonstrate that the proposed scheme is resistant to obfuscation techniques and is effective in detecting all types of polymorphic malware. The results of the experiments conducted allowed to affirm that by choosing 0.4 as the cluster radius and 0.05 as the similarity threshold, the system achieves a detection rate of 90.83% with a false alarm rate of 0.80%.

Dynamic analysis specifically of permissions also allowed the authorsHuang, Tsai, and Hsu (2013)to implement an Android malware detection tool . To do this, four machine learning algorithms namely AdaBoost, Naïve Bayes, Decision Tree (C4.5) and Support Vector Machine, were used to evaluate the relevance of permissions in malware detection using a dataset of 124,769 benign apps and 480 malicious apps. The experimental results show that the proposed framework can detect more than 81% of malicious samples.

In another paper, Mahindru and Sangal(2021b)studied a dynamic feature selection technique for detecting malware on Android using machine learning techniques, named FSdroid. This model uses a Least Square Support Vector Machine (LSSVM) connected to three distinct kernel functions, namely linear, radial basis, and polynomial. The experiments were performed using 200,000 distinct Android applications. The empirical results reveal that the model is capable of detecting 98.8% of malware with a detection time of 12 seconds.

Artificial Intelligence and Machine Learning

The integration of artificial intelligence and machine learning marks a significant advancement. Machine learning models, such as decision tree-based classifiers, have shown increased effectiveness in detecting malware by learning from dynamic and static features.

In this other paper, Sandeep (HR 2019), proposes a tool for detecting malware on Android using static analysis and deep learning.In fact, it uses on the one hand, the fully connected deep learning technique to identify the different types of permissions and on the other hand the random forest to establish a classification between malware and benign software.

The dataset used to evaluate the data model consists of 331 features and a classification label. It consists of benign samples collected from Google Play Store and various platforms and malware collected from Virsushare. The proposed model achieves an accuracy of 94.64%.

Authors Eskandari, Khorshidpour, and Hashemi (2013)further developed HDM Analyser, a hybrid analysis approach based on data mining techniques to detect malware. They demonstrate that HDM Analyser has a better overall accuracy of 95.27% and excellent time complexity than static or dynamic analysis methods.

Jerlin and Marimuthu (2018)also proposed a technique for malware detection in API call sequences based on Rete algorithm and Multidimensional Naive Bayes Clustering (MDNBS). According to the experiments, the Rete-MDNBS model has a detection accuracy of 88.7%, a recall of 6.8% and reduces the time consumption and computational complexity by 0.11 seconds.

Mahindru and Sangal (2021a)also used machine learning algorithms (i.e., deep learning algorithm, furthest-reach clustering, Y-MLP approach, and nonlinear decision tree) and dynamic approach to develop MLDroid, a web-based framework for detecting malware on Android devices. The results of the experiments, conducted on 500,000 Android applications, reveal that MLDroid achieves a detection rate of 98.8%.

Use of Neural Networks (NN)

Neural networks, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been successfully applied to extract complex representations from data, thereby improving detection capability.

Artificial Neural Networks (ANNs) have enabled Imtiaz et al. (2021)to propose DeepAMD, a real-world Android malware application detection mechanism.

They evaluate the effectiveness of DeepAMD and conventional machine learning classifiers (Decision Tree (J48), Naive Bayesian (NB), Sequential Minimal Optimization (SMO) and Multilayer Perceptron (MLP)) using the CICAndMal2019 dataset. Considering the experimental results on the static layer, DeepAMD achieves a maximum accuracy of 93.4%.

In a study by Alkahtani and Aldhyani(2022), machine learning and deep learning approaches have been implemented to detect malicious attacks targeting Android. The Android mobile benchmark datasets CICAndMal2017 and Drebin were used to test the cybersecurity system. Thus, the CNN-LSTM (convolution neural network-long short-term memory with an accuracy of 97.82%, is among the best for malware detection in the Android environment.

Djenna et al. (2023)also proposed a systematic approach to detect modern malware, using dynamic methods based on deep learning combined with heuristic approaches and the CICAndMal2017 dataset.

Specifically, they implemented two types of approaches using deep learning algorithms (Convolutional Neural Network (CNN) and Deep Neural Network (DNN)) for binary classification.

The model evaluation shows that the combination of behavior-based DNN and heuristic approach has an accuracy of 100%, and the behavior-based CNN and heuristic approach has an accuracy of 94%. On the other hand, these results are better than those obtained with the random forest (RF) and decision tree (DT) models which show an accuracy of 91% and 89% respectively.

Along the same lines, Alomari et al.(2023)proposed a malware detection system that uses deep learning and feature selection methods. They apply a correlation-based feature selection method to two different datasets to produce different datasets. These selected dataset versions will be used to train dense deep learning models and long short-term memory (LSTM)-based models. The experimental results indicate that their methodology yields a score of 98.38%, higher than the score of the work of Gumaa et al. (“Graph Approach for android malware detection using machine learning techniques” 2021)with a score of 97.77%.

In this paper (Almalehand al. 2023)present a new malware detection method based on a hybrid model trained using API call sequences. More precisely, the proposed hybrid model uses weights derived from logistic regression,

trained with a dataset composed of API call sequences, and integrates them as initial weights into a recurrent neural network (RNN) model to detect malware in different categories. The tests were carried out with two sets, one unbalanced and the second balanced (sub-sample). The first dataset composed of 40% more malware than benign software, includes 42,797 malicious API call sequences and 1,079 non-malicious API call sequences, each sequence consisting of the first 100 consecutive non-repeated API calls. The evaluation results allow to have the accuracies of 98% for the first set and 83% for the second. These values surpass the accuracies of the Xavier model, the reference model, which are 92% and 83% respectively, for both datasets.

A recurrent neural network (RNN) has on the other hand been proposed by Jha and al. (2020)to detect malware. They measure the effectiveness of RNN with "one-hot encoding feature vector", "random feature vector" and "Word2Vec feature vector" using hyper-parameters.

The experiments were conducted on a total of 10,000 datasets, including 5,000 malware and 5,000 benign datasets. They also performed a pairwise t-test with a 95% confidence level.

The results show that the vector obtained by the Skip-gram architecture of the trained Word2Vec model is effective with RNN for malware classification. Because its Area under the curve (AUC), f-1 score and accuracy values 91.69%, 91.75% and 91.92% respectively are the highest among the three feature vectors.

Nasser, Hasan, and Humaidi (2024)implemented DL-AMDet, a deep learning-based malware detection tool for Android. It exploits static features, using CNN-BiLSTM deep learning method, and dynamic features, using deep autoencoders, to detect malware.

The tests were performed with a dataset extracted from CICMalDroid2020 and Androzoo consisting of 21000 malicious and safe applications with a total of 3100 static features and 100 dynamic features per application. The proposed DL-AMDet architecture achieves an F-score of 99.935 thus outperforming related works.

Zhou et al. (2024)present a software supply chain malware detection model based on LSTM and SVM. Indeed, this model is characterized by two approaches namely the support vector machine (SVM) based on the use of Bayesian optimization (BO-SVM) to improve its hyperparameters and the support vector machine based on long short-term memory (LSTM-BO-SVM) to further improve the accuracy. Extensive experiments have been conducted on the balanced ClaMP dataset and the imbalanced CICMalDroid2020 dataset. The results obtained indicate that the accuracy of the LSTM-BO-SVM model on the two datasets is 98.2% and 98.6%, respectively, which is 2.9% and 2.2% higher than that of the BO-SVM model on these two datasets. Furthermore, the BO-SVM model outperforms other models in terms of accuracy.

Convolutional neural network (CNN) allowed Kiraz and Doğru (2024)to propose a technique for visualizing static features and classifying Android malware.

This approach is based on the use of image file of each feature, obtained by applying the BERT algorithm as a text classification algorithm for the numerical representation of textual features. The CICMalDroid2020 set was used for classification and learning operations. Evaluating the model, it was confirmed that the two-output CNN provided the best performance with an accuracy of 91%, an F1 score of 89%, a recall of 90% and a precision of 91%.

In this paper Kou et al. (2023)propose SSCL-TransMD, a semi-supervised continuous learning malware detection model based on Transformers.

The proposed model improves the LUMP (lifelong semi-supervised mixture algorithm) for feature memory replay and the LLGC (Learning with Local and Global Consistency) algorithm for pseudo-label acquisition.

In fact, LUMP is used throughout the life of the SSCL-TransMD model as a basis. Modifications are made to incorporate adaptive weight computation. Then enhanced LUMP is used for dynamic and proportional sampling of a mixture of labeled historical samples and new unlabeled input samples.

Furthermore, in order to obtain pseudo-labels, LLGC is used to iteratively calculate similarity scores for unlabeled samples in the mixture samples.

Finally the basic neural network is used as the detection component of the classification.

Moreover, experiments conducted on the CICMalDroid2020 datasets show that the SSCL-TransMD model outperforms other reference malware detection models. More precisely, for binary classification, the SSCL-TransMD method gives an F1-score of 0.8549. This value is higher than 0.8498 which is the highest F1-score among the reference methods (DroidOL: F1-score 0.8498; SSL–MD: F1-score 0.8375; DroidEvolver: F1-score 0.8483; PLDNN: F1-score 0.8412). The F1-score of SSCL-TransMD therefore presents an average improvement of 1.27%.

Furthermore, after introducing hybrid samples four times, the F1-score improves by an average of 1.96%. In this context DroidEvolver has an F1-score of 0.8298 compared to 0.8498 F1-Score of the SSCL-TransMD method.

AI, RN, and DL-based malware detection methods have been widely used and have enabled the implementation of detection systems that surpass the capabilities of traditional signature- and behavior-based techniques. However, some issues such as the increasing complexity of malware, poor performance of overfitting models, accuracy and efficiency issues in detecting next-generation malware further expose Internet users to cyber threats. As a result, organizations and individuals continue to fall victim to hackers, who exploit vulnerabilities and use cutting-edge techniques to evade existing anti-malware solutions. In our approach, we leverage the notable advances in Transformers in Large Language Models (LLM) to improve malware detection capabilities.

Proposed approach:-

Despite advances and innovations in malware analysis and detection, attacks remain numerous and destructive. Attackers and malware authors frequently exploit unknown vulnerabilities or bypass security updates to make their malware more effective. Malware detection systems are still under intense research as they play a crucial role in cyber-security. Our contribution is to propose an innovative approach leveraging the capabilities of Transformers-like language models, such as BERT and GPT-2, for malware detection, and to use a recent and comprehensive dataset, called CICMalDroid2020 ("MalDroid 2020 | Datasets | Research | Canadian Institute for Cybersecurity | UNB," n.d.), provided by the Canadian Institute for Cyber-security (CIC). Different evaluation metrics were used to demonstrate the performance of our malware detection approach.

Description of the approach

Dataset

CICMalDroid2020is a recent and comprehensive dataset provided by the CIC. It allows researchers and cybersecurity specialists to evaluate the robustness of implemented Android malware detection and classification systems. It is characterized by a large number of recent, sophisticated and complete Android samples. It includes static and dynamic features of 17341 APK samples composed of five distinct categories (adware, banking malware, SMS malware, risky software and benign software).

Table 1:- Description of the dataset.

Content	Number of samples
Adware	1253
Banking	2100
SMS malware	3904
Riskware	2546
Benign	1795
Not Analyzed	5743

The dataset made available to researchers includes 17,341 APK files, the capture of analysis logs performed with CopperDroid on 13,077 samples, and four CSV files describing 51,230 features extracted for 11,598 APK files. We implemented our model using the "future_vectors_syscallsbinders_frequency_5_Cat.csv" dataset, which contains 470 features extracted from 11,598 APKs.

Feature Preprocessing and Representation

Since the quality of the model depends on the data used, it is important to place particular emphasis on this step.

Consider the matrix X, of dimension $X \in R^{n \times m}$containing the characteristics of the samples, that is, the values that describe the files or processes analyzed.

with

- n: Number of samples, representing the total number of files or processes to analyze.
- m: Number of features per sample, representing attributes such as system call frequency, network indicators, or binary characteristics of files.

The data preprocessing phase consisted of:
- Separate features and labels;
- Label encoding;
- Create a label encoder (label_encoder)
- Apply the encoder to y to get y_encoded (the encoded labels)
- Standardize characteristics;
- Create a normalizer (scaler) for the features
- Fit the normalizer on X and transform X to get X_scaled (the normalized features)

Normalization helps ensure that all features are of comparable scale, preventing one large-scale feature from dominating others in training.
Each feature is standardized so that it has a mean of 0 and a standard deviation of 1, following the formula:

$$\mathbf{X_{ij}}=\frac{X_{ij}-\mu_j}{\sigma_j}$$

with:

- X_{ij}: Normalized value of characteristic j for sample i.
- μ_j: Average of the values of characteristic j.
- σ_j: Standard deviation of the values of characteristic j.

- Convert data to PyTorch tensors
- Convert X_scaled to PyTorch tensor (X_tensor) of type float32
- Convert y_encoded to PyTorch tensor (y_tensor) of type long (integer)
- We split our data into training data (70% of X_tensor size), testing data (15% of X_tensor size), and validation data (15% of X_tensor size). As mentioned below, Tables 2 and 3 detail the number of trainings, testing, and validation runs for each classification:

Table 2:- Data distribution of the binary classification dataset.

Class	train_dataset: 70%	val_dataset : 15%	test_dataset: 15%
Malware	6862.4	1471.45	1471.45
Benign	1256.5	269.25s	269.25

Table 3:- Data distribution of the multi-class classification dataset.

Class	train_dataset: 70%	val_dataset : 15%	test_dataset: 15%
Adware	877.1	187.95	187.95
Banking	1470	315	315
SMS malware	2732.8	586.6	586.6
Riskware	1782.2	381.9	381.9
Benign	1256.5	269.25	269.25

- Create DataLoaders for each set
- Create train_loader with train_dataset, batch_size of 32 and shuffle data (shuffle=True)
- Create val_loader with val_dataset, batch_size of 32, without shuffle of data (shuffle=False)
- Create test_loader with test_dataset, batch_size of 32, without shuffle of data (shuffle=False)

Model architecture

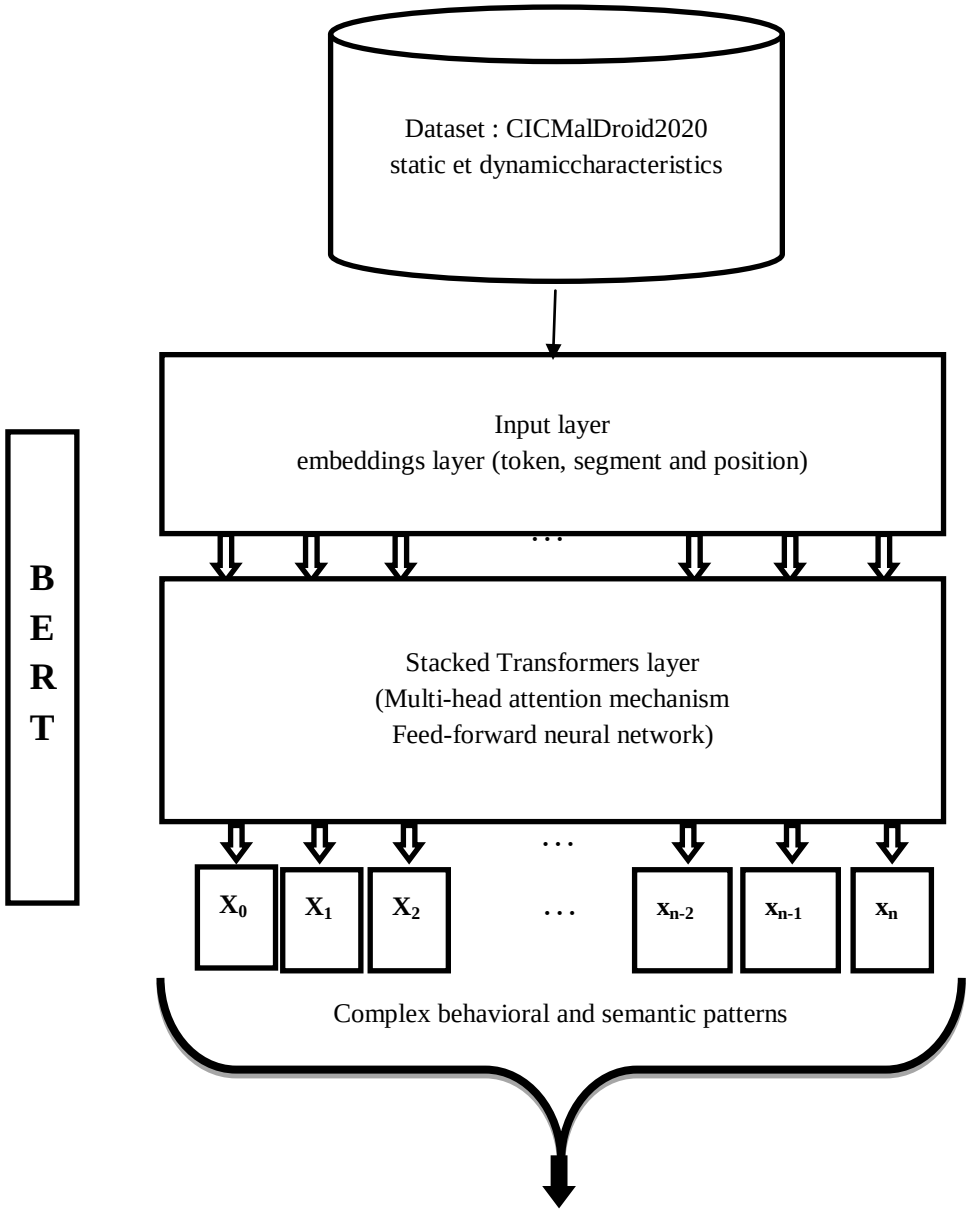
We implemented a Transformers-based model that uses a combination of architectures like BERT (Bidirectional Encoder Representations from Transformers) and GPT-2 (Generative Pre-trained Transformer) to capture complex behavioral and semantic patterns characteristic of malware in order to perform binary and multi-class clustering.

Given a dataset of Android E malware characteristics, the task is to develop a malware detection model. Mathematically, this classification problem can be formulated as follows.
Let E = {C1, C2, ..., Cn} represent the features of n APK files and L € {0,1} represent the software labels, where:

0 means the software is benign
1 means the software is malicious

The goal is to model a prediction function D that takes a feature vector "X(C)" of an APK file as input and predicts the file label,
that is, D(C) → {0, 1},
where:
D(C) = 0 if file "C" is predicted to be benign
D(C) = 1 if file "C" is classified as malicious

Functional diagram of our proposal.
As a first step, we applied the BERT algorithm to the CICMalDroid2020 dataset. BERT's contextual capabilities allowed us to extract relevant embeddings. This information was then used to feed the GPT-2 algorithm, whose predictive capabilities allowed us to determine the nature of a given file.



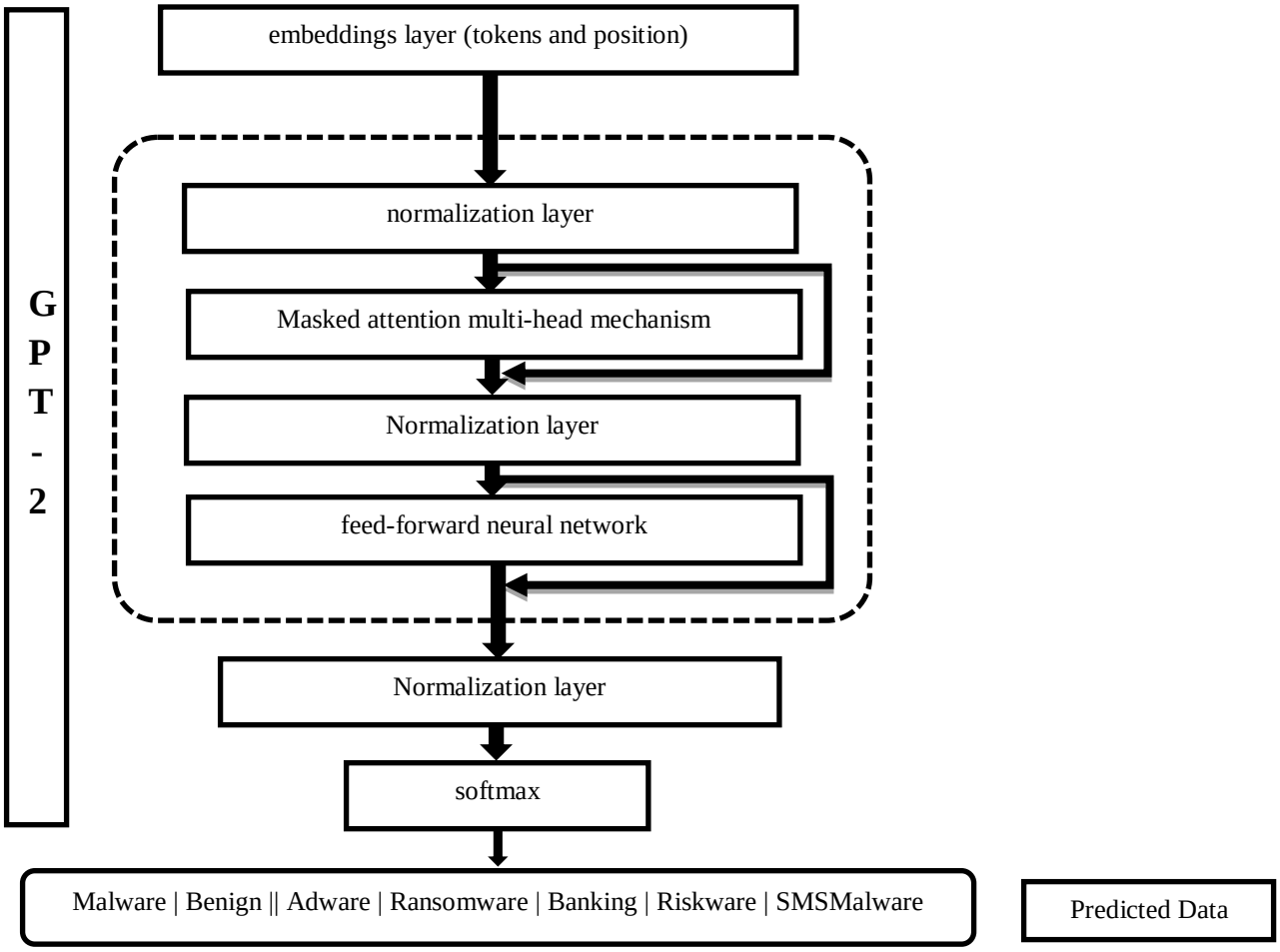


Fig 1:- System architecture based on BERT and GPT-2 for two classifications.

Description of the steps

Embedding Malware Characteristics

At this stage, richer representations of features are captured to enable the model to better distinguish malware from legitimate software.

$x_i \in R^m$ is the vector representing the characteristics of sample i.

Each vector x_i is projected into a higher-dimensional space to facilitate learning of complex relationships:

$x_i^{(e)} = W_e \cdot x_i + b_e$

where:

- $x_i^{(e)}$: Vector projected into a higher-dimensional space.
- W_e: Embedding weight matrix, of dimension d×m, where d is the new dimension.
- b_e: Bias vector, of dimension d.

Self-Attention Mechanism

This mechanism helps identify complex relationships between malware characteristics, such as correlations between system call sequences, by giving higher weights to significant relationships.

In the self-attention mechanism, each vector $x_i^{(e)}$ is transformed into three vectors:

$$Q = W_q \cdot x_i^{(e)},$$
$$K = W_k \cdot x_i^{(e)},$$
$$V = W_v \cdot x_i^{(e)}.$$

where:

- Q(query), K(key), V(value) are vectors representing different projections of the features.

- $W_q, W_k,$ and W_v are weight matrices for queries, keys, and values, respectively, of dimension d × d.

Calculation of self-attention:

Attention (Q, K, V) = Softmax$\left(\frac{QK^T}{\sqrt{dk}}\right)V$

where:

- QK^T: Dot product between queries and keys, measuring similarity.
- $d_{k:}$ Normalization term to stabilize gradients, with d_k as the dimension of the key vectors.
- Softmax: Function that transforms similarity scores into probabilities.

Transformer Encoder Structure

In this section, more complex patterns are extracted from malware characteristics, enhancing the model's ability to detect anomalies in software behavior.

Calculation steps:

- Adding self-attention:

$x^{(l+1)}$ = LayerNorm($x^{(l)}$ + Attention($Q^{(l)}$, $K^{(l)}$,$V^{(l)}$))

Where: $x^{(l)}$: Representation of the input vector at layer l.

- Application of the feedforward transformation:

$x^{(l+1)}$ = LayerNorm($x^{(l+1)}$ + FeedForward($x^{(l+1)}$))

Where: FeedForward: Two-layer neural network with ReLU activation, capturing non-linear transformations.

- **Loss Function and Training**

Minimize the loss to improve the model's ability to correctly classify malware.

The loss function measures the prediction error:

$L(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$

with:

- y : Actual sample label (1 for malware, 0 for legitimate).
- $\hat{y}$: Predicted probability of being malware.

- Gradient Descent:

$\theta_{t+1} = \theta_t - \eta \nabla_\theta L$

with:

- $\theta_{t:}$ Model parameters at iteration t.
- η: Learning rate, controlling how fast the weights are updated.
- $\nabla_\theta L$: Gradient of the loss function with respect to the model parameters.

Experimental result and discussion:-

Evaluation of results

In this section the performance of our model is evaluated based on different performance measures such as true positive rate (TPR), false positive rate (FPR), confusion matrix, accuracy, precision, recall and f-measure.

The main aspects to consider when measuring performance are:

- TP (true positive): malware correctly classified as such;
- TN (true negative): normal correctly classified as benign;
- FP (false positive): normal incorrectly classified as malware;
- FN (false negative): malware incorrectly classified as benign.

True Positive Rate (TPR)

The TPR (true positive rate) is defined as the proportion of malware correctly identified by the classifier. It is calculated as follows:

$$\mathbf{TPR} = \frac{\mathbf{TP}}{\mathbf{TP + FN}}$$

True Positive Rate (FPR)

Similarly, the FPR (false positive rate) is defined as the proportion of legitimate software incorrectly identified as malicious by the classifier. It is calculated as follows:

$$\mathbf{FPR} = \frac{\mathbf{FP}}{\mathbf{TN + FP}}$$

Precision: Precision, defined as the positive predictive value, is mainly used to return relevant results. Additionally, it is a measure of accuracy that indicates the number of classes predicted correctly, calculated as follows:

$$\textbf{Accuracy} = \frac{TP}{TP + FP}$$

F-Measure: The f-measure combines precision and recall by giving them equal weight. High values of precision, recall, and f-measure indicate effective classification. The f-measure is calculated as follows:

$$\textbf{F} - \textbf{Measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

Recall: Recall, also known as true positive rate or sensitivity, represents the ability to detect all positive cases. In our case, it is the percentage of malware identified as such among all malware in the dataset. It is calculated as follows:

$$\textbf{Recall} = \frac{TP}{TP + TN}$$

Area Under the ROC Curve (AUC): evaluates the model's ability to distinguish between malware and legitimate software:

$$\textbf{AUC} = \int_0^1 TPR(t)dt$$

Experimental result and discussion

CICMalDroid2020dataset Specifically, our model achieved an accuracy of 90.75%in the final tests. This accuracy demonstrates that our approach is robust and capable of effectively detecting modern malware.

- **Key performance metrics:**
- Accuracy (overall precision): 90.75%
- Recall: 91%
- F1 score (F measure): 91%
- AUC (area under the curve): 98%
- **Precision curve - recall by class**

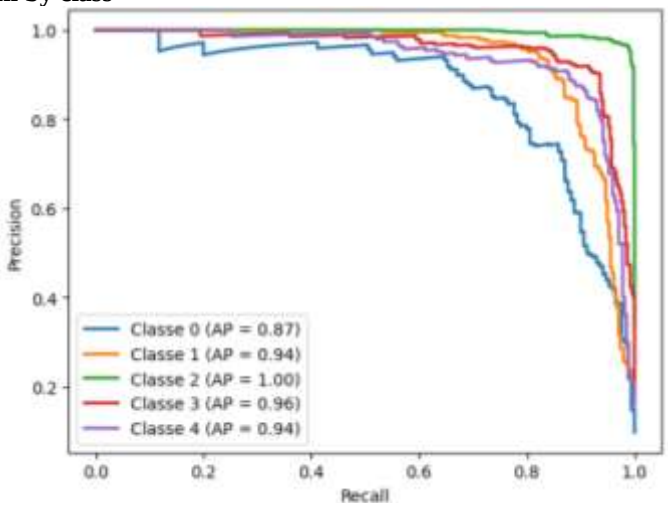


Fig 2:- Precision-recall curve by class.

- **Result of multi-class classification with BERT and GPT-2**

	precision	recall	f1-score	support
0	0.83	0.78	0.80	170
1	0.86	0.87	0.87	338
2	0.95	0.99	0.97	585
3	0.91	0.91	0.91	375
4	0.91	0.85	0.88	273
accuracy			0.91	1741
macro avg	0.89	0.88	0.89	1741
weighted avg	0.91	0.91	0.91	1741

Fig 3:- Multi-class classification result.

Regarding the confusion matrix**,** our results show a good classification of malware and benign software, with few false positives and false negatives, which demonstrates the effectiveness of our model in a complex environment where zero-day attacks and obfuscation techniques are increasingly common.

Comparison with the work of Zhou et al. (2024)
The work of Zhou et al. (2024) proposed a hybrid model based on LSTM and SVM, achieving an accuracy of 98.6%on the same dataset. However, while their model demonstrated high accuracy, we observe that our Transformers-based approach offers a significant advantage in generalization. Unlike LSTM, which uses ordered temporal sequences, our model better handles long-distance dependencies, which are crucial for capturing similar malware behaviors.

We also observed that Transformers enable better handling of polymorphic attacks and malware using evasion techniques, where LSTM-SVM models may face limitations due to the need to frequently optimize their hyperparameters.

ComparisonwithKiraz et al. (2024)
Kiraz et al. (2024) used CNNs to visualize static features of malware, achieving an accuracy of 91%. Although CNNs are effective in extracting spatial features, they lack the flexibility to capture the complex semantic dependencies that characterize modern malicious behaviors.

Our approach, which combines the contextual capabilities of BERT and the predictive capabilities of GPT-2, stands out for its ability to capture more complex behavioral and semantic patterns. This combination allows us to improve robustness against obfuscation and evasion techniques that CNNs, mainly oriented towards static analysis, do not always manage to detect effectively.

Comparison with Kou et al. (2023)
The work of Kou et al.(2023), with their **SSCL-TransMD model** for continuous semi-supervised learning, achieved an F1 score of 0.8549for binary classification. In comparison, our hybrid BERT-GPT-2 approach achieved an F1 score of 0.91**,** which highlights the ability of our model to outperform semi-supervised methods in malware classification.

We believe that using BERT and GPT-2 together in our approach allowed us to capture deeper behavioral patterns and better generalize against previously unseen attacks. This strengthens our ability to handle continuous learning scenarios where new malware variants emerge frequently.

Table 4:- Comparative study of AI-based malware detection.

Source	Dataset	Model	Accuracy	Recall	score F1
Our proposal	CICMalDroid2020	BERT - GPT -2	90.75%	91%	91%
Kiraz et al. (2024) (Kiraz and Doğru 2024)	CICMalDroid2020	CNN (Static Feature)	91%	90%	89%
Z hou et al. **(2024)** (Zhou et al. 2024)	CICMalDroid2020	LSTM-BO-SVM	98.6%	99.8%	93.6%
Kou et al.(2023) (Kou et al. 2023)	CICMalDroid2020	SSCL-TransMD	84.98%	85%	84.83%

In summary, our experimental results show that our Transformer-based approach offers significant advantages over existing methods. We demonstrated that our model is not only accurate but also able to generalize to emerging threats and obfuscation techniques. Unlike traditional approaches such as CNNs or LSTMs, our model can capture deeper contextual relationships in malware data, thus providing a more robust and flexible solution for malware detection.

We believe this approach can be effectively applied in real-time detection systems, particularly the need for frequent model retraining and providing better protection against next-generation malware. This flexibility and generalization capability make it an ideal solution to strengthen cyber security in an environment where threats are increasingly sophisticated and diverse.

Conclusion:-

Malware detection is a topic of increasing importance in the field of cyber security. As malware continues to evolve and become more sophisticated, it can easily bypass existing detection systems, and often cause devastating damage. To counter this threat, it is necessary to develop more effective detection methods. Our study proposes an innovative approach using BERT and GPT-2 to improve malware detection. By comparing our method with those of Zhou et al., Kiraz et al., and Kou et al., we showed that Transformers offer significant advantages in terms of accuracy, robustness, and generalization against modern threats. These results confirm that transformers represent a major advancement for proactive malware detection, and we are convinced that their integration into current cyber security systems will provide greater resilience against emerging cyber threats.

To pursue this research, we consider several promising avenues to improve and extend the Transformers-based approach in malware detection. First, optimizing Transformers for sequential malware data could be explored. Although BERT and GPT-2 have shown good performance, it would be interesting to test more specialized transformers models, such as temporal or hierarchical transformers. This will allow to better capture temporal or hierarchical dependencies present in API call sequences and complex malware behaviors.

Another promising avenue would be to extend this approach to cross-platform environments. Malware is not limited to Android devices, and evaluating our model on various operating systems, such as Windows, iOS, or Linux, allows us to test its generalization ability in cross-platform contexts, thereby improving its usefulness in global cyber security.

Finally, integrating other behavioral or dynamic data sources, such as system logs or network traces, could further improve the robustness of the model by enriching its understanding of malicious behaviors across multiple attack vectors. These perspectives offer interesting avenues to strengthen the resilience and efficiency of malware detection systems in the future.

Reference:-

1. Aboaoja, Faitouri A., Anazida Zainal, Fuad A. Ghaleb, Bander Ali Saleh Al-rimy, Taiseer Abdalla Elfadil Eisa, and Asma Abbas Hassan Elnour. 2022. “Malware Detection Issues, Challenges, and Future Directions: A Survey.” Applied Sciences 12 (17): 8482. https://doi.org/10.3390/app12178482.
2. Alam, Tanvirul, Akib Khan, and Firoj Alam. 2020. “Bangla Text Classification Using Transformers.” arXiv. https://doi.org/10.48550/arXiv.2011.04446.
3. Alkahtani, Hasan, and Theyazn H. H. Aldhyani. 2022. “Artificial Intelligence Algorithms for Malware Detection in Android-Operated Mobile Devices.” Sensors 22 (6): 2268. https://doi.org/10.3390/s22062268.
4. Alomari, Esraa Saleh, Riyadh RahefNuiaa, Zaid Abdi AlkareemAlyasseri, Husam Jasim Mohammed, Nor Samsiah Sani, Mohd Isrul Esa, and Bashaer Abbuod Musawi. 2023. “Malware Detection Using Deep Learning and Correlation-Based Feature Selection.” Symmetry 15 (1): 123. https://doi.org/10.3390/sym15010123.
5. Alrammal, Muath, Munir Naveed, and Samer Rihawi. 2018. “Using Heuristic Approach to Build Anti-Malware.” In 2018 Fifth HCT Information Technology Trends (ITT), 191–96. https://doi.org/10.1109/CTIT.2018.8649499.
6. “Applied Sciences | Free Full-Text | Malware API Calls Detection Using Hybrid Logistic Regression and RNN Model.” n.d. Accessed February 19, 2024. https://www.mdpi.com/2076-3417/13/9/5439.
7. Djenna, Amir, Ahmed Bouridane, Saddaf Rubab, and Ibrahim Moussa Marou. 2023. “Artificial Intelligence-Based Malware Detection, Analysis, and Mitigation.” Symmetry 15 (3): 677. https://doi.org/10.3390/sym15030677.
8. Eskandari, Mojtaba, Zeinab Khorshidpour, and Sattar Hashemi. 2013. “HDM-Analyser: A Hybrid Analysis Approach Based on Data Mining Techniques for Malware Detection.” Journal of Computer Virology and Hacking Techniques 9 (2): 77–93. https://doi.org/10.1007/s11416-013-0181-8.
9. “Graph Approach for android malware detection using machine learning techniques.” 2021.(blog). October 31, 2021. https://www.hnjournal.net/2-11-15/.
10. HR, Sandeep. 2019. “Static Analysis of Android Malware Detection Using Deep Learning.” In 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 841–45. https://doi.org/10.1109/ICCS45141.2019.9065765.
11. Huang, Chun-Ying, Yi-Ting Tsai, and Chung-Han Hsu. 2013. “Performance Evaluation on Permission-Based Detection for Android Malware.” In Advances in Intelligent Systems and Applications - Volume 2, edited by

Jeng-Shyang Pan, Ching-Nung Yang, and Chia-Chen Lin, 111–20. Smart Innovation, Systems and Technologies. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-35473-1_12.

12. Imtiaz, Syed Ibrahim, Saif ur Rehman, Abdul Rehman Javed, Zunera Jalil, Xuan Liu, and Waleed S. Alnumay. 2021. "DeepAMD: Detection and Identification of Android Malware Using High-Efficient Deep Artificial Neural Network." Future Generation Computer Systems 115 (February):844–56. https://doi.org/10.1016/j.future.2020.10.008.
13. Jerlin, M. Asha, and K. Marimuthu. 2018. "A New Malware Detection System Using Machine Learning Techniques for API Call Sequences." Journal of Applied Security Research 13 (1): 45–62. https://doi.org/10.1080/19361610.2018.1387734.
14. Jha, Sudan, Deepak Prashar, Hoang Viet Long, and David Taniar. 2020. "Recurrent Neural Network for Detecting Malware." Computers & Security 99 (December):102037. https://doi.org/10.1016/j.cose.2020.102037.
15. Karnik, Abhishek, Suchandra Goswami, and Ratan Guha. 2007. "Detecting Obfuscated Viruses Using Cosine Similarity Analysis." In First Asia International Conference on Modelling & Simulation (AMS'07), 165–70. https://doi.org/10.1109/AMS.2007.31.
16. Kim, Jinsung, Younghoon Ban, Eunbyeol Ko, Haehyun Cho, and Jeong Hyun Yi. 2022. "MAPAS: A Practical Deep Learning-Based Android Malware Detection System." International Journal of Information Security 21 (4): 725–38. https://doi.org/10.1007/s10207-022-00579-6.
17. Kiraz, Ömer, and İbrahim Alper Doğru. 2024. "Visualising Static Features and Classifying Android Malware Using a Convolutional Neural Network Approach." Applied Sciences 14 (11): 4772. https://doi.org/10.3390/app14114772.
18. Kou, Liang, Donghui Zhao, Hui Han, Xiong Xu, Shuaige Gong, and Liandong Wang. 2023. "SSCL-TransMD: Semi-Supervised Continual Learning Transformer for Malicious Software Detection." Applied Sciences 13 (22): 12255. https://doi.org/10.3390/app132212255.
19. Mahindru, Arvind, and A. L. Sangal. 2021a. "MLDroid—Framework for Android Malware Detection Using Machine Learning Techniques." Neural Computing and Applications 33 (10): 5183–5240. https://doi.org/10.1007/s00521-020-05309-4.
20. Mahindru, Arvind, and A.L. Sangal. 2021b. "FSDroid:- A Feature Selection Technique to Detect Malware from Android Using Machine Learning Techniques." Multimedia Tools and Applications 80 (9): 13271–323. https://doi.org/10.1007/s11042-020-10367-w.
21. "MalDroid 2020 | Datasets | Research | Canadian Institute for Cybersecurity | UNB." n.d. Accessed October 8, 2024. https://www.unb.ca/cic/datasets/maldroid-2020.html.
22. Nasser, Ahmed R., Ahmed M. Hasan, and Amjad J. Humaidi. 2024. "DL-AMDet: Deep Learning-Based Malware Detector for Android." Intelligent Systems with Applications 21 (March):200318. https://doi.org/10.1016/j.iswa.2023.200318.
23. Nguyen, Giang, Binh Minh Nguyen, Dang Tran, and Ladislav Hluchy. 2018. "A Heuristics Approach to Mine Behavioural Data Logs in Mobile Malware Detection System." Data & Knowledge Engineering 115 (May):129–51. https://doi.org/10.1016/j.datak.2018.03.002.
24. Qin, Chengwei, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. "Is ChatGPT a General-Purpose Natural Language Processing Task Solver?" arXiv. https://doi.org/10.48550/arXiv.2302.06476.
25. Razeghi, Borojerdi, Mahdi Abadi, and Haniye. 2013. "MalHunter: Automatic Generation of Multiple Behavioral Signatures for Polymorphic Malware Detection." In ICCKE 2013, 430–36. https://doi.org/10.1109/ICCKE.2013.6682867.
26. Samaneh Mahdavifar, Andi Fitriah Abdul Kadir, Rasool Fatemi, Dima Alhadidi, Ali A. Ghorbani; Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning, The 18th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC), Aug. 17-24, 2020. and Samaneh Mahdavifar, Dima Alhadidi, and Ali A. Ghorbani (2022). Effective and Efficient Hybrid Android Malware Classification Using Pseudo-Label Stacked Auto-Encoder, Journal of Network and Systems Management 30 (1), 1-34. n.d. "Dataset: CICMalDroid 2020."
27. Zheng, Min, Mingshen Sun, and John C.S. Lui. 2013. "Droid Analytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware." In 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 163–71. https://doi.org/10.1109/TrustCom.2013.25.

28. Zhou, Shuncheng, Honghui Li, Xueliang Fu, and Yuanyuan Jiao. 2024. “A Novel Malware Detection Model in the Software Supply Chain Based on LSTM and SVMs.” Applied Sciences 14 (15): 6678. https://doi.org/10.3390/app14156678.
29. Zhu, Huijuan, Yang Li, Ruidong Li, Jianqiang Li, Zhuhong You, and Houbing Song. 2020. “SEDMDroid: An Enhanced Stacking Ensemble Framework for Android Malware Detection,” 2020.
30. Zolkipli, Mohamad Fadli, and Aman Jantan. 2010. “A Framework for Malware Detection Using Combination Technique and Signature Generation.” In 2010 Second International Conference on Computer Research and Development, 196–99. https://doi.org/10.1109/ICCRD.2010.25.