Malware Detection Among Contact Tracing Apps with Deep Learning

Irene Kilanioti[1(✉)] and George A. Papadopoulos[2]

[1] School of Electrical and Computer Engineering,
National Technical University of Athens,
9 Heroon Polytechneiou Street, Zografou Campus, 157 80 Athens, Greece
eirinikoilanioti@mail.ntua.gr
[2] Department of Computer Science, University of Cyprus, 1 University Avenue,
Aglantzia, 2109 Nicosia, Cyprus
george@ucy.ac.cy

Abstract. Contact tracing has built into a cost-effective social tool, complementary for the prevention and containment of the coronavirus and similar pandemics. Especially these days that meningitis, influenza, but also streptococcus and respiratory syncytial virus (RSV) are on the rise and affect vulnerable populations, e.g. young children, whose immune system is unprepared due to lack of exposure to common viruses during the recent pandemic, contact tracing apps can be used to detect behavioural trends of the users and contribute to outbreak management. This work suggests a comprehensive framework for the identification of malware within contact tracing applications, leveraging deep learning technology, and experimentally corroborates its efficiency. We also introduce a safe and efficient retrieval mechanism for apps associated with the Sustainable Development Goal (SDG) 3 associated with communicable diseases. We aspire to contribute to the safe dissemination of tracing apps in the era of contagious viruses.

Keywords: contact tracing app · security · social · malware · sustainable

1 Introduction

Contact tracing has evolved into a cost-effective tool for the prevention and containment of the coronavirus and similar contagious viruses [1] and their uptake should increase. The prevailing apps are examined thoroughly especially these days that meningitis, influenza, streptococcus and respiratory syncytial virus (RSV) are on the rise and affect vulnerable populations. This work describes up-to-date architectures and protocols of a plethora of current innovative applications for monitoring of contagious diseases, introduced by official governed entities and private companies. These apps include diagnostic apps, contact tracing apps and certificate checkers [2,3] for COVID-19. In this direction, several communication technologies that support these systems, such as Bluetooth, GPS, quick response codes, and Zigbee, prove to be efficient for contact tracing systems [4].

N. T. Nguyen et al. (Eds.): ICCCI 2024, LNAI 14811, pp. 137–150, 2024.
https://doi.org/10.1007/978-3-031-70819-0_11

Contact tracing comprises essentially the identification process of those who may have encountered an infected individual. It is one of the major non-pharmaceutical public health intervention strategies, that include social distancing, isolation, and quarantining in response to emerging outbreaks. Covid tracing apps are mobile software applications used for digital contact tracing. Aside from identifying potential infections, apps serve for the subsequent gathering of further details regarding such contacts aiming to stop virus spread [5]. Technically, there exist variations, i.e., centralized vs decentralized deployments, with their accompanying sensing technologies, i.e., GPS combined with QR code scanners and big data processing as well as wireless Bluetooth devices supported by millimetre-wave and microwave communications [6]. In centralized contact tracing a central server undertakes storage of user information and notifies users of infectious encounters, while in decentralized deployments each smartphone user shares solely the information of infected users with centralized server. The device regularly downloads the contact list from servers to perform contact matching locally and notify users of exposure to the virus.

Malicious tactics involve phishing attempts (app developers and knowledgeable users could decipher the identity of those who have been exposed to the virus), ransomware attacks, suggestions for fraudulent donations, etc. The number of such attempts has risen the last years, mainly due to the increase of people affected globally by the pandemic. Virus scanners are not always effective at detecting new and emerging malware, because they rely on known signatures and patterns of previously identified malware to detect and label new malware [7].

This work describes briefly up-to-date current architectures and protocols of COVID19-tracing apps and emphasizes on a simple and efficient malware detection scheme among them. We introduce a comprehensive framework for the identification of malware within contact tracing applications, leveraging deep learning technology. Furthermore, we incorporate an associated data retrieval model designed to ensure the secure and efficient storage of users' information and aspire to align our research with the United Nations Sustainable Development Goal SDG3 [8]. The deep learning-based scheme uses text classification as a complementary technique to virus scanners, and can be useful in detecting new and emerging malware, as well as other types of malicious content.

Text classification can help to fill this gap by identifying suspicious patterns and anomalies in the text of an application that may indicate the presence of malware [9]. It can be useful in cases where malware has managed to bypass the virus scanner and has been installed on a device. In such cases, text classification can help to identify the malware and mitigate its impact. The apps are characterized by a wide spectrum of functionalities and accompanying text description that spans from offering useful instructions to notifying users if they have been exposed to the virus.

Section 1 introduces the problem and describes our motivation. Section 2 briefly explains the mechanism of contact tracing frameworks, existent deficiencies and our contributions. Section 3 describes the methodology and the experi-

mental setup we used to construct and test a comprehensive framework for the identification of malware within contact tracing applications, and Sect. 5 discusses the results. Furthermore, Sect. 4 incorporates an associated data retrieval model designed to ensure the secure and efficient storage of users' information on decentralized apps and aspires to align our research with the United Nations Sustainable Development Goal SDG3. Finally, Sect. 6 concludes the paper.

2 Related Work

Contact tracing [2,3] has been studied as a fundamental cost-effective complementary approach for virus containment [1]. Centralized apps mostly leverage Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) protocol (e.g., TousAntiCovid, Stop Covid Georgia), BlueTrace/OpenTrace (e..g, COVIDSafe, TraceTogether, careFIJI), and NHS contact tracing protocol (e.g., NHS Covid app, integration of digital contact tracing app with public health programmes and interventions). Decentralized protocols include Google/Apple privacy-preserving tracing (e.g., Stopp Corona, COVID Alert, eRouška, Smittestopp Norway, Smittestop, ASI, Koronavilkku, Corona-Warn-App, Radar COVID), Decentralized Privacy-Preserving Proximity Tracing (DP-3T) (e.g., SwissCovid, HOIA Estonia), TCN Protocol (e.g., NOVID, coEpi, Covid Community Alert), Whisper Tracing Protocol (e.g., Coalition App), Privacy Automated Contact Tracing (East Coast PACT), and Privacy-Sensitive Protocols for Contact Tracing (West Coast PACT) (e.g., CovidSafe). Despite the decrease of the apps' diffusion, policymakers should heed the evidence to leverage this transformative tool and contain outbreaks in the future [10].

Malware detection of generic android apps has been studied with methods based on: system-level data [11,12], API functions calls [13], machine learning of dynamically generated data [14], etc. Malicious tactics among existent contact tracing apps include phishing attempts (app developers and knowledgeable users could decipher the identity of those who have been exposed to the virus, and certain apps collect specific user data, e.g., postcode, for outbreak management reasons), ransomware attacks, suggestions for fraudulent donations, etc. The number of such attempts has risen the last years, due to the increase of people affected globally by the pandemic [15,16].

Virus scanners are not always effective at detecting new and emerging malware, because they rely on known signatures and patterns of previously identified malware to detect and label new malware [7,17,18]. This work introduces a simple and efficient malware detection scheme among contact tracing apps. The scheme uses text classification as a complementary technique to virus scanners, and can be useful in detecting new and emerging malware, as well as other types of malicious content. Text classification can help to fill this gap by identifying suspicious patterns and anomalies in the text of an application that may indicate the presence of malware. It can be useful in cases where malware has managed to bypass the virus scanner and has been installed on a device. In such cases, text classification can help to identify the malware and mitigate its impact.

In our work we aim to detect malware among existent COVID-19 tracing apps and for this purpose we examine a publicly released dataset with multiple COVID-19 themed APK samples, we conduct a grouping of the apps based on how many Anti-Viruses detected that the apps belong to malware and augment the dataset with new features that can then be used in conjunction with existent features to further understand the distribution and characteristics of potentially malicious apps. This research paper introduces a comprehensive framework for the identification of malware within contact tracing applications, leveraging deep learning technology. Furthermore, it incorporates an associated data retrieval model designed to ensure the secure and efficient storage of users' information and aspires to align its research with the United Nations Sustainable Development Goal SDG3.

3 Malware Detection

3.1 Experimental Setup

In this work we examined a publicly released dataset with 4,322 COVID-19 themed Android Package Kit (APK) samples, the format that incorporate all files to install and distribute an application, with 2,500 different apps and 611 apps (370 unique malicious apps) belonging to potential malware considered to be malicious [15]. The majority of malicious apps appeared as benign apps using the same app identifiers (e.g., app name, package name and app icon). The authors studied malware developers' characteristics (habitual developers/ newcomers, location), malware installation methods and malicious behaviours (private information extraction, phishing for the purpose of profit).

We conduct a grouping of the apps based on how many Anti-Viruses detected that the apps belong to malware (Table 1) (https://github.com/GithubIrene00/Transformers_Covid19Apps). VirusTotal malware scanning service aggregating over 60 anti-virus(AV) engines was used to scan the apk files. Given that AV-Rank represents the number of Anti-Viruses that have detected a virus on a particular app, it can serve as an indicator of the app's potential malware status. In order to further analyze this information, we create a new column in the existent dataset that categorizes apps based on their AV-Rank, where a value of "malware" is assigned to apps with an AV-Rank of 1 or greater, and "non-malware" is assigned to apps with an AV-Rank of 0. This new column can then be used in conjunction with other columns such as release date and app name to further understand the distribution and characteristics of potentially malicious apps.

Table 1. Grouping of apps based on how many Anti-Viruses detected these apps as malware

Key	Value
Group A: 0–10	54%
Group B: 11–20	24%
Group C: 20 or greater	22%

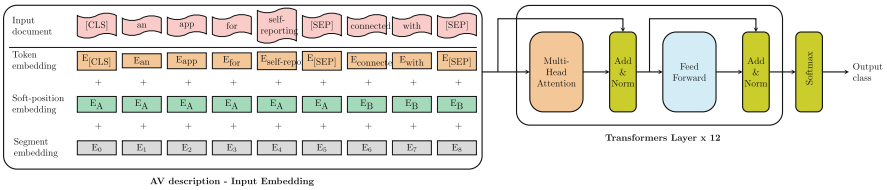

Fig. 1. Transformer architecture for COVID-19 tracing app description, implemented with 12 stacked transformer layers and consisting of token, position and segment embeddings.

The augmented dataset (https://github.com/GithubIrene00/Transformers_Covid19Apps) provides valuable insights into the distribution of COVID-19 related apps that circulate in the internet. We notice that a significant number of apps were released in April and May, and that most of the apps had an AV-Rank of 0, indicating that they were not detected as malware by any anti-virus software. However, a small number of apps had a high AV-Rank, indicating that they were detected as malware by multiple anti-virus software. Additionally, the version and APK size of an app did not have a significant impact on whether or not it was detected as malware. Overall, this dataset highlights the importance of being cautious when downloading apps related to COVID-19 and checking their AV-Rank before installing them.

The augmented dataset contains 3073 features. Each sample comprises crucial details about the app, such as the app name, package name, AV-Rank, version, and APK size. Additionally, every entry has a unique MD5 and SHA256 hash to differentiate between the various apps in the dataset.

3.2 Transformer-Based Model

Transformer models achieve fast training and inference due to their parallel processing capability and they encompass all kinds of noise. The BERT architecture is a bidirectional transformer-based model that has been pre-trained on a massive amount of text data. The BERT model is made up of an encoder stack of transformer blocks. Each transformer block has two sub-layers, a self-attention layer and a feedforward neural network layer. The self-attention layer computes the attention weights for each token in the input sequence, which are then used to compute a weighted sum of the embeddings for all tokens. The feedforward neural network layer applies a linear transformation and a non-linear activation

function to the output of the self-attention layer. Figure 1 depicts the transformer architecture for contact tracing app description with 12 stacked transformer layers and token, position and segment embeddings.

3.3 Data Splitting

To facilitate the process of training and validating the machine learning model, the data was split into three sets. The first split was between X_train, X_test, y_train, and y_test, where the test size was 20%, and the random state was set to 42 for reproducibility. Next, X_train and y_train were further divided into X_train, X_val, y_train, and y_val, with a validation set size of 20%. This split is essential to train the machine learning model on the training set, optimize the model's hyperparameters on the validation set, and eventually test the model's performance on the test set.

3.4 Methodology

We conduct an experiment based on a transformer-based model. Character tokenizer breaks down text into individual characters, rather than words or subwords. This method of tokenization is often used in natural language processing tasks where the focus is on understanding the structure of individual characters in a text rather than the meaning of words or phrases. For this purpose we use BertTokenizer. We use an existent dataset of apps and their package names, along with the label indicating whether they are malware or not. We preprocess the data to clean and format it in a way that can be fed into the transformer. We split the data into training, validation, and test sets to estimate model's generalization performance by evaluating it on unknown test data. Splitting also enables the tuning of model's hyperparameters with the help of validation set without affecting test set.

Afterwards we tokenize the package names in the train, test, and validation sets into individual words or subwords, so that they can be passed as input to the transformer. We create a custom PyTorch dataset called PackageNameDataset for training, validation, and test sets. The dataset takes in two inputs, tokens and labels, and has two methods, len and getitem, that are required by the PyTorch Dataset class. Finally, we create DataLoader for each dataset with batch size of 16 and sets shuffle to false, which will be used to train, validate, and test the model.

As a transformer-based model we used BertForSequenceClassification architecture from transformers package. Some weights of the model checkpoint at bert-base-uncased were not used, as it is expected, and we train this model on a down-stream task to be able to use it for predictions and inference.

Concerning implementation of other models to compare with, there follow the steps:

– Data Preprocessing: The dataset consisted of the "Package Name" column as the input feature (X) and the "malware" column with binary values (0 or 1) as the target variable (y). The "Package Name" column was encoded using

the One Hot Encoder technique to convert categorical data into a numerical format suitable for SVM training.
- Model Training: A Support Vector Machine (SVM) model, with a linear kernel and a regularization parameter (C) of 1, that is effective with limited training samples as well, was selected for training. The SVM model was trained using the encoded features (X) and the target variable (y).
- Results: The recall, which measures the proportion of true positive predictions out of all actual positive instances, was calculated.

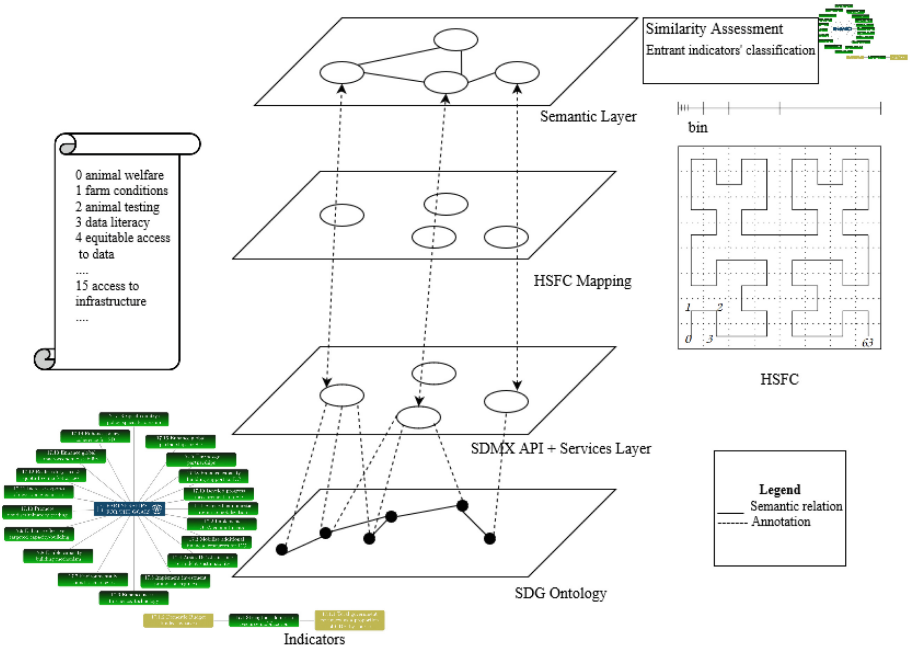

Fig. 2. Suggested knowledge graph-based framework for efficient content similarity search of SDG3 data consisting of i) semantic representation of data, ii) a substrate of the network topology, where the indexing area is divided into semantically homogeneous areas through HSFCs, iii) SDMX-standardized code equivalents of data entries, iv) mapping to SDG ontology, when applicable.

4 A Safe and Efficient Storage Scheme Based on SDG3 Data About Communicable Diseases

The combined framework presented here can be utilised on a long term basis to support the secure and efficient contact tracing in the realm of the medical world.

Sustainable Development Goals (SDGs) [8] were established by the United Nations (UN) in the framework of the UN 2030 Agenda as a measurable international initiative to safeguard the future for the next generations by maintaining social welfare [19,20], [21]. SDG data consists of sustainable development goals, targets, indicators and data series for the quantification of their accomplishment [22] and, especially, SDG3 aims to ensure healthy lives and safeguard well-being. Particularly, target 3.3 refers to containment of communicable diseases.

Hilbert approximations for multidimensional data result in more efficient maintenance of local features as opposed to that achieved by linear ordering [19]. The next order Hilbert Space Filling (HSFC) curve comprises of four gyrated reiterations of the previous order curve. In the next repetition, quadrants are split up into four sub-quadrants each and so on. The line is repetitively folded in such a way that passes by successive neighboring points without intersecting itself and with infinite iterations of the curve construction algorithm it will not omit any point on a continuous plane. HSFCs are always bounded by the unit square, with Euclidean length exponentially growing with τ. Continuity of the curve ensures that affinity of bins on the unit interval signifies affinity in the unit square as well. Two points (x_1, y_1) and (x_2, y_2) with affinity in HSFC of order τ_1 depict affinity in HSFC of order $\tau_2 > \tau_1$ as well.

The first layer of the suggested distributed knowledge graph store (Fig. 2) will entail semantic representation of data. In the next layer of Fig. 2, which acts as a substrate of the network topology, we split up the indexing area in semantically homogeneous areas through dimensionality reducing Hilbert Space Filling Curves (HSFC). Use of curves in this building block proves beneficial for preserving the neighbourhood property of concepts expressed by the indicators of an SDG3 target, as semantically related terms, more probable to respond to a user query, will be placed in the vicinity. In our suggestion linearization is implemented as an overlay upon existing two-dimensional search structures and the distributed file system, that ensures distribution and sharding that scale. Multidimensional queries upon the distributed knowledge graph can be mapped to two-dimensional queries, that range from the minimum to maximum linearization points of the initial query.

Retrieval of SDG3 Data on Communicable Diseases. The algorithm for matching k-semantically closest indicators is based on multi-step filtering and refinement, that consecutively removes irrelevant results and narrows the candidate set (Algorithm 1). In order to optimally calculate distances, we use the algorithm proposed in [23], that performs optimally as far as the number of distance calculations is concerned, and modify it for HSFC representation. We create a ranking by means of the lower bound $l_{\delta H}$, that for all objects o_1, o_2 ensures that $l_{\delta H}(o_1, o_2) \leq \delta_H(o_1, o_2)$ for a distance function δ_H among HSFC projections. Reranking takes place provided that the lower bound does nor surpass the k^{th}-nearest neighbor distance and the results are updated with objects of smaller distances [21].

Algorithm 1. Algorithm for safe filtering of similarity search results among SDG3 data from decentralized contact tracing apps

Input: *app_id*, k, query q, distances $l_{\delta H}$, δ_H
Output: result_set S
Parameters: *indicator*, T=(x,y) $\in$ N, *Hilbert_Space_Filling_Curve* HSFC
$S \leftarrow \emptyset$
$R_H \leftarrow$ ranking(q, l_{δ_H})
$\epsilon \leftarrow$ next value $\in R_H$
while $l_{\delta_H}(q, \epsilon) \leq max_{\alpha \in S} \delta_H(q, \alpha)$ **and** $Transformers(app\_id) > threshold\_value$ **do**
  if $|S| < k$ **then**
    $S \leftarrow S \cup \epsilon$
  else
    if $\delta_H(q, \epsilon) \leq max_{\alpha \in S} \delta_H(q, \alpha)$ **then**
      $S \leftarrow S \cup \epsilon$
      $S \leftarrow S\text{-}argmax_{\alpha \in S} \delta_H(q, \alpha)$
    end if
  end if
  $\epsilon \leftarrow$ next value $\in R_H$
end while
end
return S

The process of refining multi-dimensional data to answer a query of k-closest semantically indicators after projecting on a HSFC is depicted in Fig. 3. After having reduced dimensionality with application of HSFCs, the query for semantically similar indicators for target data of SDG3 can be handled as a nearest neighbor search and implemented with a multi-step filter-and-refine approach [23,24] in an efficient way. The main idea is to filter at a later stage results falsely retrieved at first stage. Creating a lower bound with a simple distance function filters out initially irrelevant results, and in the next step evaluation of results returned at the previous stage takes place with the use of the original distance function. There are multiple properties describing each observation (data entry) and their Statistical Data and Metadata eXchange (SDMX)-standardized code equivalents are also provided. Dimensions (standard demographic info, the whole variety of different age profiles, etc.), time periods and area codes, described through the UNM49 standard are available in the dataset for each indicator from 2000 onwards [21].

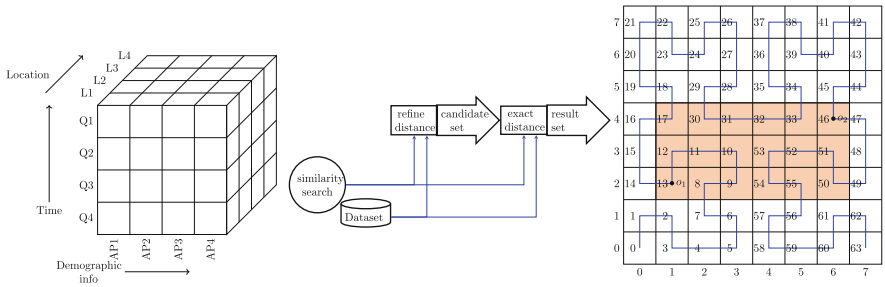

Fig. 3. Process of refining SDG3 multi-dimensional data from decentralized contact tracing apps to answer a query of k-closest semantically indicators after projecting on a HSFC. After having reduced dimensionality (standard demographic info, the whole variety of different age profiles (AP), etc.), time periods and area codes, described through the UNM49 standard) with application of HSFCs, the query for semantically similar indicators can be handled as a nearest neighbor search and implemented with a multi-step filter-and-refine approach for target data of SDG3. Creating a lower bound with a simple distance function filters out initially irrelevant results, and in the next step evaluation of results returned at the previous stage takes place with the use of the original distance function.

5 Results

We notice that transformer-based model performs with higher accuracy and can help to detect and filter out malicious app content that poses a threat to users, thereby improving user safety and security. It's important to evaluate the malware detection model on metrics such as precision, recall, F1-score, and ROC-AUC score, as they provide a more comprehensive understanding of the model's performance. While a high recall is desirable in malware detection, it's also important to balance recall with precision. In malware analysis recall is the most reliable and interpretable metric which measures the percentage of malicious instances found by the model as explained earlier. However, for model selection, ROC-AUC is recommended, as it evaluates malware classifer performance in threshold-free manner.

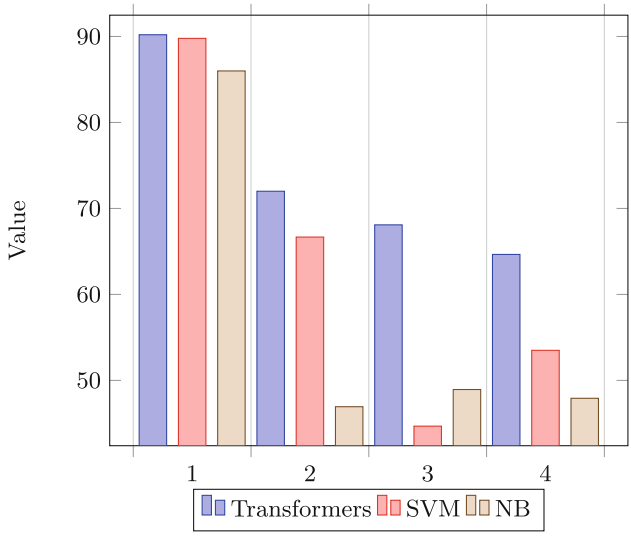

Fig. 4. The model is performing well with an accuracy of 0.9020, precision of 0.72, recall of 0.6809, f1-score of 0.6465, and AUC-ROC of 0.8082. The model depicts a higher recall than SVM, NB.

The model is performing well with an accuracy of 0.9020, precision of 0.72, recall of 0.6809, f1-score of 0.6465, and AUC-ROC of 0.8082 (Fig. 4). Using SVM classification, we have achieved quite lower recall than using BERT. SVM accuracy is 0.8978, precision is 0.6667, recall is 0.4468, f1-score is 0.5350 and AUC-ROC 0.706. NB depicts values: accuracy 85.99, precision 46.94, recall 48.94, f1-score 47.92 and AUC-ROC 70.27. In conclusion, the transformer-based model depicts better balance than SVM and Naive Bayes. A larger dataset would show the superiority of deep learning to an even greater extent. Despite of its small size, the dataset still allows us to exploit additional information such as app category.

Table 2. Comparison with the state-of-the-art malware detection solutions

security framework	precision
Transformer-based model	72%
MobSF (OWASP)	47%
Flowdroid	40%

In general, an accuracy of 0.9020 is quite high and indicates that the model is performing well in terms of correctly classifying the samples. A precision of 0.72 (Table 2) means that out of all the samples predicted as positive, 72%

are actually positive (substantially better than existent security frameworks for apps, e.g., suggested by OWASP MobSF [25] (47%) and FlowDroid [26] (40%)). A recall of 0.6809 indicates that the model is correctly identifying 68% of the positive samples. The f1-score is the harmonic mean of precision and recall and provides an overall measure of the model's performance on both metrics, with a value of 0.6465 indicating a reasonable balance between precision and recall. The AUC-ROC is 0.8082, substantially higher than other models.

The loss calculated during training is a measure of how well the model performs at correctly predicting the labels for the input data. In general, as training progresses, the loss should decrease, indicating that the model is becoming better at making predictions. A lower loss value and higher accuracy value indicate that the model is performing well also on the test set. Test loss: 0.4771, accuracy: 0.9139.

6 Conclusions

Contact tracing has built into a cost-effective social tool, complementary for the prevention and containment of the coronavirus and similar pandemics. Especially these days that meningitis, influenza, streptococcus and respiratory syncytial virus (RSV) are on the rise and affect vulnerable populations, e.g. young children, whose immune system is unprepared due to lack of exposure to common viruses during the recent pandemic, contact tracing apps can be used to detect behavioural trends of the users and contribute to outbreak management.

This work suggests a simple and effective transformer-based malware detection scheme for a plethora of current innovative and future contact tracing apps introduced by official governed entities and private companies, and experimentally corroborates its efficiency. We also propose a scheme for quick retrieval of relevant data associated with SDG3 among existent decentralised and potential future contact tracing apps, as semantic cohesion is preserved.

In conclusion, this paper presents a deep learning-based framework for malware identification within contact tracing apps and proves its efficiency through experiments. Additionally, the study introduces a safe retrieval method for associated apps' data related to SDG3, aiming to ensure the safe distribution of tracing apps amid frequent outbreaks of contagious viruses.

References

1. Hong, X., Han, Y., Wang, B.: Impacts of detection and contact tracing on the epidemic spread in time-varying networks. Appl. Math. Comput. **439**, 127601 (2023). https://www.sciencedirect.com/science/article/pii/S0096300322006749
2. Pozo-Martin, F., Beltran Sanchez, M.A., Müller, S.A., Diaconu, V., Weil, K., El Bcheraoui, C.: Comparative effectiveness of contact tracing interventions in the context of the COVID-19 pandemic: a systematic review. Eur. J. Epidemiol. **38**(3), 243–266 (2023)
3. Juneau, C.-E., Briand, A.-S., Collazzo, P., Siebert, U., Pueyo, T.: Effective contact tracing for COVID-19: a systematic review. Glob. Epidemiol. 100103 (2023)

4. Li, J., Guo, X.: Global deployment mappings and challenges of contact-tracing apps for COVID-19. Available at SSRN 3609516 (2020)
5. Raman, R., Achuthan, K., Vinuesa, R., Nedungadi, P.: COVIDtas COVID-19 tracing app scale-an evaluation framework. Sustainability **13**(5), 2912 (2021)
6. Kallel, A., Rekik, M., Khemakhem, M.: IoT-fog-cloud based architecture for smart systems: prototypes of autism and COVID-19 monitoring systems. Softw. Pract. Exp. **51**(1), 91–116 (2021)
7. Sun, R., Wang, W., Xue, M., Tyson, G., Camtepe, S., Ranasinghe, D.C.: An empirical assessment of global COVID-19 contact tracing applications. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1085–1097 (2021)
8. UN: Sustainable development goals, September 2015. https://www.un.org/sustainabledevelopment/sustainable-development-goals/
9. Gorment, N.Z., Selamat, A., Cheng, L.K., Krejcar, O.: Machine learning algorithm for malware detection: taxonomy, current challenges and future directions. IEEE Access (2023)
10. Salathé, M.: COVID-19 digital contact tracing worked-heed the lessons for future pandemics. Nature **619**(7968), 31–33 (2023)
11. Feng, P., Ma, J., Sun, C., Xu, X., Ma, Y.: A novel dynamic android malware detection system with ensemble learning. IEEE Access **6**, 30:996–31:011 (2018)
12. Razgallah, A., Khoury, R., Hallé, S., Khanmohammadi, K.: A survey of malware detection in android apps: recommendations and perspectives for future research. Comput. Sci. Rev. **39**, 100358 (2021). https://www.sciencedirect.com/science/article/pii/S1574013720304585
13. Khanmohammadi, K., Khoury, R., Hamou-Lhadj, A.: On the use of API calls for detecting repackaged malware apps: Challenges and ideas. In: 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 116–117. IEEE (2019)
14. Wen, L., Yu, H.: An android malware detection system based on machine learning. In: AIP Conference Proceedings, vol. 1864, no. 1. AIP Publishing (2017)
15. Wang, L., et al.: Beyond the virus: a first look at coronavirus-themed mobile malware. arXiv preprint arXiv:2005.14619 (2020)
16. Ho, K.K., Chiu, D.K., Sayama, K.L.: When privacy, distrust, and misinformation cause worry about using COVID-19 contact-tracing apps. IEEE Internet Comput. (2023)
17. Wen, H., Zhao, Q., Lin, Z., Xuan, D., Shroff, N.: A study of the privacy of COVID-19 contact tracing apps. In: Park, N., Sun, K., Foresti, S., Butler, K., Saxena, N. (eds.) SecureComm 2020. LNICST, vol. 335, pp. 297–317. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63086-7_17
18. Baumgärtner, L., et al.: Mind the gap: security & privacy risks of contact tracing apps. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 458–467. IEEE (2020)
19. Kilanioti, I., Papadopoulos, G.A.: An efficient storage scheme for sustainable development goals data over distributed knowledge graph stores. In: Proceedings of 16th IEEE International Conference on Knowledge Graph (ICKG) '22, Orlando, FL, USA, November 2022. Best paper award
20. Kilanioti, I.: Teaching a serious game for the sustainable development goals in the scratch programming tool. Eur. J. Eng. Technol. Res. Spec. Issue 14th Conf. Inform. Educ. CIE, Nov 2022 **7**(7) (2022)

21. Kilanioti, I., Papadopoulos, G.A.: A knowledge graph-based deep learning framework for efficient content similarity search of sustainable development goals data. Data Intell. 1–19 (2023). https://doi.org/10.1162/dint_a_00206
22. UN: Global SDG indicator framework after 2022 refinement (2022). https://unstats.un.org/sdgs/indicators/indicators-list/
23. Seidl, T., Kriegel, H.-P.: 'Optimal multi-step k-nearest neighbor search. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 154–165 (1998)
24. Yu, C.: High-Dimensional Indexing: Transformational Approaches to High-Dimensional Range and Similarity Searches. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45770-4
25. LaMalva, G., Schmeelk, S.: MobSF: mobile health care android applications through the lens of open source static analysis. In: 2020 IEEE MIT Undergraduate Research Technology Conference (URTC), pp. 1–4 (2020)
26. Arzt, S., et al.: Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. ACM SIGPLAN Not. **49**(6), 259–269 (2014)