Android Malware Analysis

Name: - Ashok Reddy Medikonda (C00540105)

Subject: - ENGR480 - Applied Machine Learning - Project2 Report

Abstract-

Leveraging Data Science for Improved Android Malware Detection Android's popularity and open-source nature attract malware developers. Despite existing security mechanisms, Android remains vulnerable. Data science offers promising avenues for enhancing detection capabilities through analytical models. This paper explores static and dynamic analysis techniques for extracting features relevant to malware detection. It further highlights the potential of data science for predicting malicious activities and reviews the current state of the art in Android malware detection frameworks and research. We emphasize the need for combined static and dynamic analysis to capture a broader spectrum of features and propose further research on utilizing virtual cloud devices for effective malware detection.

Keywords- Android, Static and Dynamic Security

1.Introduction

Android is one of the most used mobile operating systems worldwide. Due to its technological impact, its open-source code, and the possibility of installing applications from third parties without any central control, Android has recently become a malware target. Even if it includes security mechanisms, the last news about malicious activities and Android´s vulnerabilities point to the importance of continuing the development of methods and frameworks to improve its security.

To prevent malware attacks, research and developers have proposed different security solutions, applying static analysis, dynamic analysis, and artificial intelligence. Indeed, data science has become a promising area in cybersecurity, since analytical models based on data allow for the discovery of insights that can help to predict malicious activities.

We can analyze cyber threats using two techniques, static analysis, and dynamic analysis, the most important thing is that these are the approaches to get the features that we are going to use in data science.

Static analysis: it includes the methods that allow us to get information about the software that we want to analyze without executing it, one example of them is the study of the code, their callings, resources, etc.

Dynamic analysis: it is another approach where the idea is to analyze the cyber threat during its execution, in other words, get information about its behavior, some of their features are the net flows.

Models: -

Navie Bytes – Naive Bayes is a simple probabilistic classifier based on Bayes' theorem and assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. It's widely used in text classification, spam filtering, and various other applications including Android malware analysis.

Kneighbors: -The k-nearest neighbors (KNN) algorithm is a straightforward, non-parametric, and instance-based learning method used for classification and regression tasks. It works by finding a predefined number (k) of training samples closest in distance to a new input and assigning a label based on the majority vote of its neighbors.

Random Forest: - Random Forest is an ensemble learning method used for both classification and regression tasks. It works by constructing multiple decision trees during the training phase and outputs the class that is the mode of the classes or mean prediction of the individual trees.

Decision Trees: - A decision tree algorithm is a supervised learning method used for both classification and regression tasks. In the context of Android malware analysis, decision trees can be used to classify whether an Android app is malicious or benign based on various features or attributes extracted from the app.

CNN: - A Convolutional Neural Network (CNN) is a type of deep learning algorithm commonly used in image recognition and classification tasks. Its architecture is designed to process structured grid-like data such as images by learning local patterns through convolutional layers, pooling, and fully connected layers.

2. RELATED WORK

Android, as a widely used mobile operating system, has experienced exponential growth, rendering it a prime target for malware attacks owing to its open-source nature and the ability to install third-party applications sans centralized control. The surge in reported malicious activities and identified vulnerabilities in Android has underscored the critical necessity for further advancements in its security protocols.

This paper explores existing security methodologies and frameworks proposed by researchers and developers to combat malware attacks. These solutions primarily rely on static analysis, dynamic analysis, and the integration of artificial intelligence. The rise of data science in cybersecurity presents a promising avenue, leveraging data-driven models to proactively predict and preempt potential security breaches.

Two fundamental techniques, static analysis and dynamic analysis, serve as pivotal approaches in the extraction of features essential for data science-based analysis. Static analysis encompasses methodologies enabling the extraction of information about software without its execution, delving into code studies, resource analysis, and call structures. Conversely, dynamic analysis focuses on real-time execution insights, scrutinizing behavioral patterns and network flows.

The proliferation of Android as a leading mobile operating system has inevitably attracted the attention of malevolent actors, necessitating robust security measures to counter malware infiltrations. The amalgamation of technological prowess, open-source architecture, and unrestricted third-party app installations has made Android a prime target for malicious activities.

In addressing these vulnerabilities, researchers and developers have devised multifaceted security solutions harnessing static analysis, dynamic analysis, and the potential of artificial intelligence, particularly data science. Data-driven models exhibit promise in the realm of cybersecurity, offering predictive insights crucial in preempting and combatting malevolent cyber activities.

Static analysis and dynamic analysis emerge as two pivotal techniques instrumental in extracting crucial features for data-driven analysis. Static analysis methodologies enable information extraction without software execution, encompassing code inspection, resource analysis, and call structure evaluation. In contrast, dynamic analysis centers on real-time execution insights, probing behavioral patterns and network flows.

This paper emphasizes the role of data science-based models in fortifying Android security, spotlighting several key methodologies:

Navie Bytes (Naive Bayes): A probabilistic classifier leveraging Bayes' theorem, apt for text classification and Android malware analysis, assuming feature independence.

K-nearest neighbors (KNN): A non-parametric learning algorithm crucial for classification, deriving input labels through majority voting from neighboring samples.

Random Forest: An ensemble learning technique comprising multiple decision trees, valuable for classification and regression tasks.

Decision Trees: A supervised learning methodology pivotal in classifying Android apps based on extracted attributes.

CNN (Convolutional Neural Network): A deep learning architecture adept in discerning local patterns within structured data, primarily applied in image recognition.

3.PROBLEM DEFINITION

Who is the end user of the system?

The end user of this ML-based product primarily comprises:

Cybersecurity Analysts: Individuals responsible for assessing, analyzing, and mitigating potential threats to Android applications.

Android Developers: Professionals engaged in app development seeking preemptive measures against potential security vulnerabilities.

Security Researchers: Those involved in studying, identifying, and rectifying security flaws within Android ecosystems.

How is he going to use this ML-based product?

The end user will leverage this ML-based product in the following ways:

Malware Detection and Analysis: Utilizing machine learning algorithms to identify and classify potential threats within Android applications.

Preventive Measures: Employing predictive insights derived from the ML models to preemptively fortify Android apps against potential security breaches.

Feature Extraction: Utilizing static and dynamic analysis techniques facilitated by machine learning to extract essential features crucial in identifying malware patterns.

Decision Support: Relying on the ML models to assist in decision-making regarding app validation, categorization as benign or malicious, and security reinforcement strategies.

State the problem as an objective ?

The objective of this ML-based product is to:

Enhance Android Security: Develop a robust framework utilizing machine learning to detect, analyze, and fortify Android applications against potential security threats, especially malware infiltrations.

Leverage Data-Driven Insights: Utilize data science and machine learning to extract features, analyze patterns, and provide predictive insights crucial for preemptive security measures in the Android ecosystem.

Enable Proactive Measures: Empower cybersecurity professionals, developers, and researchers with tools to preemptively detect, categorize, and mitigate potential security vulnerabilities within Android applications.

4. DATASET & FEATURES

In 2016 we explored Android Genome Project (MalGenome), it is a dataset which was active from 2012 until the end of the year 2015, this set of malware has a size of 1260 applications, grouped into a total of 49 families. Today, we can find other jobs such as: Drebin, a research project offering a total of 5560 applications consisting of 179 malware families; AndrooZoo, which includes a collection of 5669661 applications Android from different sources (including Google Play); VirusShare, another repository that provides samples of malware for cybersecurity researchers; and DroidCollector, this is another set which provides around 8000 benign applications and 5560 malware samples, moreover, it facilitates us samples of network traffic as pcap files.

Static Analysis DataSet:-

android.permission.INTERNET: Allows the app to create network sockets and communicate over the internet. Often used for sending or receiving data over the web.

android.permission.READ_PHONE_STATE: Grants access to information about the device's phone number, IMEI, or other telephony-related information.

android.permission.ACCESS_NETWORK_STATE: Enables the app to access information about networks, including whether a network is available or connected.

android.permission.WRITE_EXTERNAL_STORAGE: Permits the app to write to external storage. This permission is commonly used to save files, documents, or data on the device's external storage.

android.permission.ACCESS_WIFI_STATE: Allows access to information about Wi-Fi networks, such as whether Wi-Fi is enabled and the names of connected Wi-Fi devices.

android.permission.READ_SMS: Grants access to read SMS messages stored on the device.

android.permission.WRITE_SMS: Provides the app with permission to write or modify SMS messages on the device.

android.permission.RECEIVE_BOOT_COMPLETED: Allows an application to receive the "boot completed" notification when the system finishes booting.

android.permission.ACCESS_COARSE_LOCATION: Grants access to approximate location derived from network-based methods.

android.permission.CHANGE_WIFI_STATE: Allows the app to change Wi-Fi connectivity state.

Dynamic Analysis Dataset:-

Name: Likely the name of Anti-virus the application being analyzed.

TCP Packets: Count of TCP packets used during network communication.

Dist Port TCP: Destination of TCP ports used.

External IPs: Number of external IPs the application is communicating with.

Volume Bytes: Amount of data transmitted or received in bytes.

UDP Packets: Count of UDP packets used during network communication.

TCP Urg Packet: Count of TCP Urgent packets used.

Source App Packets: Packets sent by the source application.

Remote App Packets: Packets received by the remote application.

Source App Bytes: Bytes sent by the source application.

Remote App Bytes: Bytes received by the remote application.

Duration: Duration of the communication or activity.

Avg Local Packet Rate: Average local packet rate.

Avg Remote Packet Rate: Average remote packet rate.

DNS Query Times: Count of DNS query times.

Type: Likely the type or category of the application (e.g., benign or malicious).

Preprocessing:-

For this approach, we used a set of pcap files from the Droid Collector project integrated by 4705 benign and 7846 malicious applications. All of the files were processed by our feature extractor script (a result from [4]), the idea of this analysis is to answer the next question, according to the static analysis previously seen a lot of applications use a network connection, in other words, they are trying to communicate or transmit information, so.. is it possible to distinguish between malware and benign applications using network traffic. In this case, we have an unbalanced dataset, so another model evaluation will be used. When we processed each pcap we had some problems getting three features (duration, avg remote package rate, avg local package rate) this why got during the feature processing script, we don't have this issue nowadays. We have many outliers in some features, I will omit the depth analysis and only get the set of the data without the noise.

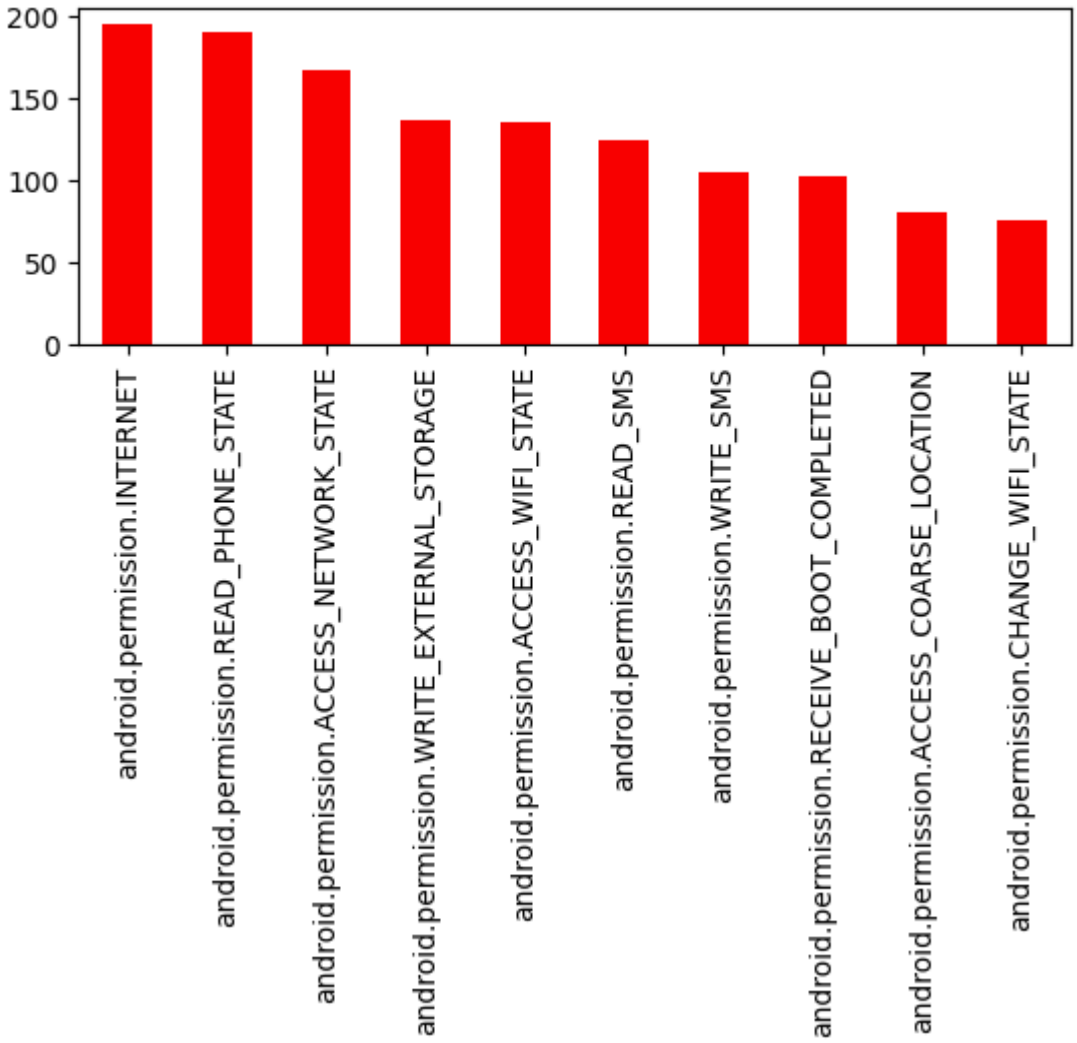
From we concluded that the best network features are:

- TCP packets, it has the number of packets TCP sent and got during communication.
- Different TCP packets, it is the total number of packets different from TCP.
- External IP, represents the number the external addresses (IPs) where the application tried to communicated
- Volume of bytes, it is the number of bytes that was sent from the application to the external sites
- UDP packets, the total number of packets UDP transmitted in a communication.

- Packets of the source application, it is the number of packets that were sent from the application to a remote server.
- Remote application packages, number of packages received from external sources.
- Bytes of the application source, this is the volume (in Bytes) of the communication between the application and server.
- Bytes of the application remote, this is the volume (in Bytes) of the data from the server to the emulator.
- DNS queries, number of DNS queries.

5. MODELS & PERFORMANCE METRICS

Static Analysis Features



Dynamic Analysis

	name	tcp_packets	dist_port_tcp	external_ips	vulume_bytes	udp_packets	tcp_urg_packet	source_app_packets	remote_app_packets	source_app_bytes	remote_app_bytes	duracion	avg_local_pkt_rate	avg_remote_pkt_rate	source_app_packets.1	dns_query_times	type
0	AntiVirus	36	6	3	3911	0	0	39	33	5100	4140	NaN	NaN	NaN	39	3	benign
1	AntiVirus	117	0	9	23514	0	0	128	107	26248	24358	NaN	NaN	NaN	128	11	benign
2	AntiVirus	196	0	6	24151	0	0	205	214	163887	24867	NaN	NaN	NaN	205	9	benign
3	AntiVirus	6	0	1	889	0	0	7	6	819	975	NaN	NaN	NaN	7	1	benign
4	AntiVirus	6	0	1	882	0	0	7	6	819	968	NaN	NaN	NaN	7	1	benign

6. Results & Analysis

now we have seen the two approaches to analyse a cyber threat. Of course, we can use a lot of variables that in this case we didn't use them, for example net flows, methods callings, graph analysis, and many others, but the idea behind this work is to understand that we need to pay attention of all the environments because when we are working in cybersecurity we face with a complex problem.

I'm going to continue with my cybersecurity kernels, in spite of this is the first one, this kernel will be updated once I find an improvement and new results.

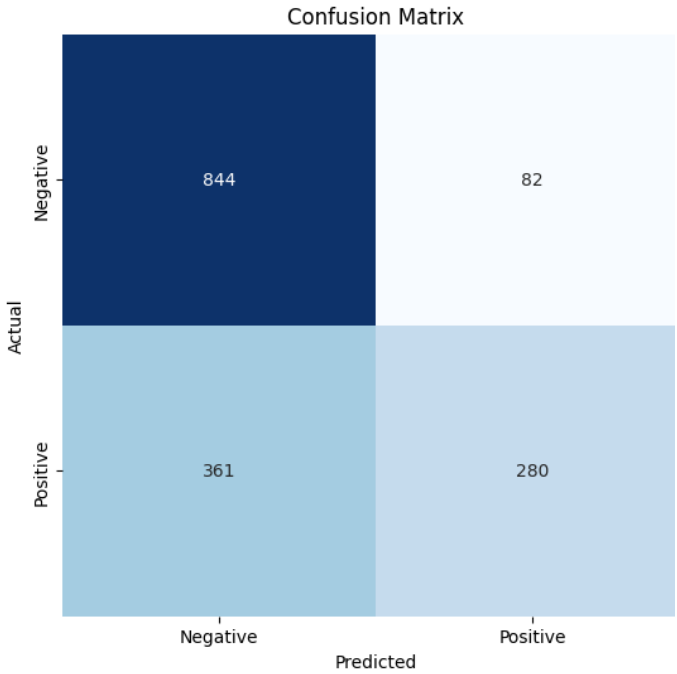
Some of the next kernels that I will publish are:

- Malicious Websites Detection
- Secure Learning
- Cyber Web Attacks Detection
- Deep Fakes Detection

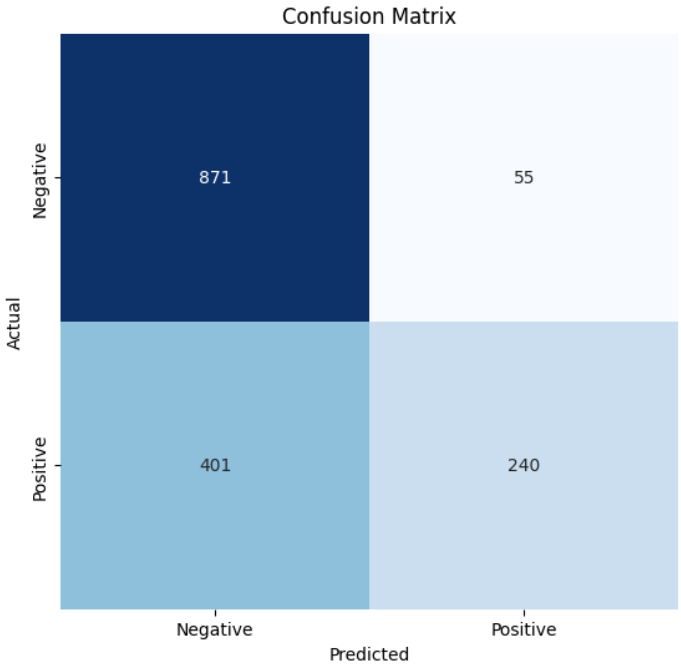
S.No	Static Analysis	Dynamic Analysis
Naive Bytes	83%	44%
Random Forest	91.25%	91.7%
CNN	91.25%	70.90%

Confusion Matrix:-

Static Analysis CNN Model



Dynamic Analysis CNN Model



7. Limitations and Discussion

- Dataset we are using old from 2015.
- Attack vectors become more complex now a days.
- Privacy issue to share public datasets.

REFERENCES

- https://www.kaggle.com/code/xwolf12/android-malware-analysis#Datasets
- [1] López, U., Camilo, C., García Peña, M., Osorio Quintero, J. L., & Navarro Cadavid, A. (2018). Ciberseguridad: un enfoque desde la ciencia de datos-Primera edición.
- [2] Navarro Cadavid, A., Londoño, S., Urcuqui López, C. C., & Gomez, J. (2014, June). Análisis y caracterización de frameworks para detección de aplicaciones maliciosas en Android. In Conference: XIV Jornada Internacional de Seguridad Informática ACIS-2014 (Vol. 14). ResearchGate.
- [3] Urcuqui-López, C., & Cadavid, A. N. (2016). Framework for malware analysis in Android.
- [4] Urcuqui, C., Navarro, A., Osorio, J., & Garcıa, M. (2017). Machine Learning Classifiers to Detect Malicious Websites. CEUR Workshop Proceedings. Vol 1950, 14-17.

- [5] López, C. C. U., Villarreal, J. S. D., Belalcazar, A. F. P., Cadavid, A. N., & Cely, J. G. D. (2018, May). Features to Detect Android Malware. In 2018 IEEE Colombian Conference on Communications and Computing (COLCOM) (pp. 1-6). IEEE.