A Surrogate-Based Technique for Android Malware Detectors' Explainability

Martina Morcos*, Hussam Al Hamadi*, Ernesto Damiani*, Sivaprasad Nandyala† and Brian McGillion†
* Center for Cyber-Physical Systems, EECS Dept., Khalifa University, Abu Dhabi, UAE
† Secure Systems Research Centre (SSRC), Technology Innovation Institute (TII), Abu Dhabi, UAE
Email: *{martina.morcos, hussam.alhamadi, ernesto.damiani}@ku.ac.ae, †{sivaprasad.nandyala, brian.mcgillion}@tii.ae)

***Abstract*—With the emergence of Android malware and behavioral polymorphism, it has been increasingly popular to use advanced machine learning and deep learning approaches for malware detection. Despite the fact that such classifiers have proven accurate in real-life settings, they remain uninterpretable and difficult for analysts and users to comprehend how they arrive at their classification decisions. Considering that the exfiltration of sensitive information is one of the most significant security threats, we examined both monograms and trigrams of system calls for normal and malicious software and received higher detection accuracy by using the trigram dataset. Based on this, we propose an auxiliary architecture for model explainability of complex data features via enhancement and aggregation of the auxiliary model with the main model based on the degree of disagreement between the two models. In this study, we employ the SHAP (Shapley Additive Explanations) framework to interpret the random forest models in order to identify the features most influential in predicting the model's predictions, along with quantifying their contributions to individual predictions. The obtained results confirm that the models are not biased and the features that influence the classification prediction are intuitive in terms of the exfiltration problem in question. In addition, our proposed methodology increases transparency and interpretability of our exfiltration detection model running in production, increasing the users' trust in the model's predictions.**

***Index Terms*—Android Malware, Exfiltration, SHAP, LIME, Explainable AI, Random Forest, TreeSHAP**

I. Introduction

Android malware constitutes one of the greatest threats to mobile security, with more than 32 million malicious applications detected in 2021 alone [1]. The results of numerous malware detection strategies proposed by companies and academic researchers have shown that machine learning algorithms are the most popular and preferred methods due to their high detection accuracy and robustness against malware variants and obfuscation techniques [2] [3]. Although recent research showed impressive results in detection accuracy, adversarial analysis questioned those findings by demonstrating that only a few changes to a malicious Android application can help it evade detection [4]–[6]. Furthermore, despite the vulnerability to well-crafted evasion attacks, many academics have questioned whether the lack of interpretation in black-box machine learning models is appropriate for computer security and intrusion detection systems, arguing that while this lack of interpretation may have been acceptable in the past, today it is expected to be able to understand why and how AI made a specific decision [7]. To avoid dataset bias, it is imperative to have explainable results in security domains. An inspection of the true positives and negatives of model prediction is crucial since sometimes it is not obvious what the system has learned from all of the data and features provided [8]. Accordingly, through explanations, a predictive model is more likely to be accepted since professionals will be able to identify potential biases in these models and decide whether they are trustworthy.

In order to address this issue partially, some Android malware detection research relies on inherently interpretable linear models, often referred to as white box models, that allow the identification of which features contribute to the final prediction. Using linear Support Vector Machine (SVM) as a classifier, Drebin was the first to utilize top weighted features in Android malware detection to determine the contribution of features and output explanations of the obtained results [9]. Another already interpretable ML model is the decision tree which is utilized by TrafficAV tool. TrafficAV analyzes network traffic generated by mobile applications and builds a C4.5 decision tree model. It then generates explanations of how the model predictions are made by implementing a scoring mechanism [10]. In contrast to linear transparent models, which output feature contributions based on model weights, black-box models are not directly interpretable to humans. Non-linear models such as Random Forest (RF) and Artificial Neural Networks (ANNs) are examples of black-box models. However, despite being hard to interpret by humans, black-box models tend to have higher predictive accuracy than their transparent counterparts. As a result, there have been recent advances in post hoc methods to make black box models more understandable to humans. Most prominent among them are local, model-agnostic methods that explain individual predictions made by a black box classifier, such as Local Interpretable Model-agnostic Explanations (LIME) [11] and Shapley Additive ExPlanations (SHAP) [12]. Although neither LIME nor SHAP assumes any knowledge about the internal workings of the trained model, they use perturbations in the data and observe how these perturbations affect the black-box classifier's output to estimate the contribution of individual features towards a particular prediction. The SHAP method is based on Shapley values [13], a concept in co-

operative game theory in which the sample values from a data sample serve as the players in a coalition. The Shapley value is calculated by averaging the marginal contribution of a feature value across all possible coalitions. Alternatively, LIME generates a new dataset based on perturbed samples and corresponding predictions derived from the black-box model. Then, LIME trains an interpretable linear local surrogate model on this new dataset, which is based on the proximity of the sampled instances to the instance to be analyzed. Recent advances in these Explainable Artificial Intelligence (XAI) techniques have resulted in several studies on the application of these tools. SHAP explanations, for example, authors in [14]–[20] observed to differentiate between normal and adversarial samples in a deep neural network classification system. Additionally, authors in [21]–[26] and [27] used SHAP and LIME explanations to determine how different features, such as network traffic and system calls, contribute to the performance of an different ML and DL algorithms.

In this paper, we present the use of an RF model to identify exfiltration behavior in android phones using system call data, collected from real interactions and not emulators, and demonstrate explanations for the model predictions. The contributions of this paper are the following:

1) We propose an auxiliary XAI model architecture using the model agnostic method SHAP for interpreting the RF predictions.
2) We present several approaches for displaying intuitive and comprehensible explanations on models' disagreement.

The structure of the paper is as follows: In the next section, we present the research methodology used in this paper. The Experiment and Results are discussed in Section III . Finally, in Section IV the conclusion and future direction are discussed.

II. Methodology

There are two main components to the proposed methodology: the first part relates to the standard processes of data collection, preparation, and model training; the second part is about the interpretability of the model predictions using an auxiliary model architecture.

A. Feature Preparation & Model Training

In this study, datasets were collected and prepared in-house using Androzoo [28] for malign applications and Google Play Store for benign applications. Although there are several types of Android malware families, we are concerned with malware that uses the Internet to exfiltrate data. As a result, we have devised a process for identifying malware applications which perform this behavior among the myriad of applications present on Androzoo. Once the exfiltration application has been identified, system call data is collected for it while the behavior is constantly triggered, if known. The benign system call data, on the other hand, were collected with a variety of settings using user interaction and in an idle state. The system calls data collected from clean and exfiltrating applications were prepared using two different methods in order to create two distinct datasets that are then used in the proposed auxiliary model architecture. These datasets are:

1) Monogram dataset - the monogram dataset consists of the frequency of each single system call present in a 20s time window.
2) Trigram dataset - the trigram dataset consists of the frequency of three consecutive system calls present in a 20s time window.

Following the preparation of both datasets, we utilize the RF algorithm, an ensemble learning method, as the machine learning classifier for each dataset to classify application processes as either benign or exfiltrating. Since the RF model is an ensemble classifier consisting of hundreds of trees, it is considered a black-box model since the inner workings of the model are hidden and the process between input and output is opaque.

B. Auxiliary Model for XAI

As RF constructs the decision trees, it calculates the importance scores for each feature. This built-in method is called Gini Importance (GI) or Mean Decrease Impurity (MDI) and it calculates each feature's importance as the sum of the number of splits across all trees, proportional to the number of samples it splits [29]. The scores indicate how relevant each feature is to the building of the model's decision trees. While this approach can tell us what features are used to make key decisions within each decision tree, it still does not explain the reasons the RF model makes individual predictions. This is where explainable AI comes in. In this paper, we utilize SHAP [30], an open-source tool that provides global and local explanations between any black-box model and the input features.

We propose an auxiliary structure as shown in Figure 1 to solve the problem of opaque model predictions. Besides the main model, trained on trigram data, an auxiliary model, trained on simple monograms, is used. Due to the fact that the main model trained on trigram data has more information on the sequence of system calls, it results in a higher level of accuracy. 98.9% detection accuracy on trigram data and 96% detection accuracy using the monogram data. Consequently, when the two models agree on a prediction, the explanations of the simpler monogram model are output to the user as explanations. On model disagreement however, two methods can be used to output simple comprehensible explanations to the user, these include:

1) Displaying the complementary SHAP explanations of the monogram model.
2) Filtering of the 3-gram SHAP explanations with the help of monogram SHAP output.

This auxiliary architecture is beneficial in justifying the model predictions in a simple manner and increasing the users' trust towards the deployed model. When both the main and auxiliary models make the same prediction on a given sample, the SHAP explanations of monogram features are far easier to understand than trigram features. Further, since the number

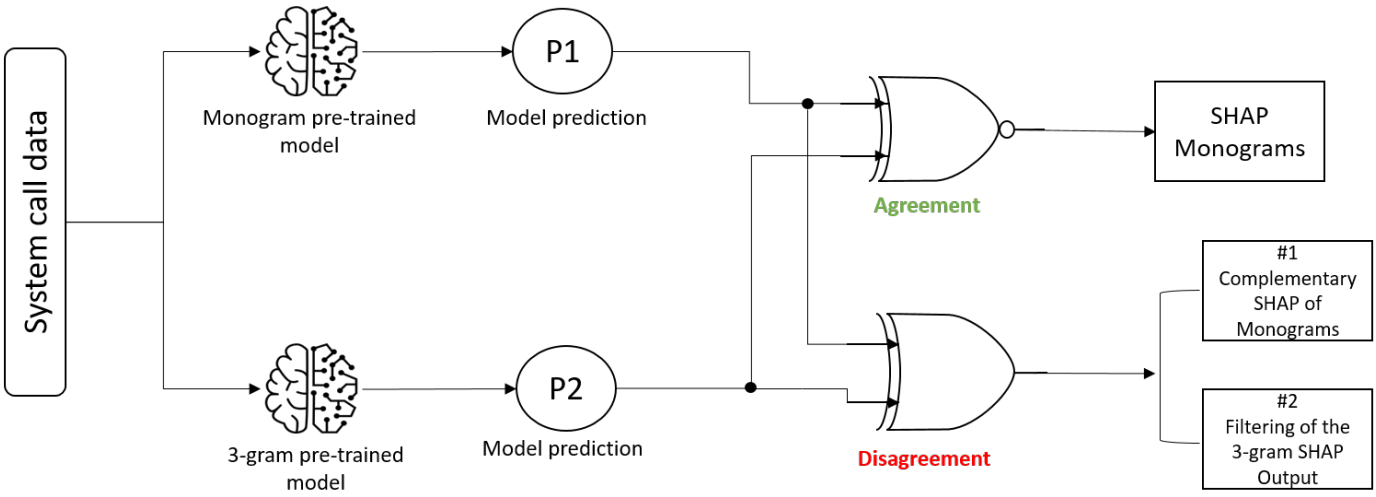


Fig. 1. Auxiliary XAI Model Architecture

of monogram features is much smaller than that of trigram features, their SHAP values can be computed more quickly since the computing time increases exponentially with the number of features.

III. Experiment and Results

In this section, we present the implementation and results of our proposed auxiliary architecture for XAI. This study aims to evaluate whether the features the model uses to make predictions are unbiased, and whether they can be used to provide explanations to a user. Further, we use the auxiliary model to contextualize the results of multiple SHAP explainability tests.

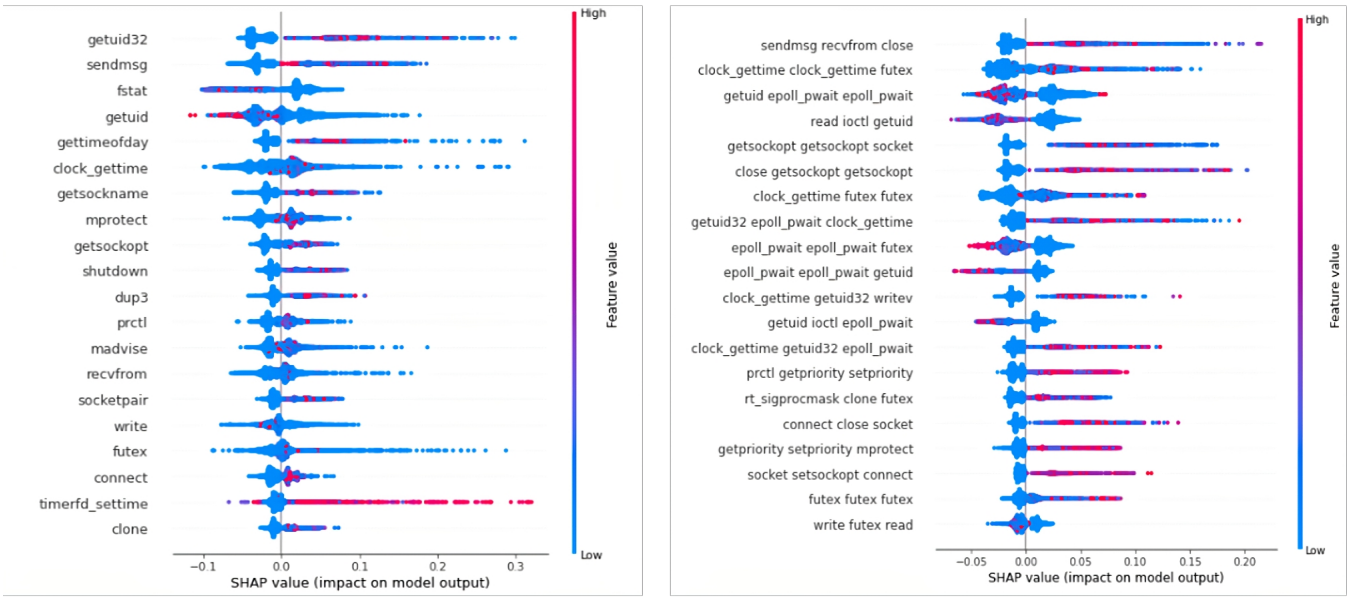
A. Experimental Setup

The following experiment was implemented using Python programming language on a Windows 10 operating system, powered by Intel(R) Core(TM) i9-8950HK CPU@2.90GHz and 32 GB of RAM. We used the SHAP library as well as the SHAP TreeExplainer object [31] which is a variant of SHAP for tree-based ML models like our RF models. The training dataset consists of balanced benign and malign samples collected from 60 benign apps including android system applications and 150 exfiltrating apps. The data was collected from OnePlus 7 mobile phones with Android 10 OS.

B. Global Explanations & Directionality Impact

Global analysis of the models was performed by aggregating the individual Shapley values of each feature in a sample. According to the impact each feature has on each class, Figure 2a and Figure 2b show the decreasing importance of the top 20 features for the monogram model and trigram model, respectively. Additionally, the figures illustrate the directionality impact of the features. On the y-axis, features are arranged in order of importance according to their Shapley values, while on the x-axis, the Shapley values of those features are displayed. Positive Shapley values indicate a positive impact on the prediction, which will lead the model to predict 1 (malign or exfiltrating), and negative Shapley values indicate a negative impact, which will lead the model to predict 0 (benign). The colors indicate whether the feature has a high or low value.

According to Figure 2a, higher values of “getuid32”, ”sendmsg”, ”gettimeofday”, ”getsockname”, ”shutdown”, ”dup3”, ”socketpair”, ”connect” and ”timerfd_settime”, increase the likelihood of class ”Exfiltrating”. On the other hand, higher values of “getuid” and “fstat” decrease the likelihood of predicting the class ”Exfiltrating”. Evidently, these results are intuitive as ”sendmsg”, ”getsockname”, ”shutdown”, ”socketpair” and ”connect” are all network system calls [32] and are used to communicate with the various network interfaces, which is logical since we want to distinguish between exfiltrating and benign behavior. Additionally, two other system calls of interest are ”gettimeofday” and ”timerfd_settime”, which are time-related, and may be invoked in time-based exfiltration attacks. Figure 2b summarizes the directionality impact of the top 20 features used to base the model’s decisions on the trigram global explanations. While interpretation of the trigrams can be more difficult than that of their monogram counterparts, some explanations can still be concluded based on the results. As we look at the system call sequences that increase the prediction of the ”Exfiltrating” class, we can see the sequences contain a lot of network and time related system calls indicating that most malware authors use time-based queries, such as waiting for a certain condition/trigger to become true or waiting for any amount of time before performing network exfiltration. The purpose of this is not only to help understand the single most important system calls that contribute to network exfiltration but also how different system calls interact with one another to accomplish this. As a result, when we compare these insights with information about the overall problem, we can ensure that the model is intuitively correct and makes the right classification decisions.



(a) Monogram Directionality Impact

(b) Trigram Directionality Impact

Fig. 2. SHAP Global Explanations & Directionality Impact

C. Local Explanations

Presented in this section is SHAP's force plot, which displays local explanations and provides an intuitive visualization of the effect of each feature on each sample's predicted result. Further, it demonstrates how each feature influences the model output from the base value prediction which is the average projected outcome across the whole training set. The output of these local explanations is used to display explainable results for the auxiliary model described in Figure 1. In a study conducted with a test set of 20 unseen benign and exfiltrating applications, the trigram model and the monogram model consistently agreed on 95% of the sample predictions and disagreed on 5% of them.As a result, the auxiliary model will generate simple monogram SHAP explanations 95% of the time. Figure 3 illustrates how output explanations for true positives and true negatives are displayed on the models agreement. Based on the figure, it can be seen that some of the features that resulted in the local sample's benign prediction are also among the top features in the global explanations, such as high values for "getuid" and "fstat" and low values for "getuid32". Furthermore, the fact that a 0 value for "getuid32" affects the prediction demonstrates that not only are system calls important to making decisions, but their absence is equally important as well. Additionally, the combination of network ("sendmsg", "recvmsg"), time-based ("timerfd_settime", "clock_gettime"), and file-related ("ppoll", "getdents64", "close") system calls contributes to the true positive case. We can infer from this that the malware sample we are inspecting is accessing the files on the user's phone and using time-based attacks to exfiltrate them over the network to a remote IP address.

Due to the presence of more information on the sequences of system calls in the trigram dataset, the trigram model is more accurate than the monogram, resulting in 5% disagreement between the models. This disagreement occurs when the monogram classifies a true positive sample as benign while the trigram classifies it as exfiltrating. Since we trust the classification of the trigram model as it contains more contextual information, we consider the sample in question positive/exfiltrating and issue explanations accordingly. The output of the trigram explanations is complex and difficult to interpret for the user, so we propose two methods that can be used to simplify the trigram SHAP explanations and output meaningful results. Figure 4 shows an example explanation of the same sample input from the monogram side as well as the trigram side. It can be seen that the monogram explanations that are used to explain the positive/exfiltrating side, while containing a meaningful system call that indicates suspicious behavior, have a weak impact when compared to the benign side explanations. For the trigram explanations however, it can be seen that a combination of such system calls in sequence has a higher impact, resulting in correct positive classification. The two proposed methods for displaying simple interpretable explanations for Figure 4 are:

1) To display the complementary monogram SHAP explanations. The top features from the positive/exfiltrating side is displayed, for example in Figure 4, "timerfd_settime", "madvise", "recvmsg" and "socket".
2) To display a filtered version of the trigram SHAP explanations. With the assistance of the monogram explanations, we can examine the trigram explanations to

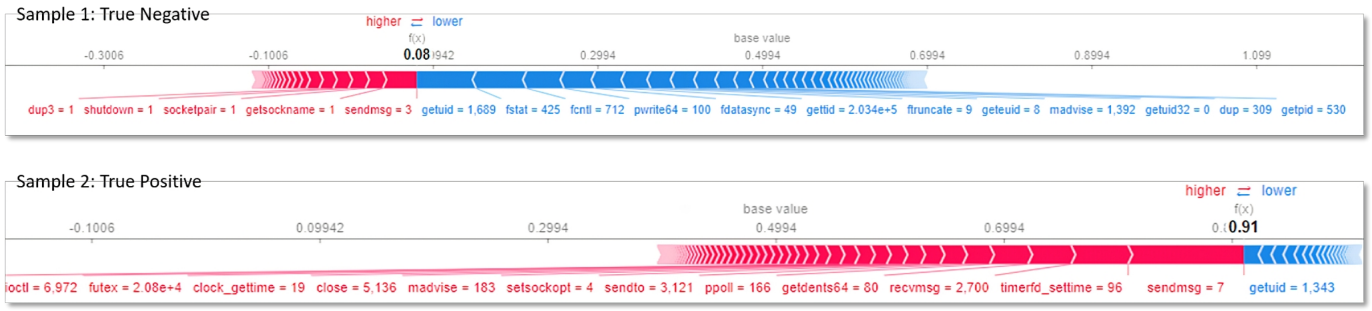


Fig. 3. SHAP Local Explanations on Models Agreement

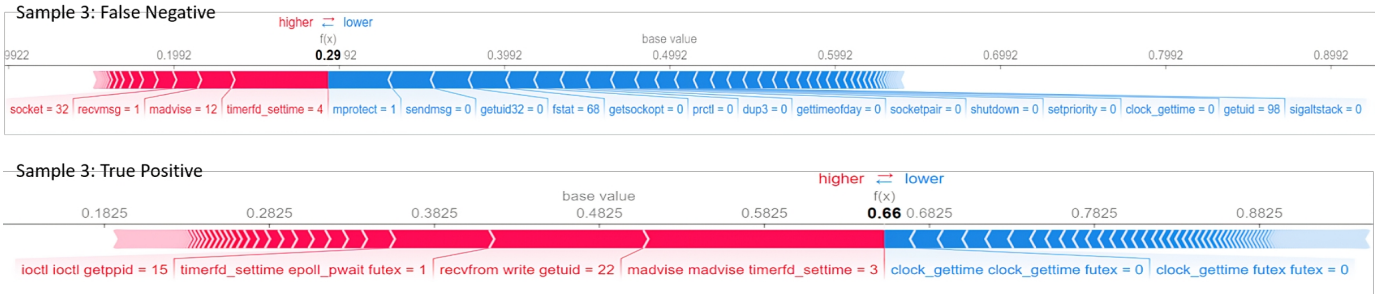


Fig. 4. SHAP Local Explanations on Models Disagreement

identify the single system calls that were not used by the monogram model to determine positive/exfiltrating processes. From Figure 4, the system calls to be displayed are ”recvfrom”, ”write”, ”getuid”, ”epoll_pwait”, ”futex”, ”ioctl” and ”getppid”. These system calls were impactful enough to appear as top features in the trigram model in sequence with other features, but on their own they are weak contributors.

With the absence of system calls associated with completing the action of exfiltrating like sending information over the network or reading from a file descriptor from the example above, it is possible that this combination of system calls in the example sample occurs before the exfiltration process in which the malware author initiates the connection to send users' information to a remote server, making it equally important and crucial to detect. Other examples on which the models can disagree on can include lightweight malware processes in which the malware author exfiltrates small chunks of users' files with long time intervals in between them or by achieving the exfiltration behavior while imitating a benign process' sequence of system calls. With the inspection of both monogram and trigram SHAP explanations, we can identify when the models are outdated due to newer malware and hence resolve to model retraining and update the SHAP explanations.

IV. Conclusion and Future Research Direction

In this paper, we provided a study on the use of Shapley explanations for a Random Forest model on Android exfiltration detection. Using the SHAP library to highlight both global and local features, the output explanations are easy to understand and confirm our analysis of how this type of malware behaves. In our proposed auxiliary model architecture, we display simple explanations to the user regardless of models' agreement, which increases transparency of the model to the user and ensures that meaningful explanations will always be inferred regardless of the complexity of the features. In addition, our proposed methods for displaying explanations on model disagreement are useful for explaining events that would otherwise be overlooked by the auxiliary model, enhancing the robustness of our approach. This work paves the way for a variety of interesting future research directions in XAI. Firstly, it would be interesting to investigate the explanations behind other malware families and compare them with the exfiltration explanations. Second, it would be interesting to build explanations that could withstand attacks that are adversarially robust. Finally, we intend to examine the effectiveness and suitability of different model-agnostic explainability techniques, such as LIME, on our data.

V. Acknowledgments

This work was jointly supported between the Center for Cyber Physical Systems (C2PS) at Khalifa University and the Technology Innovation Institute (TII) under Fund number 8434000379.

References

[1] AV-TEST - The Independent IT-Security Institute, “Av-atlas - malware & pua,” Apr 2022. [Online]. Available: https://portal.av-atlas.org/malware/statistics

[2] S. Chen, M. Xue, Z. Tang, L. Xu, and H. Zhu, “Stormdroid: A streaminglized machine learning-based system for detecting android malware,” in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 377–388.

[3] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Detecting android malware using sequences of system calls," in *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, 2015, pp. 13–20.
[4] H. Dang, Y. Huang, and E.-C. Chang, "Evading classifiers by morphing in the dark," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 119–133.
[5] A. Calleja, A. Martín, H. D. Menéndez, J. Tapiador, and D. Clark, "Picking on the family: Disrupting android malware triage by forcing misclassification," *Expert Systems with Applications*, vol. 95, pp. 113–126, 2018.
[6] A. Ananya, A. Aswathy, T. Amal, P. Swathy, P. Vinod, and S. Mohammad, "Sysdroid: a dynamic ml-based android malware analyzer using system call traces," *Cluster Computing*, vol. 23, no. 4, pp. 2789–2808, 2020.
[7] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 305–316.
[8] N. Fraser, "Neural network follies," Sep 1998. [Online]. Available: https://neil.fraser.name/writing/tank/
[9] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *Ndss*, vol. 14, 2014, pp. 23–26.
[10] S. Wang, Z. Chen, L. Zhang, Q. Yan, B. Yang, L. Peng, and Z. Jia, "Trafficav: An effective and explainable detection of mobile malware behavior using network traffic," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 2016, pp. 1–6.
[11] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.
[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.
[13] L. S. Shapley, "A value for n-person games, contributions to the theory of games, 2, 307–317," 1953.
[14] G. Fidel, R. Bitton, and A. Shabtai, "When explainability meets adversarial learning: Detecting adversarial examples using shap signatures," in *2020 international joint conference on neural networks (IJCNN)*. IEEE, 2020, pp. 1–8.
[15] T. Liu and Li, "Explainable ai for android malware detection: Towards understanding why the models perform so well?" 2022.
[16] M. Ullah and Cheng, "Cyber-threat detection system using a hybrid approach of transfer learning and multi-model image representation," 2022.
[17] Lu and Thing, "How does it detect a malicious app?" explaining the predictions of ai-based android malware detector," 2021.
[18] Alani and Awad, "Paired: An explainable lightweight android malware detection system," 2022.
[19] C. Wu and Lyu, "Why an android app is classified as malware: Toward malware classification interpretation." 2021.
[20] B. Srivastava and Alazab, "Xai for cybersecurity: State of the art, challenges, open issues and future directions," 2022.
[21] B. Hsupeng and Shih-Hao, "Explainable malware detection using predefined network flow," in *2022 24th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2022, pp. 27–33.
[22] T.-H. Kundu and Zhou, "Detection and classification of botnet traffic using deep learning with model explanation." 2022.
[23] Lu and Thing, "Philaex: Explaining the failure and success of ai models in malware detection." 2022.
[24] A. Ullah and Shah, "Explainable malware detection system using transformers-based transfer learning and multi-model visual representation," 2022.
[25] S. Wang and Zhu, "Exposing weaknesses of malware detectors with explainability-guided evasion attacks," 2021.
[26] WANG, "Dissecting malicious behaviours of mobile applications)," 2022. [Online]. Available: https://digital.library.adelaide.edu.au/dspace/handle/2440/135572
[27] R. Alenezi and S. A. Ludwig, "Explainability of cybersecurity threats data using shap," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 01–10.
[28] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2016, pp. 468–471.
[29] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[30] S. Lundberg, "Shap (shapley additive explanations)," 2018. [Online]. Available: https://github.com/slundberg/shap
[31] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*, 2018.
[32] J. Black, "Linux network system calls," 2012. [Online]. Available: http://linasm.sourceforge.net/docs/syscalls/network.php