Network-Based Classification and Analysis of Malware Threats in Android

Sachin Sharma	Mohit	Samyak Jain	Harshit Garg
sachin21559@iiitd.ac.in	mohit21542@iiitd.ac.in	samyak21560@iiitd.ac.in	Harshit20301@iiitd.ac.in
2021559	2021560	2021542	2020301

Guide: Jainendra Shukla
Indraprastha Institute of Information Technology Delhi (IIIT-Delhi), New Delhi

1. MOTIVATION

This project is taken by the urgent necessity to protect Android users from the increasing threat of malicious software. With a global user base of 3.6 billion Android users and approximately 140 billion app downloads yearly, the potential for malware attacks is vast. While numerous current Android security solutions concentrate on strategies centered on the device itself, we propose a network-based approach that could provide a lighter, side-channel-based solution, eliminating the need for the intricate capture of permission and API call data.

2. Introduction

The project aims to develop an Android malware detection system leveraging ML models. Its primary focus is to help enhance the security of Android devices by identifying and classifying malicious applications (Adware, Scareware, SMS Malware). Drawing from past work on Android Fingerprinting and Dynamic Analysis, the goal is to monitor app network activity to discern malware behaviors from benign apps. We also intend to analyze key network features and patterns, enabling future research on compacting the model and on-the-fly Android malware detection.

3. Literature Survey

The paper[1] presented a comprehensive examination of the existing body of research on Android malware detection using network traffic analysis. It meticulously reviews a range of studies spanning the past decade, providing readers with a clear understanding of the various methodologies employed and the key findings of these investigations. This thorough review demonstrates the authors' depth of knowledge regarding the field's historical development. Moreover, the paper adeptly identifies limitations and research gaps in prior work, including the shortcomings of specific detection techniques, such as their reliance on static behaviors or their inability to detect specific malware types. By highlighting these gaps, the authors effectively build a compelling case for the necessity of their research.
However, one notable area for improvement is the currency of the references. The literature review predominantly cites studies up to 2017, potentially missing out on the latest developments in the rapidly evolving landscape of mobile technology and malware threats. Incorporating more recent references would enhance the paper's relevance and ensure that it aligns with the field's current state. Additionally, while the review successfully summarizes prior research, offering a critical analysis of the strengths and weaknesses of each study discussed could provide readers with a deeper understanding of the limitations of previous work and how the proposed research aims to address those limitations. Lastly, improving the organization and clarity of the literature review section, such as categorizing studies into subsections based on common themes or methodologies, would enhance the overall readability of this part of the paper.

This paper[2] explores the intersection of the Internet of Things (IoT) and smartphones, emphasizing the necessity for robust security measures in this ecosystem. It underlines the escalating concern of Android malware and attributes its growth to Android's open-source nature and lax application verification.

Consequently, it underscores the urgency of deploying effective malware detection systems to safeguard user data and IoT services.
The paper primarily delves into trends in mobile malware detection. It segments detection techniques into three main categories: signature-based detection, behavior-based detection, and taint analysis-based detection. Signature-based methods rely on predefined patterns, behavior-based techniques scrutinize application behavior for anomalies, and taint analysis-based methods track data flow within applications. The paper also delves into the role of machine learning classifiers in mobile malware detection, spotlighting the Support Vector Machine (SVM) for its versatility in handling linear and nonlinear data and its high performance.
The paper provides a comprehensive overview of the challenges presented by mobile malware in the context of IoT and smartphones. It serves as a foundation for the proposed malware detection method based on resource monitoring and SVM classification.

4. Dataset Details

The dataset is taken from the CIC repository. which contains 4 labels, namely Android_Adware, Android_Scareware, Android_SMS_Malware and Benign. The dataset includes 355630 entries or instances (rows) with 85 columns.

Some of the important features are as follows:-

Source IP: This is the IP address of the device that initiated the network flow. It represents the origin of the network traffic.

Source Port: It's the port number on the source device that is used for this communication. Ports are used to identify specific services or applications running on the source device.

Destination IP: This is the IP address of the device that received the network traffic. It represents the destination of the network communication.

Destination Port: Similar to the source port, this is the port number on the destination device that is used for this communication. It identifies the specific service or application on the destination.

Protocol: This indicates the network protocol used for this flow. Common values include 6 for TCP (Transmission Control Protocol) and 17 for UDP (User Datagram Protocol). Different protocols have different rules for communication.

Timestamp: The timestamp when this network flow occurred. It provides the date and time when the communication took place, which can be useful for analyzing traffic patterns over time.

Flow Duration: The duration of the network flow, measured in some time unit (e.g., milliseconds, seconds). It tells you how long the communication lasted.

Total Fwd Packets: Total number of packets sent in the forward direction (from source to destination). It provides information about the volume of data transmitted in the forward direction.

Total Length of Fwd Packets: The total length (in bytes) of all packets sent in the forward direction. It gives you the cumulative size of the data transmitted in the forward direction.

Flow Bytes/s: The flow's average data transfer rate in bytes per second. It measures the speed of data transfer in this flow.

Flow Packets/s: The flow's average packet transfer rate in packets per second. It measures how many packets were transmitted per second in this flow.

Flow IAT Mean: The mean (average) inter-arrival time between two consecutive flows. It provides insight into the average time gaps between flows.

Fwd PSH Flags: Flags indicating whether PSH (Push) is set in forward packets. The PSH flag is used to indicate that the data should be pushed to the application layer immediately.

Bwd PSH Flags: Flags indicating whether PSH (Push) is set in backward packets. Similar to the forward PSH flag, this indicates whether data should be pushed to the application layer on the receiving end.

Fwd URG Flags: Flags indicating whether URG (Urgent) is set in forward packets. The URG flag is used to signal that certain data within the packet is urgent.

The dataset was examined and cleaned using the following steps:

Identification of Mixed-Type Features:
Column 56 contained only values of 0, NaN, and unwanted string entries, leading to its removal.Column 58 primarily consisted of numeric values; thus, invalid values of other types were removed to render it usable.Column 63 contained an unnecessary '0' value and was consequently eliminated.

We removed unnecessary columns: Zeroth and the first columns were also unnecessary for analysis, so they were dropped.

We removed duplicates and NANs

As a result, unnecessary columns were removed, and the dataset was cleaned.

5. Feature Selection and Preprocessing

We perform the following tests on the dataset to select the more substantial and relevant features for the classification.

Correlation Test:
We constructed the correlation matrix for all the numerical features. The correlation coefficient (ρ) between two features, X and Y, was calculated using the formula:
$\rho = cov(X, Y) / (\sigma X * \sigma Y)$
Features with correlation coefficients exceeding 90% with at least one other feature were dropped. A total of 29 features were removed, leaving 43 for further analysis.
Mann-Whitney U Test (for Numerical Features):
We used the U test to assess the significance of individual numerical features.
A feature's p-value indicates whether it is significant for the classification. We selected features with p-values < 0.05 and discarded Others.

Chi-squared Test (for Categorical Features):
The Chi-squared test evaluated the significance of categorical features regarding their ability to differentiate between target classes.
Features with $p < 0.05$ were considered informative and retained for further processing.
Information Gain Test:
IG is defined as the difference between the entropy of the dataset before and after partitioning based on a particular attribute.
We used this test to validate all the features selected so far and note the features with the highest and lowest IG.

After the features were selected, following preprocessing steps were done:
Label Encoding:

We used the Label encoding technique to encode the categorical features.
One Hot Encoding was not preferred since the number of values in most categorical features was huge (~6000).

Separating Timestamp variables:
The 'Timestamp' column stores datetime values. Thus, new columns containing month, day, hour, minute and second were formed.

Standardization:

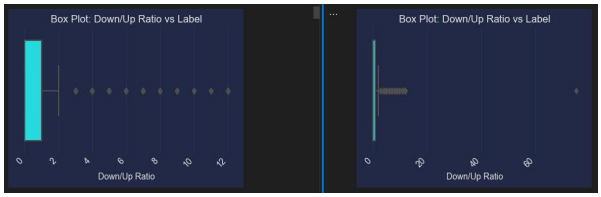
The dataset was standardized during the model Training.

Outlier Detection and removal using LOF:

We detected and removed some of the outliers from

Our data, which increased our best model accuracy by a little bit.

Left side is after outlier removal using LOF, While right side is before outlier removal.



In total, around 56000 data points were removed as outliers from LOF

Principle Component Analysis(PCA):-

We tried this technique both on raw dataset and Refined dataset.

When feature reduction was done on raw dataset, 26 prime

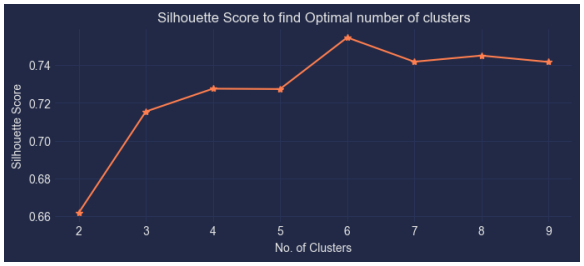
features were found and accuracy improved quite well. However, on

Refined dataset, Accuracy decreased when PCA was applied. Ultimately,

The idea was dropped, and PCA was not applied.

Feature extraction by applying K-means clustering:-

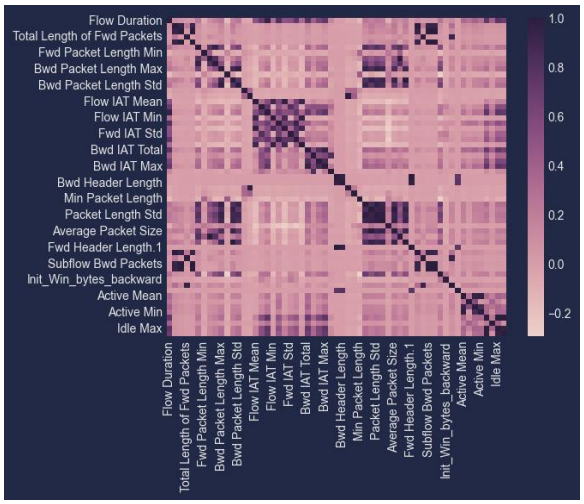
We extracted a feature for our dataset, which is the cluster id of a datapoint, wherein the clusters were formed using K-means clustering. To do this optimally, we performed random undersampling on our dataset for equal representation of all classes. Then, we performed K-means clustering on this dataset, and found the best silhouette score with K=6.



6. Data Analysis

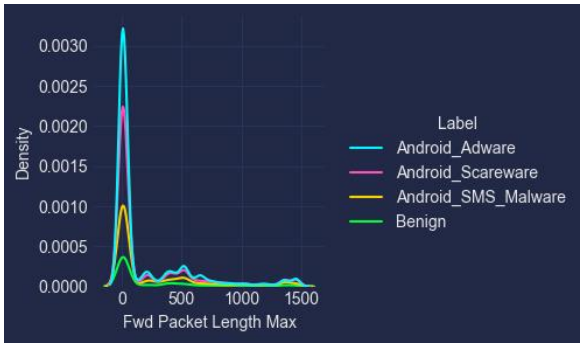
Correlation Heatmap

Many features strongly correlated with each other, leading to their removal during Feature Selection.
Reduced features are suitable for classification due to very little correlation.

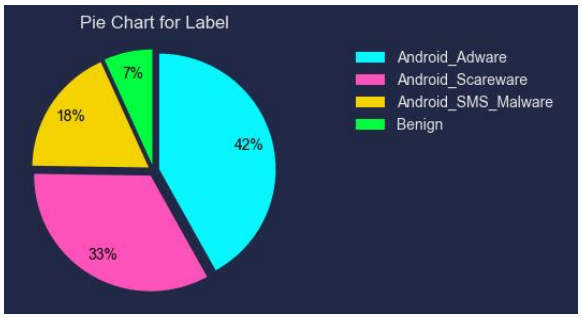


Distribution plots for numerical features

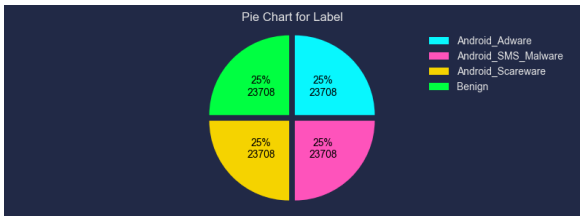
Most features had a minimal range of values occurring most of the time.
Features did not have a Gaussian distribution.



Pie Chart for the target variable before undersampling



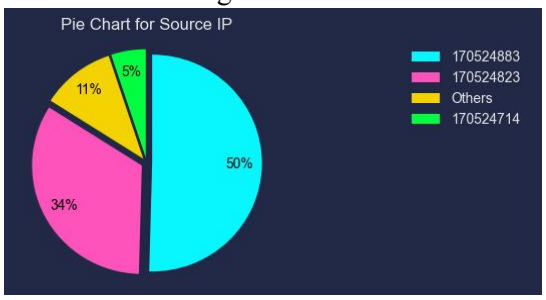
Pie Chart for the target variable after undersampling



Distribution of target variable:

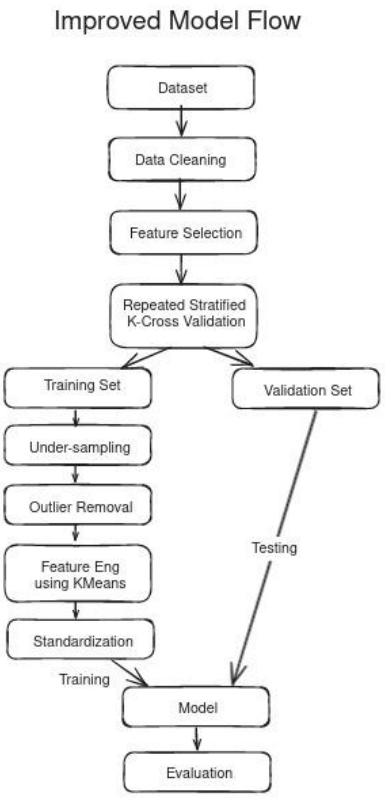
Android_Adware =>147443 (41%)
Android_Scareware => 117082 (33%)
Android_SMS_Malware => 67397 (18%)
Benign => 23708 (12%)

Pie Charts for categorical features



We can see that for the given dataset:-
50% has a common source IP of 170524883
19% has a common destination IP of 170524673
The most common source port is 443
39% of the destination port is 443
The most common protocols the data has is 6.0 (78%) and 17.0 (22%)
ACK flag count and PSH Flag count are binary features

7.Methodology



To establish a suitable baseline model, we selected three straightforward models:

Gaussian Naive Bayes
Logistic Regression
Random Forest Classifier

Subsequently, we executed essential preprocessing steps, including label encoding and standardization.

We employed K-fold cross-validation (with k=3) to train and validate each model and calculate performance metrics for training and validation datasets, including accuracy, precision, recall, and F1 score.

After setting up the baseline, with best validation accuracy as 0.964, as planned, we decided to experiment with following tree models on our dataset:-

1. Gradient Boosting
2. Support Vector Machine(SVM)

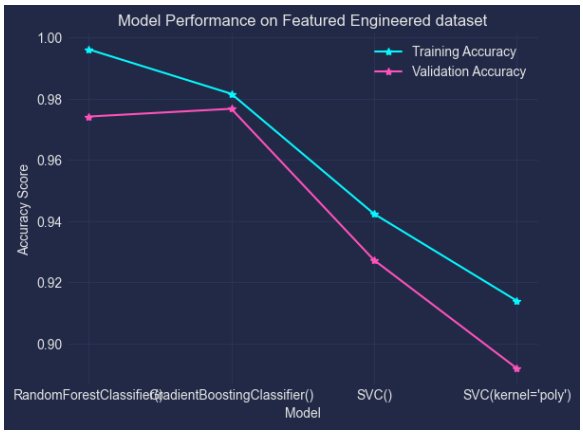
Also, To train our model without any bias towards a specific label, we performed undersampling on our dataset, to produce an undersampled dataset. This will ensure that any kind of skewness is removed from our training dataset.

After this, we executed essential preprocessing steps, including Label encoding and Repeated Stratified K-cross Validation technique, and found performance metrics for training and validation datasets, including accuracy, precision, recall, and F1 score.

W

8. Results

Our experiments showed that GradientBoosting showed the best validation accuracy and better precision, recall and R2_Score on the validation dataset. Another thing that is noticeable with this model is that our training dataset didn't overfit with our new model.



Training Scores

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.996158	0.996161	0.996158	0.996158
GradientBoostingClassifier()	0.981651	0.982370	0.981651	0.981637
SVC()	0.942425	0.942612	0.942425	0.942416
SVC(kernel='poly')	0.914047	0.915252	0.914047	0.914015

Validation Scores

	accuracy	precision	recall	f1_score
RandomForestClassifier()	0.974210	0.975052	0.974210	0.974390
GradientBoostingClassifier()	0.976743	0.978839	0.976743	0.977016
SVC()	0.927319	0.932604	0.927319	0.928488
SVC(kernel='poly')	0.891983	0.907468	0.891983	0.895870

9. Conclusion

In conclusion, this project addresses the critical issue of Android malware detection in light of the ever-expanding user base, surpassing 3.6 billion users and an annual app download count of approximately 140 billion. This escalating threat landscape necessitates innovative security solutions. The project introduces a network-centric approach to complement traditional device-centric security measures, aiming for a more lightweight and effective solution.

The project introduces a baseline model employing Gaussian Naive Bayes, Logistic Regression, and Random Forest Classifier, with the latter demonstrating promising results. As the Tree based model, i.e. Random Forest performed best, we experimented on more Tree-based models like XGBoost and Support Vector Machines

This project lays a solid groundwork for Android malware detection through machine learning, offering a network-based approach to bolster existing security measures. With further refinements and optimizations, this work has the potential to significantly enhance Android device security and protect users from evolving and sophisticated malware threats.

10. Resources

[1] A. H. Lashkari, A. F. A.Kadir, H. Gonzalez, K. F. Mbah and A. A. Ghorbani, "Towards a Network-Based Framework for Android Malware Detection and Characterization," 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 2017, pp. 233-23309, doi: 10.1109/PST.2017.00035.

[2] Ham, Hyo-Sik & Kim, Hwan-Hee & Kim, Myung-Sup & Choi, Mi-Jung. (2014). Linear SVM-Based Android Malware Detection for Reliable IoT Services. Journal of Applied Mathematics. 2014. 1-10. 10.1155/2014/594501.