Spring 2017

Masquerade Detection on Mobile Devices

Swathi Nambiar Kadala Manikoth
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Artificial Intelligence and Robotics Commons](#), and the [Information Security Commons](#)

Recommended Citation

Manikoth, Swathi Nambiar Kadala, "Masquerade Detection on Mobile Devices" (2017). *Master's Projects*. 550.

DOI: <https://doi.org/10.31979/etd.wmtr-yw5y>
https://scholarworks.sjsu.edu/etd_projects/550

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Masquerade Detection on Mobile Devices

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Swathi Nambiar Kadala Manikoth

May 2017

© 2017

Swathi Nambiar Kadala Manikoth

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Masquerade Detection on Mobile Devices

by

Swathi Nambiar Kadala Manikoth

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2017

Dr. Mark Stamp	Department of Computer Science
----------------	--------------------------------

Dr. Robert Chun	Department of Computer Science
-----------------	--------------------------------

Dr. Christopher Pollett	Department of Computer Science
-------------------------	--------------------------------

ABSTRACT

Masquerade Detection on Mobile Devices

by Swathi Nambiar Kadala Manikoth

A masquerade is an attack where the attacker avoids detection by impersonating an authorized user of a system. In this research we consider the problem of masquerade detection on mobile devices. Our goal is to improve on previous work by considering more features and a wide variety of machine learning techniques. Our approach consists of verifying the authenticity of users based on individual features and combinations of features for all users to determine which features contribute the most to masquerade detection. Also, we determine which of the two approaches - the combination of features or using individual features has performed better.

ACKNOWLEDGMENTS

I am very thankful to my advisor Dr. Stamp who has been a guiding light, constantly supporting me throughout the Master's project and helping me focus on the right path to complete the project.

I would also like to thank my committee member Dr. Christopher Pollett and Dr. Robert Chun for their valuable time and suggestions for this project.

Last but not least, I would like to thank all my family and friends for always supporting me and believing in me.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
2	Previous Work	5
3	Overview of the dataset and algorithms	9
3.1	Dataset	9
3.1.1	Data Exploration	9
3.1.2	Data Munging	10
3.2	Algorithms used for Predictive Modeling	12
3.2.1	SVM	12
3.2.2	Fine KNN	13
3.2.3	Decision Tree	14
3.2.4	Extreme Gradient Boosting	15
3.2.5	Random Forest	16
3.2.6	Regularized Greedy Forest	16
4	Approach	17
4.1	A potential application architecture	17
4.2	Training model, Validating and testing the model	19
5	Results	22
5.1	Training user based data	22
5.2	Testing user based data	24
5.3	Training feature based	27

5.4	Testing feature based data	29
6	Conclusion and Future Work	33
	LIST OF REFERENCES	34
	APPENDIX	
	Results	38
A.1	ROC curves for User data	38
A.1.1	User10 Results	38
A.1.2	User11 Results	39
A.1.3	User79 Results	40
A.1.4	User55 Results	41
A.1.5	User48 Results	42
A.1.6	User22 Results	43
A.1.7	User48 Results	44
A.1.8	User92 Results	45
A.1.9	User81 Results	46
A.1.10	User36 Results	47
A.2	ROC curves for User data	48
A.3	Important features	51

LIST OF TABLES

1	Examples of masquerade detection research	6
3	Feature : Date	11
4	Feature : Description	11
5	Values of algorithm parameters while training	23
6	Test Results for user data	27
7	Test Results for feature data	27
8	Test Results for feature data	30

LIST OF FIGURES

1	Rates of smartphone malware over several years	2
2	Date feature statistics	11
3	Description Feature Statistics	12
4	Linear SVM	13
5	Image of a simple decision tree for App usage feature	14
6	A high level view of the types of decision trees available	15
7	Architecture for a practical application.	18
8	Diagram depicting the Pipeline	19
9	Flowchart for the process	21
10	Values of AUC at various stages of training using RF	24
11	Graph depicting importance of features	24
12	AUC for Regularized Greedy Forest with User1 data	25
13	AUC for Random Forest with User1 data	25
14	AUC for K-Nearest Neighbor with User1 data	25
15	AUC for Extreme Gradient Boosting with User1 data	25
16	AUC for trees with User 1 data	28
17	Mean of Balanced Accuracy and AUC values for 94 users	28
18	Graph depicting average Accuracies and AUC's for userbased data over different time frames	29
19	Range of values for the time all users were active on their phones	29
20	Graph depicting Feature Accuracies using Logistic Regression . .	31

21	Variance between values of the Charge feature for all users	31
22	Variance between values of the loc_ids feature for all users	31
23	AUC for Direction feature using SVM	31
24	AUC for Direction feature using LR	31
25	AUC for apps feature using SVM	32
26	AUC for apps feature using LR	32
A.27	AUC for Regularized Greedy Forest with User10 data	38
A.28	AUC for Random Forest with User10 data	38
A.29	AUC for Extreme Gradient Boosting with User10 data	38
A.30	AUC for trees with User 10 data	38
A.31	AUC for K-Nearest Neighbor with User10 data	38
A.32	AUC for Regularized Greedy Forest with User11 data	39
A.33	AUC for Random Forest with User11 data	39
A.34	AUC for Extreme Gradient Boosting with User11 data	39
A.35	AUC for trees with User 11 data	39
A.36	AUC for K-Nearest Neighbor with User11 data	39
A.37	AUC for Regularized Greedy Forest with User79 data	40
A.38	AUC for Random Forest with User79 data	40
A.39	AUC for K-Nearest Neighbor with User79 data	40
A.40	AUC for trees with User 79 data	40
A.41	AUC for Extreme Gradient Boosting with User79 data	40
A.42	AUC for Regularized Greedy Forest with User55 data	41
A.43	AUC for Random Forest with User55 data	41

A.44	AUC for K-Nearest Neighbor with User55 data	41
A.45	AUC for Extreme Gradient Boosting with User55 data	41
A.46	AUC for trees with User55 data	41
A.47	AUC for Regularized Greedy Forest with User48 data	42
A.48	AUC for Random Forest with User48 data	42
A.49	AUC for K-Nearest Neighbor with User48 data	42
A.50	AUC for Extreme Gradient Boosting with User48 data	42
A.51	AUC for trees with User48 data	42
A.52	AUC for RGF with User22 data	43
A.53	AUC for RF with User22 data	43
A.54	AUC for KNN with User22 data	44
A.55	AUC for EGB with User22 data	44
A.56	AUC for Trees with User22 data	44
A.57	AUC for RGF with User92 data	45
A.58	AUC for RF with User92 data	45
A.59	AUC for KNN with User92 data	45
A.60	AUC for EGB with User92 data	45
A.61	AUC for Trees with User92 data	45
A.62	AUC for RGF with User81 data	46
A.63	AUC for RF with User81 data	46
A.64	AUC for EGB with User81 data	46
A.65	AUC for Trees with User81 data	46
A.66	AUC for KNN with User81 data	46

A.67	AUC for RGF with User36 data	47
A.68	AUC for RF with User36 data	47
A.69	AUC for KNN with User36 data	47
A.70	AUC for EGB with User36 data	47
A.71	AUC for Trees with User36 data	47
A.72	AUC for Active feature using LR	48
A.73	AUC for Active feature using SVM	48
A.74	AUC for CellAreaID feature using LR	48
A.75	AUC for CellAreaID feature using SVM	48
A.76	Variance between values of the CellAreaID feature for all users .	48
A.77	Variance between values of the Charge feature for all users	48
A.78	AUC for Charge feature using SVM	49
A.79	AUC for Charge feature using LR	49
A.80	AUC for Description feature using SVM	49
A.81	AUC for Description using LR	49
A.82	AUC for Direction feature using SVM	49
A.83	AUC for Duration feature using LR	49
A.84	AUC for Device_list_macs feature using LR	50
A.85	AUC for loc_ids feature using LR	50
A.86	AUC for loc_ids feature using SVM	50
A.87	Values of AUC at various stages of training using EGB	51
A.88	Graph depicting importance of features	51
A.89	Graph depicting importance of features	52

CHAPTER 1

Introduction

A masquerade is an attack where the attacker avoids detection by impersonating an authorized user of a system and at times tries to gain greater privileges than they are authorized for [1]. A study conducted in the Symposium on Usable Privacy and Security (SOUPS) in 2014 unconcealed that 57% of smartphone owners don't lock their devices when they leave them unattended [2]. This opens opportunities for malicious intruders to gain access to the devices and highlights the requirement for sturdy mechanisms to forestall unauthorized access. An application can also gain access to the resources on the phone that it is not supposed to access. These applications gain access to the phone and simulate the phone usage of a user. So, if there is any Intrusion detection mechanism running on the system, it would assume that the phone is being used by the authorized user. Not certainly enough but the mechanism could, in such a case verify the authenticity of the usage by performing a behavior analysis check. It is certain that malware apps have undesirable behaviors but masquerading can even make supposedly benign apps show unwanted characteristics. A few examples of the kinds of malware which masquerade as an authorized user and that people welcome when they install free applications are given below.

1. A mobile banking app called Zitmo (ZeuS in the mobile) listens to all incoming SMS messages and forwards them to a remote server. What is more alarming is that the one-time passwords that are sent by the bank as a step towards secure 2-step authentication can also be broken by the malware attached to this app [3].
2. There is an app that masqueraded users after tricking them into downloading it thinking it is Google Plus [4].

Figure 1: Rates of smartphone malware over several years

As a result of the rampant increase in the number of smartphone devices [5, 6], intrusion detection on such devices is an important research problem. Figure 1 shows the increasing rates of smartphone malware in the U.S.

A masquerade may be attempted basically through bypassing any kind of authentication mechanism. It could be the use of stolen log on IDs or stolen/hacked passwords and also through discovering security gaps in programs. These techniques could be attempted by an insider in the organization or from an outside user. Masquerading can be a severe threat to the security of computer systems and infrastructures [1]. Constructing practical detection systems masqueraders has been a challenge ever since 1988 [8]. There are a few types of Intrusion Detection Systems (IDS) that have been used. A knowledge based or signature based IDS works by having a database of previous profiles and known system vulnerabilities. It then compares any new signature against the list but this does not work well with attacks that haven't been recorded. Another approach towards solving the problem of masquerade detection is to find deviations from the normal user profiles. Behavior detection has become an accepted means of detecting anomalous behavior [9].

There have been several approaches to improve security and privacy for mobile devices. Firstly, a Personal Identification Number (PIN) or passwords are used [10].

Nevertheless it is upto the user to make sure the passwords or PIN remain a secret. Another approach used is behavior based active authentication. Continuous authentication is a form of behavior based authentication wherein the users keystrokes dynamics, gait recognition or touchscreen usage statistics are used to identify the user [11]. All of these methods are vulnerable to privacy theft. With many malicious applications and ways of masquerading coming up, there's a need for there's a need for better dependable methods of access control that do not trust the users to do what is required to ensure security. Machine learning is indeed an efficient in detecting users based on usage behavior as can be seen in Chapter 2. This could be considered as an active IDS or more commonly known as an Intrusion Prevention System that is configured to automatically detect and block the suspected users. We will later present a practical application architecture in this paper. Accuracy is important in such cases because if the model results in too many false positives then the system might end up denying access to authorized users. Imagine having a biometric based authentication on the phone such as fingerprint verification. If the software on the phone denies your fingerprint one fine day and does not unlock the phone when your really need to do something important, it could get very annoying. Implicit authentication based on behavior conjointly has potential as an auxiliary component component in multi-factor authentication systems, when utilized with passwords or different standard techniques [2].

The remainder of this paper is organized as follows. Chapter 2 discusses examples of previous work done on intrusion detection and masquerade detection, including the techniques used and results obtained in each case. Chapter 3 sheds light on the dataset we used. It details out the various features being used and description of each of them. We have also included details of feature engineering and the algorithms used in or research. Chapter 4 is about the approach taken---in this chapter, we define the

problem and propose a real time detection approach. Chapter 5 illustrates the various stages mentioned in the approach in Chapter 4 in detail along with our experimental results. Results for both user based and feature based data in terms of accuracies and AUC, ROC curves are included in this chapter. Finally, Chapter 6 concludes the paper and suggests future work.

CHAPTER 2

Previous Work

Detecting masqueraders has long been a research topic [9], where a model is built for a user's behavior, and masqueraders are detected based on deviations from normal user profiles. Mobile devices present some additional challenges for masquerade detection. Relevant prior work has focused on the problem of masquerade detection by adopting various approaches. Few papers have used system calls to establish software behavior because system calls capture specific details of the applications interaction with its underlying system [12, 13, 14]. One work [15] was based on behavioral recognition system for portable handsets. The paper starts with extraction of behavior signatures. The authors have created a malicious signature database in view of a thorough audit of malware known to date. Since behavior signatures are lesser and shorter than conventional payload signatures, the database is minimized and can in this way be put on a handset. They executed the monitoring layer on the Symbian emulator for run-time formation of signatures and utilized SVM to prepare a classifier. They claim that the model identifies new malware which share certain behavioral patterns with existing samples in the database. Another paper created a new algorithm that uses pair-wise sequence alignment to indicate similarity between sequences of commands [16]. Another paper considered sequences of user activity and used their own clustering algorithms [17]. A platform called VetDroid was created to vet out undesirable behaviors by analysis of permissions [18]. Some work has been done on using neural networks to profile behaviors [19]. One paper especially concentrates on the temporal pattern extraction from user behavior and demonstrates that instead of considering sequential patterns only, it would be a better idea to consider both temporal and sequential patterns to develop efficient solutions for user recognition [19].

One recent paper [20] classified behaviors based on the top 10 applications used by a legitimate user on the Android device. The influence factor of an app has been calculated depending on how much the Mutual Information (MI) score of the original dataset (MI(D)) varies in absence of the instances related to the app. MI is the amount of information that an event contains about the occurrence of an other event. The users mobile usage records are processed and records which contain the current target app are removed iteratively, MI of the reduced dataset is calculated and compared to its influence factor. Then the MI score is normalized and used for ranking the apps. A few top apps are further processed to remove apps associated with less number of instances in the dataset. 4 classifiers have been used to indicate EER and execution time requirements for the dataset. Chi-square attribute evaluation had been used for feature selection to enhance the results. Table 1 summarizes the previous work done on masquerade detection.

Table 1: Examples of masquerade detection research

Author	Feature	Methods used	Performance
Alzubaidi et al., 2017	Top 10 apps used	IBK ; D.Table ;J48; Forest	9.04% EER
Lamba et al., 2017	Sequences of user activity	Clustering	70.00% Accuracy
Maxion and Tahlia, 2002	Command name and user using UNIX acc. auditing mechanism	Naive Bayes	61.4 Cost
F.Li, N.L.Clarke and M.Papadaki	Mobile device activities	Neural networks	35% EER
F.Li, N.L.Clarke and M.Papadaki	Calling Activity - time and duration	Feed-Forward Multi-Layered Perceptron and RBF Neural Network	1.3% EER
Samfat and Molva, 1997	Itinerary and Calls	Mathematical Formula	82.5% Accuracy
Boukerche and Notare, 2002	Calls	RBF Neural Network	97.5% Accuracy
Stormann, 1997	Calls	Rule Based	99% Accuracy
Alves et al., 2006	Calls	Distance-based and clustering	91% Accuracy
Buschkeset al., 1998	Mobility	Bayes decision rule	87.5% Accuracy
Sun et al., 2004	Mobility	High order Markov	87.5% Accuracy

In Table 1,

1. EER is Equal Error Rate which is the difference between expected output and generated output.

2. Cost as a ranking function is

$$\text{Cost} = \text{Misses} + 6 \times \text{False Alarms} \quad (1)$$

3. The formula to calculate accuracy is

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (2)$$

where TP is the number of True Positives, TN is the True Negatives, FP is the number of False Positives, and FN is the number of False Negatives.

One paper [21] in Table 1 described an approach that can be used to score suspicious execution sequences that are generated during the users interaction with the device. They proposed a likelihood-based statistic in order to score each user sequence based on its suspiciousness. The contextual information of the system had been exploited and a clustering technique was employed to cluster a users sequence to know the chances of their occurrence and the score would indicate level of suspiciousness. More precisely, score is inversely proportional to suspiciousness.

Another paper [22] talks about what makes a successful masquerader and attempts to solve the problem of masquerade detection by classifying sequences of user-command data into a legitimate user and the masquerader. A completely new detection algorithm influenced by Naive Bayes text classification was used on the dataset which was formed by randomly considering 50 user data as being legitimate user data and remaining 20 user data as masquerader data.

One of the papers [23] developed an Intusion Detection System (IDS) for mobile devices. The authors extracted telephony based information from MIT **Reality Mining** dataset [24] and applied Feed-Forward Multi-Layered Perceptron Neural Network and Radial Basis Function Neural Network on the data. The data used

contained 2 sets - time of call, the duration and time of call, the duration and the called number.

Paper [18] also used the MIT Reality Mining dataset [24] and built models for the features individually. The features were user data like profiling calling, device usage and Bluetooth network scanning data extracted from the dataset using neural networks.

CHAPTER 3

Overview of the dataset and algorithms

3.1 Dataset

We used the MIT Reality Mining dataset [24]. It consists of data collected from around 100 mobile phone users (students and faculty of MIT) and has logs collected from their devices over a period of 9 months between 2004 to 2005. The user's phones had a software preinstalled in their devices that would collect usage information and relay the logs to a server every night or store the data on an internal memory card.

3.1.1 Data Exploration

The dataset contains below several features for 94 features. However, we chose the below few features for our experiments. Each user in our dataset has the following fields:

date	Date on which information was collected
event	Unique event ID
contact	The contact ID in phone's address book
description	Short description of the communication medium
direction	Direction of the description data
duration	Duration in seconds
hashNum	The hashed phone number of the other party
chargeTime	Date and time the phone is charging or unplugged
charge	whether the phone was charging (1) or unplugged (0)
activeTime	Date and time the phone has been in use and not
active	Whether the phone has been in use (1) and not in use (0)
logtimes	Times when the logs were being written
onTime	The total amount of time data was recorded (in days)
on	Whether or not the phone is turned on or not
dateForLoc	Date and time at which the person was at a particular location
areaID.cellID,all	Area and cell ID of location of the person
all_locs	The unique set of towers seen by the subject
loc_ids	An indexed version of locs. Towers are replaced by a unique ID
device_macs	The MAC addresses of Bluetooth devices discovered on each scan
	(Converted to ints).
deviceDate	The time/dates of each scan

device_list_macs	A list of all the devices Bluetooth MAC addresses seen by the phone (converted from hex to int)
device_types	The discovered Bluetooth device type (as determined by the standard Bluetooth protocol)
device_list_types1	A list of the device types discovered by the phone
device_list_types2	A list of the device types discovered by the phone
device_list_types3	A list of the device types discovered by the phone
cellname	An array of areaID.cellID and the string the user named the location
logTime	Time at which the cellname was recorded
apps	The time each application was started and the total number of times the app was used
comm_voice_date	The dates of the voice calls
comm_data	The number of data sessions initiated on the phone
comm_data_date	The times when the data sessions were started

3.1.2 Data Munging

We divided the data primarily into two forms to make best use of the data available. One group consists of user based data. The user based data depicts the values of various fields mentioned above for 94 users. This can help the model learn a users complete behavior based on the users phone usage. The other group is feature based data. This group depicts user data per field. But for this group, we have only taken into account those fields that have data associated with sequence of time series.

There was some feature engineering done before using the data as input to the various machine learning algorithms like handling null and out of range data. We filled the missing values with mean values and in case of categorical variables, filled missing values with most frequent (mode) values. All categorical variables were considered as integers. Figure 2 and Figure 3 show the way the values of date and description are distributed, Table 3 and Table 4 have information on the various statistics like

Figure 2: Date feature statistics

the number of unique values, the minimum and maximum values, mean, median and standard deviation of the values and the percentage of missing values that have been replaced.

Example statistics of features:

Table 3: Feature : Date

Type	Continuous
Unique values	8945
Missing values	4.3223%
Min	732,029.60
Mean	732,227.38
Median	732,221.61
Std	40.79
Max	732,408.27

Table 4: Feature : Description

Type	Categorical
Unique values	5
Missing values	2.5625%
The most frequent 3	7,570
The least frequent Data call	2

Another such change made to the dataset is to the fields involved with string

Figure 3: Description Feature Statistics

data. We considered three fields which we thought were primarily important and converted the string data in those fields into numerical data in order to create vectors that can be passed as input to a few of the machine learning algorithms. The three features are:

- Direction
- Description
- Apps

We found a list of 154 unique apps from a list of 261619 apps and substituted them with unique values.

3.2 Algorithms used for Predictive Modeling

3.2.1 SVM

SVM is a supervised learning technique that finds a hyperplane that best separates two classes of points as clusters with maximum margin. Margin here is the minimum distance between the hyperplanes [10]. The support vectors i.e the edge feature of both clusters, compute the hyperplane using linear kernel function and by finding weights (distance between the feature and hyperplane). Figure 4 shows the hyperplane, support vectors and clusters separated using Linear kernel.

Figure 4: Linear SVM

3.2.2 Fine KNN

KNN is an unsupervised learning technique used for clustering in which *Euclidean* distance is found between each point and all other points and clusters are formed such that the distance between two points inside a cluster is less than or equal to k (defined by the user). So if we want to find similar samples, we can use KNN. But since KNN determines neighborhood and is associated with a distance metric as discussed above, all the features involved must be numeric. This is one reason why we converted a few of the string features into numerical data by mapping each string against a unique number and substituting them in the dataset. In our case, it would be interesting to find groups of users who are similar to each other and fall into the same group and group of dissimilar users fall into different groups.

Let us consider in our case the user whom we would like to authenticate as class

Figure 6: A high level view of the types of decision trees available

In our case, we have 2 categories in our response variable which are 0 and 1. Thus, we used a simple decision tree.

3.2.4 Extreme Gradient Boosting

Extreme Gradient Boosting(EGB) is generalized way of implementing ADABOOST and it can be used for basic classification. It implements the gradient boosting decision tree algorithm. Boosting and Random Forests are ways of model averaging. The primary idea of Boosting is to fit many large or small trees to re weighted versions of the training data so it averages the tree by learning from errors in the previous trees. It does the classification based on weighted majority vote. We take a random sample

of training data each time and build the tree, then we average the trees to bring the variance down. It means that each tree will at any terminal node give an estimate of the probability of being in a class. Finally, we average out the probabilities. So, the final classifier is the weighted average of classifiers [25].

3.2.5 Random Forest

Random forests(RF) or Random Decision Forests are an ensemble learning method for classification and regression like tasks. These algorithms typically function by building a plenitude of decision trees during the training phase and then in case of classification, they output mode of classes as compared to the mean prediction of individual trees in case of regression. Random forest is a sort of boosting. If Random forests should get the benefit of variance reduction by averaging then the trees constructed should be completely correlated because correlation reduces the amount by which we can reduce the variance. So de-correlating the trees by introducing randomness while growing the tree is the solution. Each time we make a split, we take a look at all the variables. Out of say P variables, we are limited to n variables to search for the split point. Randomness in splitting de-correlates the trees and results in a better reduction in variance by doing the averaging [25].

3.2.6 Regularized Greedy Forest

The training objective of Regularized Greedy Forest(RGF) is to minimize loss. Mathematically, each node of the forest is associated with a decision rule which serves as a basis function or atom for the additive model. Since exact optimum solution is difficult to find we greedily select the basis function and optimize weights.

At a high level, the algorithmic framework for RGF consists of the following components: Fix the weights, and change the structure of the forest (which changes basis functions) in such a way that we achieve the least cost [26].

CHAPTER 4

Approach

In this section, we consider a framework for masquerade detection. Say we want to do masquerade detection for user A, then the framework should build a model based on user A's phone usage logs and when tested with user B or user C or any other user should return low probability of being user A and when tested with user A's data should return a high probability of being user A.

Problem Statement: To detect legitimacy of the user of a mobile device based on anomalies in usage behavior.

4.1 A potential application architecture

Confidentiality can be achieved using something you have, something you know or something you are [10]. If an illegitimate user can break through this authentication then there should be a system that can detect unauthorized usage based on behavior.

A daemon process or to be specific a service that collects logs could be running in the phone. User can choose a time period over which the check should be performed. The logs are sent to the remote server every night. During the time of check, the server can pass the logs to the model and build user behavior. One advantage of this method could be that the user behavior can change over time. For example the apps used on the phone might not remain the same throughout the year. So, collecting logs and building model for every check would work towards updating the model with new usage characteristics. Thus saving false negatives. Moreover having logs sent to a server that is located on the cloud will decentralize the computation from the phone and keep the whole process independent of problems associated with memory constraints. Once the model is built with data over a substantial period of time say 3 weeks, logs obtained could be used as test data against the built model. We can set a threshold and whenever the result of testing the model i.e probability of user being

Figure 7: Architecture for a practical application

the authorized user is below the threshold value, server could notify the user over email that is protected by code. Also, a history of usage results could be maintained and provided to the user if required. If the user is notified, he or she could change the password or find better ways to protect the device from being misused by an unauthorized user. Also, as a security measure an authentication challenge could be given to the user. Figure 7 shows an overview of the above discussed architecture.

Considering all features to build a model would only mean that the model represents authorized users (not one particular user) but consider for example that one of features was call/sms/call duration/apps used. If the model happens to classify a particular user as being unauthorized because the app usage/ call pattern is different from what the model was built with then it would conclude that user B is unauthorized because he does not use the same apps/make the same calls like any other users whose

Figure 8: Diagram depicting the Pipeline

data was used to build the phone. Since all the data we have is valid user data, we decided to go with the authentication problem. The pipeline mentioned on the cloud in 7 can be found in 8.

4.2 Training model, Validating and testing the model

Usually, in order to perform supervised learning one will need two types of data sets - one for training and one for testing. While performing machine learning we train a model, validate the model and test the model. Figure 9 is a flowchart of the process mentioned below.

1. In one dataset we have the input data with the expected output which is either prepared manually or in an automated way, not necessarily completely automated though. It is necessary to have the expected output for every data row here. We train the model using this dataset.
2. The data we are going to test our model with. This is the data whose output we are interested in. It could be the real world data or testing data.
3. It is always good to verify how well the model has been trained. This can be done using data validation technique. 5-fold cross validation is a technique in

which we partition the data into 5 equal sized parts. Then we train the model using 4 partitions except the first part and test it against the first partition. We repeat this by holding out each of the partitions one by one against the remaining data [10].

4. Once we are done with validation, we look at the models and select the best performing approach. Then we estimate the accuracy of the selected approach using the test data. This is the testing phase. Now, the model is ready to be deployed and used by applying real-world data to the model.

Figure 9: Flowchart for the process

CHAPTER 5

Results

We considered user data for 94 users, performed masquerade detection for 10 random users with 200 values among all features of each user for all the 94 users in the training data and in the test data. We also considered features and trained model for each feature.

We used Regularized Greedy Forest, k-nearest Neighbor, Extreme Gradient Boosting, Extra Trees to build models for the user dataset and used SVM with linear kernel, Logistic Regression and Decision Trees for feature dataset.

The results given are the values of Balanced Accuracy and Area Under Curve. Since our dataset is imbalanced which means that our training and test dataset have more data labeled with 0 than 1, we chose calculating balanced accuracy over accuracy. Balanced accuracy is determined when there is unequal distribution in dependent variable which causes a bias towards the majority class. Most of the machine learning algorithms assume balanced data [27]. The formula used to calculate balanced accuracy is

$$\text{Balanced Accuracy} = \frac{1}{2} \times \left(\frac{\text{True Positive}}{\text{Positive}} + \frac{\text{True Negative}}{\text{Negative}} \right) \quad (3)$$

where True Positive stands for number of samples that have been correctly identified. True negative stands for all the samples that have been correctly rejected [28].

5.1 Training user based data

We allocated 8 CPUs and 15 GB RAM computational power over the cloud. All the models have been trained using 5 fold cross validation. The Table 5 describes information on parameters used for various algorithms during training.

Algorithm	Regularized Greedy Forest
Number of Trees/Iterations	50
Area Under Curve	0.999985

Algorithm	Random Forest
Trees/Iterations	5
criterion	entropy
Area Under Curve	1
Algorithm	k-Nearest Neighbor
neighbors	2
weights	uniform
Area Under Curve: 0.96283	
Algorithm	Logistic Regression
penalty	l2
Area Under Curve	0.813436
Algorithm	Extreme Gradient Boosting
Trees/Iterations	50
max_depth	3
Area Under Curve	0.976349
Algorithm	Decision Trees
Trees/Iterations	5
criterion	gini
Area Under Curve	1

Table 5: Values of algorithm parameters while training

Figure 10 shows the AUC values changing over various stages of training using cross-validation.

Importance of features is obtained from learnt models. Since cross-validation is used, importance is averaged over all models. The displayed importance is normalized in such a way that importances obtained with one algorithm from all features sum up to 100.

Figure 10: Values of AUC at various stages of training using RF

Figure 11: Graph depicting importance of features

5.2 Testing user based data

The user based data has classes 0 and 1 as labels where 0 indicated other user and 1 indicated user to be authorized. Testing results for 10 random users using Regularized Greedy Forest(RGF), Random Forest(RF), K-Nearest Neighbor(KNN), Extreme Gradient Boosting(EGB) and Decision Trees(DT) have been given in Table 6

and the respective ROC curves for user1 are shown in Figures 12, 13, 14, 15, 16. The mean of Balanced Accuracy and AUC values for 94 users is depicted in the form of a graph in 17. All the other ROC curves are included in the appendix section A. We also thought it will be useful to experiment with different time frames. So, we tested data collected over 1 week, 1 month and 3 months against the trained models. The results of these experiments can be seen in 18.

Figure 12: AUC for Regularized Greedy Forest with User1 data

Figure 13: AUC for Random Forest with User1 data

Figure 14: AUC for K-Nearest Neighbor with User1 data

Figure 15: AUC for Extreme Gradient Boosting with User1 data

User	Methods	Balanced Accuracy	AUC
User1	RGF	99.97%	0.91
	RF	79.54%	1
	KNN	79.65%	0.86
	EGB	99.98%	0.98
	DT	100.00%	1
User10	RGF	49.45%	0.50
	RF	99.21%	0.99
	KNN	96.22%	0.75

	EGB	51.90%	0.52
	DT	99.98%	0.99
User11	RGF	49.45%	0.50
	RF	99.91%	0.99
	KNN	63.75%	0.5
	EGB	99.48%	0.52
	DT	49.98%	0.53
User79	RGF	70.16%	0.65
	RF	80.22%	0.71
	KNN	48.25%	0.50
	EGB	96.96%	0.98
	DT	48.45%	0.50
User55	RGF	52.55%	0.73
	RF	99.74%	0.97
	KNN	89.25%	0.75
	EGB	49.96%	0.52
	DT	89.20%	0.63
User48	RGF	94.63%	0.35
	RF	99.45%	0.99
	KNN	48.45%	0.50
	EGB	48.96%	0.52
	DT	48.45%	0.50
User22	RGF	24.91%	0.87
	RF	45.25%	0.46
	KNN	89.35%	0.75
	EGB	48.13%	0.49
	DT	48.45%	0.50
User92	RGF	48.91%	0.50
	RF	99.90%	0.99
	KNN	94.40%	0.72
	EGB	48.91%	0.50
	DT	49.04%	0.56
User81	RGF	60.72%	0.69
	RF	60.65%	0.59
	KNN	49.88%	0.51
	EGB	90.01%	0.74
	DT	48.91%	0.50
User36	RGF	37.60%	0.81
	RF	99.45%	0.99
	KNN	98.82%	0.92
	EGB	48.89%	0.50
	DT	48.91%	0.50

Table 6: Test Results for user data

5.3 Training feature based

Figure 19 refers to one of the features model has been trained on. This figure only depicts the variance of values in the feature dataset. During training, for all the features Decision tree has been used with split criteria as Gini's diversity index and SVM using Linear Kernel. Gini's index is "the probability that two entities taken at random from the dataset of interest (with replacement) represent the same type" [29].

Feature	Methods	Validation Accuracy
Phone Active status	LR	89.13%
	DT	94.90%
	SVM	99%
Apps	LR	74.50%
	DT	97.90%
	SVM	98.90%
cell and area ID	LR	71.30%
	DT	98.90%
	SVM	98.90%
phone charge status	LR	86.20%
	DT	97.20%
	SVM	97.20%
location ID of the person	LR	89.20%
	DT	97.80%
	SVM	97.80%
Description	LR	94.30%
	DT	98.90%
	SVM	98.90%
direction of call	LR	85.40%
	DT	98.90%
	SVM	98.90%
mac of device	LR	74.30%
	DT	98.90%
	SVM	98.90%

Table 7: Test Results for feature data

Figure 16: AUC for trees with User 1 data

Figure 17: Mean of Balanced Accuracy and AUC values for 94 users

Figure 18: Graph depicting average Accuracies and AUC's for userbased data over different time frames

Figure 19: Range of values for the time all users were active on their phones

5.4 Testing feature based data

Feature data includes feature vectors of 94 users. The results of testing 10 random features using Logistic Regression(LR), Decision Trees(DT) and Support Vector Machines(SVM) are given in Table 8. Figures 25, 26 depict ROC curves for Apps feature and Figures A.82, A.83 depict ROC curves for Direction feature. Figure A.77 and Figure 22 shows the plotting of mean of the charge and location id

values for all 94 users and standard deviation for each user respectively. A graph depicting Feature Accuracies using Logistic Regression can be seen in 20.

Feature	Methods	Balanced Accuracy	AUC
Phone Active status	LR	48.04%	0.45
	DT	48.45%	0.50
	SVM	48.45%	0.50
cell and area ID	LR	98.13%	0.96
	DT	87.92%	0.80
	SVM	87.92%	0.80
charge	LR	48.45%	0.42
	DT	91.50%	0.80
	SVM	48.50%	0.50
Description	LR	100.00%	0.76
	DT	48.50%	0.50
	SVM	48.45%	0.50
Direction	LR	53.75%	0.91
	DT	48.50%	0.50
	SVM	48.45%	0.50
Duration	LR	50.55%	0.49
	DT	48.45%	0.50
	SVM	48.45%	0.50
Device_list_macs	LR	40.13%	0.30
	DT	48.92%	0.50
	SVM	48.92%	0.50
Apps	LR	97.30%	0.96
	DT	95.91%	0.80
	SVM	95.91%	0.80
loc_ids	LR	30.41%	0.39
	DT	48.45%	0.50
	SVM	48.45%	0.50

Table 8: Test Results for feature data

Figure 20: Graph depicting Feature Accuracies using Logistic Regression

Figure 21: Variance between values of the Charge feature for all users

Figure 22: Variance between values of the loc_ids feature for all users

Figure 23: AUC for Direction feature using SVM

Figure 24: AUC for Direction feature using LR

Figure 25: AUC for apps feature using SVM

Figure 26: AUC for apps feature using LR

CHAPTER 6

Conclusion and Future Work

We considered 31 features for 94 users, trained and tested them using several machine learning algorithms. We performed this experiment for 10 random users. We also trained models for 10 features. We see that the features charge, communication data and location features have been efficient in detecting the authorized user. Using all the 31 features for all the users, Random forests algorithm has performed comparatively well.

- In future, one could try Bayesian classification to build a model. Train and Test data can include various classes and you can predict probabilities of the test data being any of the users. Finally, one could pick the highest probability as the particular user to be authenticated.
- One could include more features and improve feature engineering. There are several features that we have not considered from the MIT dataset which can be included. Most of the features we left out were the ones with less data per user but it would be reasonable to combine the features and work on them.
- You can also apply Spiking neural networks on a few features that have been collected over a time period with time series data example charge, logs, etc. The section for Spiking Neural Networks in [30] specifies that “SNNs have an advantage of being able to process information in the time domain (signals that vary over time)”. You can refer to this tool for doing that [31]. It is a package available in python, it is a simulator for Spiking Neural Networks.
- We concentrated on the authentication problem and focused on one user at a time. A future work would be to perform well in identifying each of the users based on their behavior.

LIST OF REFERENCES

- [1] M. Rouse, “Network security: Masquerade definition,” Jun 2007. [Online]. Available: <http://searchsecurity.techtarget.com/definition/masquerade>
- [2] W. Lance, “Mobile device authentication – a look at behavior-based authentication,” Mar 2017. [Online]. Available: <https://www.finjanmobile.com/mobile-device-authentication-a-look-at-behavior-based-authentication/>
- [3] M. Elinor, “More malware targeting android,” July 2011. [Online]. Available: <https://www.cnet.com/news/more-malware-targeting-android/>
- [4] W. Lance, “Android malware masquerading as google+ app,” Aug 2011. [Online]. Available: <https://www.cnet.com/uk/news/android-malware-masquerading-as-google-app/>
- [5] V. Sam, “Infographic: Android pulls in twice as many users as apple’s ios,” Feb 2014. [Online]. Available: http://www.icrossing.com/uk/ideas/infographic-android-pulls-in-twice-as-many-users-as-apples-ios_11372
- [6] F. Cuppens, L. Wang, N. Cuppens-Boulahia, N. Tawbi, and J. Garcia-Alfaro, *Foundations and Practice of Security: 9th International Symposium, FPS 2016, Québec City, QC, Canada, October 24-25, 2016, Revised Selected Papers*. Springer, 2017, vol. 10128.
- [7] H. Alan, “How secure is android, really?” Oct 2013. [Online]. Available: <http://lifehacker.com/how-secure-is-android-really-1446328680>
- [8] R. A. Maxion and T. N. Townsend, “Masquerade detection using truncated command lines,” in *Proceedings International Conference on Dependable Systems and Networks*, 2002, pp. 219–228.
- [9] T. F. Lunt and R. Jagannathan, “A prototype real-time intrusion-detection expert system,” in *Proceedings. 1988 IEEE Symposium on Security and Privacy*, Apr 1988, pp. 59–66.
- [10] M. Stamp, *Machine Learning with Applications in Information Security*. unpublished manuscript, 2015.
- [11] I. Deutschmann, P. Nordström, and L. Nilsson, “Continuous authentication using behavioral biometrics,” *IT Professional*, vol. 15, no. 4, pp. 12–15, July 2013.

- [12] I.Burguera, U.Zurutuza, and S.Nadjm-Tehrani, “Crowdroid: Behavior-based malware detection system for android,” in *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, ser. SPSM ’11. New York, NY, USA: ACM, 2011, pp. 15--26. [Online]. Available: <http://doi.acm.org/10.1145/2046614.2046619>
- [13] M.Christodorescu, S.Jha, and C.Kruegel, “Mining specifications of malicious behavior,” in *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC-FSE ’07. New York, NY, USA: ACM, 2007, pp. 5--14. [Online]. Available: <http://doi.acm.org/10.1145/1287624.1287628>
- [14] A.Lanzi, D.Balzarotti, C.Kruegel, M.Christodorescu, and E.Kirda, “Accessminer: Using system-centric models for malware protection,” in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS ’10. New York, NY, USA: ACM, 2010, pp. 399--412. [Online]. Available: <http://doi.acm.org/10.1145/1866307.1866353>
- [15] A.Bose, Abhijit, X.Hu, K.Shin, and G.T.Park, “Behavioral detection of malware on mobile handsets,” in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’08. New York, NY, USA: ACM, 2008, pp. 225--238. [Online]. Available: <http://doi.acm.org/10.1145/1378600.1378626>
- [16] P.M.Comparetti, G.Salvaneschi, E.Kirda, C.Kolbitsch, C.Kruegel, and S.Zanero, “Identifying dormant functionality in malware programs,” in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, ser. SP ’10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 61--76. [Online]. Available: <http://dx.doi.org/10.1109/SP.2010.12>
- [17] Y.Zhang, M.Yang, B.Xu, Z.Yang, G.Gu, P.Ning, X.S.Wang, and B.Zang, “Vetting undesirable behaviors in android apps with permission use analysis,” in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS ’13. New York, NY, USA: ACM, 2013, pp. 611--622. [Online]. Available: <http://doi.acm.org/10.1145/2508859.2516689>
- [18] F. Li, N. Clarke, M. Papadaki, and P. Dowland, “Behaviour profiling on mobile devices,” in *2010 International Conference on Emerging Security Technologies*, Sept 2010, pp. 77--82.
- [19] A.Seleznyov and O.Mazhelis, “Learning temporal patterns for anomaly intrusion detection,” in *Proceedings of the 2002 ACM Symposium on Applied Computing*, ser. SAC ’02. New York, NY, USA: ACM, 2002, pp. 209--213. [Online]. Available: <http://doi.acm.org/10.1145/508791.508836>

- [20] A.A.Abdulaziz, R.Swarup, and K.Jugal, “Ranking most informative apps for effective identification of legitimate smartphone owners,” in *Proceedings IEEE Intl Conf on Computer Communications (INFOCOM 2017)*, ser. MobiSec ’17. Atlanta, GA, USA: IEEEExplore, 2017.
- [21] H. Lamba, T. J. Glazier, J. Cámara, B. Schmerl, D. Garlan, and J. Pfeffer, “Model-based cluster analysis for identifying suspicious activity sequences in software,” in *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, ser. IWSPA ’17. New York, NY, USA: ACM, 2017, pp. 17--22. [Online]. Available: <http://doi.acm.org/10.1145/3041008.3041014>
- [22] R. A. Maxion and T. N. Townsend, “Masquerade detection using truncated command lines,” in *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, ser. DSN ’02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 219--228. [Online]. Available: <http://dl.acm.org/citation.cfm?id=647883.738240>
- [23] D. Michalopoulos and N. Clarke, “Intrusion detection system for mobile devices,” *Advances in Networks, Computing and Communications 4*, p. 205.
- [24] E. Nathan, A. Pentland, and L.David, “Inferring friendship network structure by using mobile phone data,” *Proceedings of the national academy of sciences*, vol. 106, no. 36, pp. 15 274--15 278, 2009.
- [25] H.Trevor, T.Robert, and F.Jerome, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Third Edition*. Springer, 2009.
- [26] R. Johnson and T. Zhang, “Learning nonlinear functions using regularized greedy forest,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 942--954, May 2014.
- [27] A. V. C. TEAM, “Practical guide to deal with imbalanced classification problems in r,” Mar 2016. [Online]. Available: <https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/>
- [28] S. Mark, *Introduction to Machine Learning with Applications in Information Security*. CRC Press, 2017.
- [29] W. contributors, “Diversity index,” Apr 2017. [Online]. Available: https://en.wikipedia.org/wiki/Diversity_index#Gini.E2.80.93Simpson_index
- [30] W. contributors, “Types of artificial neural networks,” Apr 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Types_of_artificial_neural_networks&oldid=776306600

- [31] R. Brette, D. Goodman, and M. Stimberg, “The brian spiking neural network simulator,” Nov 2016. [Online]. Available: <http://briansimulator.org/>

APPENDIX

Results

A.1 ROC curves for User data

A.1.1 User10 Results

Figure A.27: AUC for Regularized Greedy Forest with User10 data

Figure A.28: AUC for Random Forest with User10 data

Figure A.29: AUC for Extreme Gradient Boosting with User10 data

Figure A.30: AUC for trees with User10 data

Figure A.31: AUC for K-Nearest Neighbor with User10 data

A.1.2 User11 Results

Figure A.32: AUC for Regularized Greedy Forest with User11 data

Figure A.33: AUC for Random Forest with User11 data

Figure A.34: AUC for Extreme Gradient Boosting with User11 data

Figure A.35: AUC for trees with User11 data

Figure A.36: AUC for K-Nearest Neighbor with User11 data

A.1.3 User79 Results

Figure A.37: AUC for Regularized Greedy Forest with User79 data

Figure A.38: AUC for Random Forest with User79 data

Figure A.39: AUC for K-Nearest Neighbor with User79 data

Figure A.40: AUC for trees with User79 data

Figure A.41: AUC for Extreme Gradient Boosting with User79 data

A.1.4 User55 Results

Figure A.42: AUC for Regularized Greedy Forest with User55 data

Figure A.43: AUC for Random Forest with User55 data

Figure A.44: AUC for K-Nearest Neighbor with User55 data

Figure A.45: AUC for Extreme Gradient Boosting with User55 data

Figure A.46: AUC for trees with User55 data

A.1.5 User48 Results

Figure A.47: AUC for Regularized Greedy Forest with User48 data

Figure A.48: AUC for Random Forest with User48 data

Figure A.49: AUC for K-Nearest Neighbor with User48 data

Figure A.50: AUC for Extreme Gradient Boosting with User48 data

Figure A.51: AUC for trees with User48 data

A.1.6 User22 Results

Figure A.52: AUC for RGF with User22 data

Figure A.53: AUC for RF with User22 data

A.1.7 User48 Results

Figure A.54: AUC for KNN with User22 data

Figure A.55: AUC for EGB with User22 data

Figure A.56: AUC for Trees with User22 data

A.1.8 User92 Results

Figure A.57: AUC for RGF with User92 data

Figure A.58: AUC for RF with User92 data

Figure A.59: AUC for KNN with User92 data

Figure A.60: AUC for EGB with User92 data

Figure A.61: AUC for Trees with User92 data

A.1.9 User81 Results

Figure A.62: AUC for RGF with User81 data

Figure A.63: AUC for RF with User81 data

Figure A.64: AUC for EGB with User81 data

Figure A.65: AUC for Trees with User81 data

Figure A.66: AUC for KNN with User81 data

A.1.10 User36 Results

Figure A.67: AUC for RGF with User36 data

Figure A.68: AUC for RF with User36 data

Figure A.69: AUC for KNN with User36 data

Figure A.70: AUC for EGB with User36 data

Figure A.71: AUC for Trees with User36 data

A.2 ROC curves for User data

Figure A.72: AUC for Active feature using LR

Figure A.73: AUC for Active feature using SVM

Figure A.74: AUC for CellAreaID feature using LR

Figure A.75: AUC for CellAreaID feature using SVM

Figure A.76: Variance between values of the CellAreaID feature for all users

Figure A.77: Variance between values of the Charge feature for all users

Figure A.78: AUC for Charge feature using SVM

Figure A.79: AUC for Charge feature using LR

Figure A.80: AUC for Description feature using SVM

Figure A.81: AUC for Description using LR

Figure A.82: AUC for Direction feature using SVM

Figure A.83: AUC for Duration feature using LR

Figure A.84: AUC for Device_list_macs feature using LR

Figure A.85: AUC for loc_ids feature using LR

Figure A.86: AUC for loc_ids feature using SVM

A.3 Important features

Figure A.87: Values of AUC at various stages of training using EGB

Figure A.88: Graph depicting importance of features

Figure A.89: Graph depicting importance of features