b.tech. project

by Nidheesh Panchal

Submission date: 20-May-2020 10:22PM (UTC+0530)
Submission ID: 1328520540
File name: B.Tech_Final_Year_Project_Report.pdf (816.62K)
Word count: 7079
Character count: 33872

Android Pin Cracking

Project report submitted in partial fulfilment of the requirements for the degree of Bachelor of Technology

by

Nidheesh Rajesh Panchal (2016UCP1008)

Siddhant Gupta (2016UCP1455)

Rahul Jangir (2016UCP1396)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

APRIL 2020

Certificate

We,

NIDHEESH RAJESH PANCHAL (2016UCP1008)

SIDHHANT GUPTA (2016UCP1455)

RAHUL JANGIR (2016UCP1396)

Declare that this thesis titled, "Android Pin Cracking" and the work presented in it are our own. I confirm that:

- This project work was done wholly or mainly while in candidature for a B.Tech. degree in the department of computer science and engineering at Malaviya National Institute of Technology Jaipur (MNIT).

- Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed. Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely our own work.

- We have acknowledged all of the main sources of help.

Signed: _____

Date: _____

Dr Vijay Laxmi

Professor

May 2020

Department of Computer Science and Engineering

Malaviya National Institute of Technology Jaipur

Abstract

The project aims at predicting the PIN codes of an android phone that a user may unlock using a numeric PIN code. Sensors pre-installed on every smartphone work without any acknowledgement to the user and can fetch data continuously without raising suspicion. Using the orientation and movement data from such sensors, a person can detect any touch screen activity by observing a spike in the data from sensors that logs the motion of the phone. Exploiting the loophole that sensors like accelerometer, gyroscope and gravity do not stop recording even when the phone is locked, we aim to crack the PIN of a mobile phone.

The data was collected using an app from multiple environments such as putting a phone on a table, using it in one hand. Different sensors' data used and harvesting machine learning power, we aim to predict what part of the screen is being touched and in turn, predict the pin.

Acknowledgement

We would like to express our sincere gratitude to Dr Vijay Laxmi, our project supervisor, for her patient guidance, enthusiastic encouragement and useful critiques of this work. A constant push and constructive feedbacks made this possible and helped us to have hands-on experience with the new technologies.

Nidheesh Rajesh Panchal	Siddhant Gupta	Rahul Jangir
(2016UCP1008)	(2016UCP1455)	(2016UCP1396)

Contents

Certificate i

Abstract ii

Acknowledgement iii

List of figures vi

List of tables vii

1. Introduction 1

1.2 Important Terms and Concepts 1

1.2.1 Android Sensors 1

1.2.2 Motion Sensor 2

1.2.3 Position Sensor 2

1.2.4 Accelerometer Sensor 2

1.2.5 Gyroscope Sensor 2

1.2.6 Gravity Sensor 2

1.3 Background Process 2

1.4 Multiclass Classification Models 3

1.4.1 Multi-Layer Perceptron model (MLP) 3

1.5 Multioutput Classification 3

2 Literature Survey 4

2.1 0-permission sensors 4

2.2 Using sensors for predicting touch 5

2.3 Ways to exploit leakage 6

2.3.1 Site 6

2.3.2 Android Application 7

2.4 Related Work 8

3 Proposed Method 10

3.1 Data collection 10

3.2 Pre-processing 11

3.2.1 Windowing 12

3.2.2 Three values 12

3.2.3 Three values from uniform data 12

3.2.4 Windowing 2.0 12

3.2.5 Window with uniform time 13
3.2.6 Windowing with uniform time and TS at the end 13
3.2.7 Windowing with uniform time and removing initial values 13
3.2.8 Windowing with uniform time, removing initial values, TS at the end 13
3.3 Classification model 13
3.4 Exploitation App 16
4 Experimental Results 19
4.1 Method 1 19
4.2 Method 2 20
4.3 Pin the PIN 20
4.4 Conclusion 21
References 23

List of figures

Figure 1: Sudden rise is seen in the graph when the screen is touched. 4

Figure 2: Level of self-declared knowledge about different mobile sensors 5

Figure 3: Unique spikes in the graph when different keys are pressed 6

Figure 4: Site opened on a phone 7

Figure 5: “Train” application that takes sensor data in background 7

Figure 6: Random Forest Classifier parameters 14

Figure 7: Decision Tree Classifier parameters 14

Figure 8: Logistic Regression parameters 14

Figure 9: Naive Baye’s parameters 14

Figure 10:K-Neighbours Classifier parameters 14

Figure 11: Support Vector Machine Classifier parameters 14

Figure 12: Logistic Regression parameters 15

Figure 13: Multi-Layer Perceptron parameters 15

Figure 14: Deep learning model parameters 15

Figure 15: No notification for Android Nougat 16

Figure 16: "Service running in background" notification on Android Pie 17

Figure 17: Google Firebase storage where data is uploaded 17

Figure 18: 5 spikes seen in the data for one unlock 18

Figure 19: Split and get timestamp for processing (black dot shows timestamp) 18

List of tables

Table 1: Comparison of classification models in method 1 19

Table 2: Multioutput classification results 19

Table 3: Method 2 test accuracies 20

Table 4: Count of correct digits and accuracy 21

Table 5: Effect of extension of dataset 21

Chapter 1

1. Introduction

In this advancing age of smartphones, it is essential to know how safe your phone is from all the types of cracks, how safe is your lock screen and what background processes are running in your phone.

Looking specifically in the direction of exploiting the motion and position sensors of the android smartphones in the background, we aim to record the sensor data from the phone and predict the numeric PIN used to lock the phone.

It is observed that whenever the screen is touched on specific portions, a spike is created in the motion and position sensors, extracting such information and using it to know which part of the screen was touched, we can crack the pin.

And to do so, we log the data from sensors such as accelerometer, gyroscope and gravity, to know the orientation and movement of the phone in 3D space. Using it as a dataset to feed into a supervised multiclass machine learning (ML) model, that classifies the data into ten classes which represent the keys pressed, or in other words, tell which part of the screen was touched where that particular number key will be present when the user is entering the PIN.

1.2 Important Terms and Concepts

1.2.1 Android Sensors

Almost all Android devices have built-in sensors that can measure motion, orientation and various environmental conditions. They can be used to monitor three-dimensional movement or position of a device.

The Android platform supports three broad categories of sensors:

- Motion
- Environment
- Postion

1.2.2 Motion Sensor

It monitors movements such as tilt, shake, rotation, or swing. The movement can be caused by direct user input or by the physical environment in which the device is sitting. You can monitor motion relative to the device's frame of reference and movement relative to the world's frame of reference. [2]

1.2.3 Position Sensor

The Android platform provides two sensors that let you determine the position of a device: the geomagnetic field sensor and the accelerometer. The orientation sensor was deprecated in Android 2.2 (API level 8), and the orientation sensor type was deprecated in Android 4.4W (API level 20). [3]

That is, instead of using raw data from a dedicated orientation sensor, we use a data combination from the accelerometer and geomagnetic sensor.

1.2.4 Accelerometer Sensor

An accelerometer is a sensor that measures the linear acceleration of a device in its instantaneous rest frame.[5] An accelerometer at rest will measure an acceleration due to Earth's gravity, straight upwards [6] g ≈ 9.81 m/s2. By contrast, accelerometers in free fall will measure zero.

1.2.5 Gyroscope Sensor

Gyroscope fetches the data that tells us about the angular velocity of the device on its axes, i.e. speed of rotation around its axes.

1.2.6 Gravity Sensor

The gravitational force has three vector components, in X, Y & Z directions. A gravity sensor senses these components of a smartphone and tells the alignment or orientation of the phone. It uses gravity acting on it. (Z=9.8 when at rest).

1.3 Background Process

When a SensorEventListener is registered and used on a website or in-app, even after locking the phone while keeping the app/website open in the background, it could still log the sensor data without giving any warning or message to the user of the phone.

1.4 Multiclass Classification Models

It is a classification model that labels the input and categorizes it under one class from the multiple classes that are defined. For example, there can be many weather conditions possible according to the data input, and each sample is labelled under one class.

We will be using the following classifiers in our project:

- Random Forest
- Logistic Regression
- Naïve Bayes Classifier
- KNN Classifier
- Decision Tree
- Support Vector Machines
- Multi-Layer Perceptron (MLP)

1.4.1 Multi-Layer Perceptron model (MLP)

The basic perceptron algorithm says that in the perceptron, we just multiply with weights and add bias and do this in one layer only. In MLP, there can be more than one linear layer (combinations of neurons), i.e. having hidden layers in the network of nodes. Here we will be using MLP from sklearn library and Keras library naming the models as MLP and Deep Learning model respectively. These two different libraries provide a wide range of activation functions, optimizers and the different modifiable parameters, making them both important.

1.5 Multioutput Classification

As the name suggests, it used to get multiple outputs for a single input sample. Each output has its own set of properties, and each property may have multiple classes. It is like a combination of multilabel and multiclass classification. [13]

Chapter 2

2 Literature Survey

2.1 0-permission sensors

Today we have various operating systems on our phones which includes the popular Android and iOS. According to the survey conducted by StatCounter, in March 2020, there are 72.26% Android users, 27.03% iOS users and 0.71% other mobile operating systems. We will be using Android phones that also shows that we will be targeting the larger group of devices.

By creating an Android app (GyroAccData) for the phone and using OnePlus 5T and Redmi Note 5 Pro in every experiment of data collection. Learning from the Android Developers documentation [14] about the motion and position sensors and implementing it in the app, enabling the sensors, namely gyroscope, gravity and accelerometer sensor on the foreground of the app and display a live graph of the data being generated by them. Whenever a SensorListener is registered, and an event is generated when there is motion detected for sensors. It is handled by onSensorEvent.

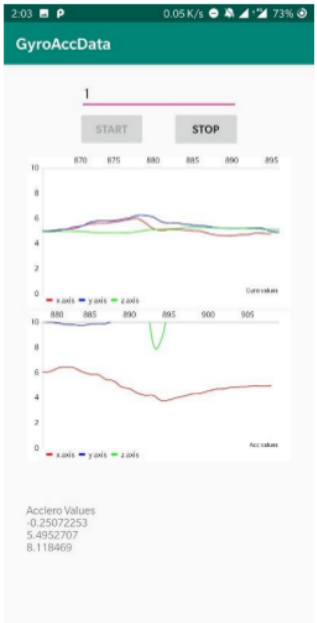

Figure 1: Sudden rise is seen in the graph when the screen is touched.

The top graph shows the gyroscope, and the bottom graph shows accelerometer events.

All the sensors were kept at maximum frequency, and it was observed from the graphs that

on a touch of the screen it creates a spike in the graph as it generates small movements when the screen is touched and released.

In the process of creating this app and checking the functions of the sensors, there was no single permission asked or any message shown on the screen that tells a user that these sensors are being used. Many such sensors are being used for activity tracking like step counting, augmented reality which also requires great computation power. Still, with advancements in processing power, the security measures are set aside for a while and not paid attention to with this detail.

These sensors are called zero-permission sensors which can leak sensitive information and, in our case, we can get the Personal Identification Number (PIN) from the user and his/her motion patterns.

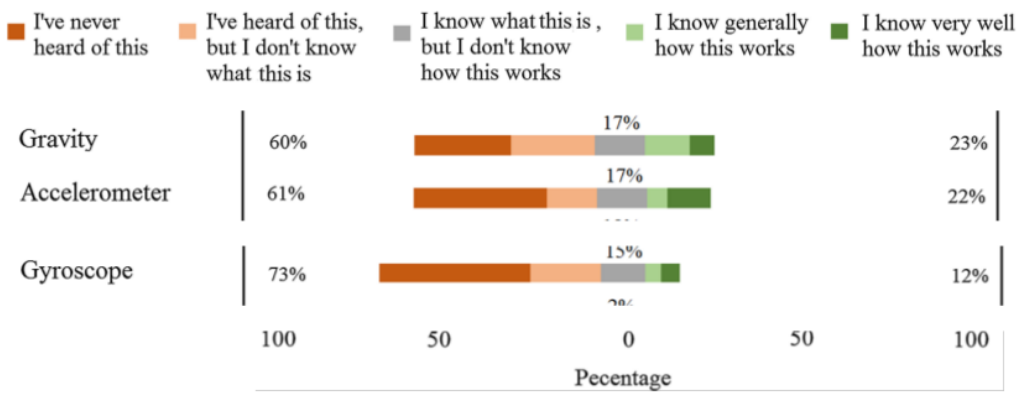

Figure 2: Level of self-declared knowledge about different mobile sensors

Figure 5., a survey conducted and published in [15] (Fig.5), we can see that majority of the people have never even heard about these sensors and hence the vulnerability increases.

2.2 Using sensors for predicting touch

From Figure 5, we know that it is possible to see when the screen is being touched. Using the high-frequency sensors, it is also possible to know what portion of the screen is being touched by the user by looking at the specific pattern in the motion that is being recorded by these sensors. Using the same app (GyroAccData) with modifications that store the data generated by the sensor events and storing them in (.csv) file format, we observe the following graph.

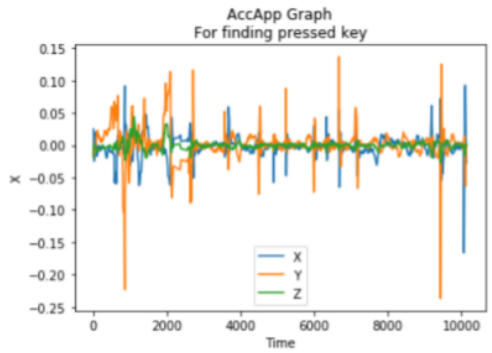

Figure 3: Unique spikes in the graph when different keys are pressed

It was observed that when the phone is held in one hand and kept stable and when numeric keypad (on-screen keyboard) keys are pressed, a unique sensor data pattern is produced which can be used to predict which part of the screen is being touched.

This experiment was tried in different environments and with multiple users. As the number of users increases, the way the phone is held changes. The person can be left-handed, right-handed, may keep the phone in both the hands, use it in one hand, touch with both thumbs or just one or use index finger. By keeping the phone on a stationary material like a table, it was observed that due to the bulge of the camera module at the rear of the phone and also due to the protective cover (may or may not be present), the spikes in the graph were observed, suggesting there might be a chance of predicting data even if it is kept on some stationary material. Different profiling can be done to the way the phone is being held, and machine learning classification models can power the prediction.

2.3 Ways to exploit leakage

Exploring how to exploit a leakage like this to steal PINs. There are two possible approaches. Either we can make a website that uses sensors in the background, or we can build an Android application.

2.3.1 Site

As seen in the implementation of the paper [15], they implemented a web page with embedded JavaScript code to collect the data from voluntary users. On the client-side, they had developed a GUI in HTML5, which brings up the virtual keyboard for users and then randomly generated 4-digit PINs are to be entered by the users. [15]

Figure 4: Site opened on a phone

The sensor data is sent to the database along with the PIN for which the data is held by the file. In a real-world attack scenario, we need to identify the keypress timestamp from the peak amplitudes in the data as the actual timestamp will not be recorded. We can see such implementation in [16].

2.3.2 Android Application

An android application "Train" created by us, uses the Gyroscope, Accelerometer and Gravity sensors to collect data from the users. As soon as the person press the "start" button, the sensor listeners are registered and start saving the data generated in (.csv) file format, with each file containing data for one key press only. As seen in Figure 8, the app also provides a numerical keypad for user to enter PINs.

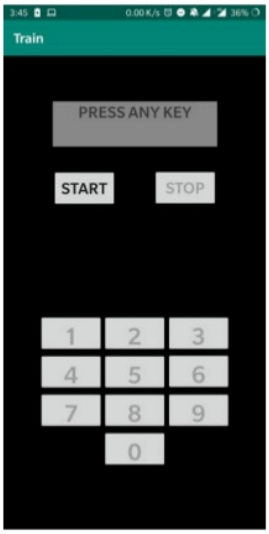

Figure 5: "Train" application that takes sensor data in background

In the training phase, a user is made to hold the phone in one hand and press the keys as displayed on the phone. Each key is to be pressed four times once the "start" button is pushed and all the ten digits having four files each is stored on the external storage. A user press "stop" button and all sensors are unregistered and stop data collection. After each keypress, the corresponding file is saved with the label key press, key down and key up timestamp. Using the onKeyDown and onKeyUp events under Android activity, we can get the required timestamps.

In the future implementation of an actual attack, we can deploy a service that will be running in the background that makes use of sensors and collect data from the user using similar approach from the section 3.3.1. It is discussed in details further in the report.

2.4 Related Work

Two of the well-known published papers on the topic of exploiting smartphone sensor for cracking a PIN are [15][17]. They have shown remarkable results in this research topic, and the methods used by both Berend et al. and Mehrnezhad et al. are different.

Method used by Mehrnezhad et al:

They are considering a set of 50 fixed PINs with uniformly distributed digits. They used a Chrome browser on Nexus 5 device for their studies. They collected data from 10 users entering 50 4-digit PINs for five times, and the experimental results are based on this data

They considered both time-domain and frequency-domain features to get the feature vectors for input. Their feature vector consisted of a total of 114 features.

They used an artificial neural network (ANN). As input, ANN received a set of 114 features for each sample. They had 2488 input samples for ANN.

They trained the network with 70% of data, validating it with 15% of the records, and testing it with the remaining 15% of their data set. In first attempt it could infer the user's 4-digit PIN correctly with an accuracy of 74.43%, and gets better as the number of attempts increases. In single-user mode, the complete dataset was filled by one user alone. Classifier performed better when it was personalized. The attacker could guess the PIN from a set of 81 possible PINs with 71.82% probability of success. [15]

Method used by D. Berend et al.:

The second approach is to identify each digit in the sensor data individually. For this, first, they cut out the corresponding part from the relevant sensor data and then classify the digit presses independently. This approach is more flexible. They chose an LG Nexus 5 for data collection. The application developed runs in foreground and saves it in storage of the phone. A candidate was asked to interact with the application in a consistent natural position. Five times 70 chosen four-digit PINs were entered for training and 50 random PINs for validating. The primarily used classification algorithm was a multi-layer perceptron (MLP) neural network. The network possesses two middle layers. As a general conclusion, they reported a PIN retrieval success of 83:7% after combining data from relevant sensors. [17]

Chapter 3

3 Proposed Method

After observing the results from the experiments performed, we propose a method to identify individual digits of a PIN.

We have decided to separate all the results in the order that it was evolved namely Method 1 Method 2.

3.1 Data collection

To start with training a Machine Learning model, first, we begin with a fundamental step of data collection from a single user. Using the application "Train" as mentioned in section 3.3.2, a user is made to either sit or stand and hold the phone in right hand in one identical position every time. The app collects four types of sensor data

- Accelerometer with gravity: It records the device motion in 3D space and shows the rate of change in each of the three axes merged with gravity. (TYPE_ACCELEROMETER). When the phone is stationary (z=9.8).
- Accelerometer without gravity: It records the device motion in 3D space and shows the rate of change in each of the three axes without merging gravity. (TYPE_LINEAR_ACCELERATION). When the phone is stationary (z=0).
- Gyroscope: Rate of rotation in all three axes.
- Gravity: Force of activity in all three axes.

When the "Start" button is pressed, the sensors are registered and starts collecting data until a number key is pressed by the user as shown on the screen. As soon as the key is pressed, a gap of 1 second is kept to allow the keypress data to be well distributed. After the gap, a file is saved in (.csv) file format from all four sensor types to their corresponding folders.

- Method 1: A new random digit is displayed on the screen. There are 40 such digits in a single session, i.e. a number (0-9) is pressed four times each.
- Method 2: A new digit is then displayed on the screen such that every digit appears before and after each digit at least once similar to [17]. There are 90 such digits in a single session, i.e. a number (0-9) is pressed nine times each.

After completion, the user is asked to press the "stop" button. Multiple sessions are taken on different days or a different time of day to allow a little variation in initial orientation of the

phone as we are presently not removing the initial values from the dataset.

To know if there is a possibility that the digits can be identified if there was minimal phone movement, we collected data by keeping the phone on the table (with protective case), and another 400 files were collected.

3.2 Pre-processing

The raw data collected from the application contains Time and X, Y, Z coordinate values. The file name is coded in such a way that it includes the timestamp of onKeyDown, onKeyUp and contains the number key that was pressed. We have used different methods of pre-processing the data and compared the same with varying models of classification.

For Method 1, we use:

- Windowing
- Three values
- Three values from uniform data

For method 2, we use:

- Windowing 2.0
- Window with uniform time
- Window with uniform time and timestamp (TS) at the end
- Window with uniform time and removing initial values
- Window with uniform time, removing initial values, timestamp (TS) at the end

In method 2, we combine two sensor type data, i.e. accelerometer without gravity combined with gyroscope and accelerometer with gravity combined with gyroscope and also the individual sensor data is used for comparison. When the two sensor type data is combined, different combinations of window size is used (200 or 400) for both accelerometer and gyroscope. The Accelerometer data is converted to a row first, and then the gyroscope data is appended at the end in the same row (more feature vectors).

3.2.1 Windowing

Taking a window of 25 values so that the initial and final values that are noisy and not helpful for classification and will simply confuse a model, such data is removed. So now the window of 25 values contains some initial values, the middle index is this window of (onKeyDown to onKeyUp) data, and the rest of the data is appended according to the space left in this window of 25 values. After stripping the "time" from the data, these values are converted to a single row as feature vectors in a dataset, so that there are (25*3) feature vectors.

3.2.2 Three values

Taking the same window frame of 25 values as windowing, but now we have considered only three values, i.e. first value, mid-value of the key down and key up timestamp and the final value of the window and then it is converted to a single row and put in the dataset as feature vectors after stripping the "time". So, this pre-processing method gives (3*3) feature vectors.

3.2.3 Three values from uniform data

In this, first, we make the data uniform as the data collected by onSensorEvent is not uniform. It will not have data at uniform intervals. So, we take the data and make it uniform in intervals of 5 milliseconds where we fill the gap by taking the nearest timestamp to the interval and copying the corresponding data. After making the data uniform, the same processing is done as in the previous section.

All these three processing methods are used for the window of 25 as specified and also for a window of 30 and applied to batches of files. There are three batches of files.

- On-table data (400 files)
- In-hand data (560 files)
- Hybrid (merged both the files) (960 files)

These batches of files are created from all four sensor data files, and comparison is made for each classification model.

3.2.4 Windowing 2.0

Learning from the results of method 1, it was observed that the window of 25 is too small as the simple key up and key down time stamp itself has a gap of approx. 100ms. So, the window size is now increased (200 or 400 is used). Then the same steps are followed as in windowing. After stripping the "Time" column from the data, these values are converted to a single row as feature vectors in a dataset, so that there are (window size*3) feature vectors.

3.2.5 Window with uniform time

As the data recorded by the sensor is entirely dependent on the "sensor event" and is not continuous, the timestamp is first converted to uniform continuous time stamp such that the missing timestamp has the X, Y, Z coordinate value from the previous row same as used in [17]. Every millisecond has got an X, Y, Z value now, and then the windowing 2.0 is used to get the processed data.

3.2.6 Windowing with uniform time and TS at the end

It is similar to the previous section. The only difference is that instead of keeping the window of key up and key down as the centre of the total window size of 200 or 400. It is kept at the end of the 200 or 400 window size.

3.2.7 Windowing with uniform time and removing initial values

It is similar to the 5.2.5, but before windowing (200 or 400) values the starting 200 values are discarded to remove noise and then from the remaining data the first 300 values of the data file are averaged, and the absolute value from subtraction replaces the original value. It is used with a logic that the initial orientation of the phone should not affect the data arrangement of the same digit file if the phone is held differently. First, we used the method of merely removing the initial first value from the complete file without using absolute, then referring to [17] and looking at their success with such processing; we continued with this method.

3.2.8 Windowing with uniform time, removing initial values, TS at the end

Taking the processed intermediate file from 5.2.7, and keeping the window of key up and key down at the end of the total window size of 200 or 400. It is similar to [17] (differs in window size) used for comparison of our results.

3.3 Classification model

The following are the models that we have tried and used for classification. Taking all types of pre-processed dataset and all of them are run with the same configurations to do justice to the comparison. The parameters set are decided by trying out different combinations of the various settings in each model, and then the figures below show the parameters for the best model that gave the highest accuracy for the train-test split.

Method 1:

- Random Forest:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=10, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

Figure 6: Random Forest Classifier parameters

- Decision Tree:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Figure 7: Decision Tree Classifier parameters

- Logistic Regression:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Figure 8: Logistic Regression parameters

- Naïve Bayes:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Figure 9: Naive Baye's parameters

- KNN:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=7, p=2,
                     weights='uniform')
```

Figure 10:K-Neighbours Classifier parameters

- SVM:

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
```

Figure 11: Support Vector Machine Classifier parameters

- Multioutput: It uses Random Forest in the multi-output classification where the input labels are set such that the keys that are adjacent to the actual key press are given as the input label.

Method 2:

- Logistic regression CV: Increased the max_iter value to allow Convergence of the data.

```
LogisticRegressionCV(Cs=10, class_weight=None, cv=3, dual=False,
                     fit_intercept=True, intercept_scaling=1.0, l1_ratios=None,
                     max_iter=1000, multi_class='auto', n_jobs=None,
                     penalty='l2', random_state=0, refit=True, scoring=None,
                     solver='lbfgs', tol=0.01, verbose=0)
```

Figure 12: Logistic Regression parameters

- SVM: Same parameters as before
- MLP: The hidden layer size is the same as to the window size

```
MLPClassifier(activation='identity', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
              hidden_layer_sizes=(400, 10), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=1000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='lbfgs',
              tol=0.01, validation_fraction=0.1, verbose=False,
              warm_start=True)
```

Figure 13: Multi-Layer Perceptron parameters

- Deep learning model using Keras library


```
{"class_name": "Sequential", "config": {"name": "sequential_1", "layers": [{"class_name": "Dense",
    "config": {"name": "dense_1", "trainable": true, "batch_input_shape": [null, 1200], "dtype": "
    float32", "units": 4, "activation": "relu", "use_bias": true, "kernel_initializer": {"
    class_name": "VarianceScaling", "config": {"scale": 1.0, "mode": "fan_avg", "distribution": "
    uniform", "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "
    kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "
    kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name":
     "dense_2", "trainable": true, "dtype": "float32", "units": 10, "activation": "softmax", "
    use_bias": true, "kernel_initializer": {"class_name": "VarianceScaling", "config": {"scale":
    1.0, "mode": "fan_avg", "distribution": "uniform", "seed": null}}, "bias_initializer": {"
    class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "
    activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}]}, "
    keras_version": "2.3.1", "backend": "tensorflow"}
```


Figure 14: Deep learning model parameters

Using the sklearn library and Keras library for python, the listed classifiers are used.

3.4 Exploitation App

After observing the results from all the above experiments to predict the individual PIN digit, it was time to test it on the field. There are two significant ways by which this can be done, either by using a site or an app. It was observed that after losing focus from the tab which runs the exploitation service, it stops recording the sensor data, so it is not usable. For the app, too, there is this limitation that the user should install the app first.

The functionality of the app must include that it should run a service in the background such that it will not alert the user of its existence. The app "Pin the PIN" was developed keeping this in mind. Due to the updates after the Nougat Android version, it is not possible to keep the service running in the background without keeping a notification in the notification panel informing the user about its existence. So, all the phones having Nougat Android version or below are vulnerable to this exploit once the app is installed on the phone.

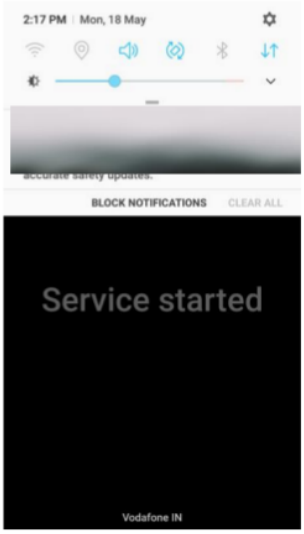

Figure 15: No notification for Android Nougat

The above screenshot is of a Samsung Galaxy A7 running Android version Nougat, and thus the service running does not get displayed in the notification panel and will not be killed.

With new Android 10 in the market, there are 39.2% of Android phones that use Nougat or below (till April 2020). To overcome this low percentage of target phones what can be done is, for the Android versions higher than Nougat the app service will simply sit there in the notification panel saying "Service running in the background" and this notification is given a top priority. If the notification is not displayed, then the service is killed after some time to save power. Considering that the service must run all the time even if the app is opened once,

the notification is to be created by the service. The app can be modified a bit to resemble any app that runs in the background and shows different functionalities from what it announces to be doing. Taking an example of Notification logger app which has to work 24/7 on the phone to log the user notifications, the user is okay with it running in the background as he/she trusts the app, can be one possible modification of the app.

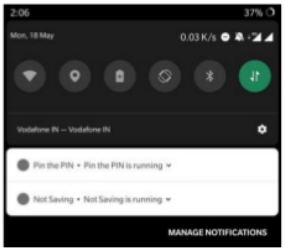

Figure 16: "Service running in background" notification on Android Pie

The above screenshot is from OnePlus 5T Using Android version Pie (9.0). As it can be seen that the notification is the same as the notification logger app “Not saving”.

What the app does is, when the phone is locked, and the screen is off, the data of the sensor is not recorded. As soon as the phone screen is turned on by the user, it starts logging the sensor values. After the phone is unlocked, the recording stops and saves the file in the internal app storage so that it is not available as a media file on the storage and along with that the files are uploaded to the Google Firebase storage so that it is accessible from anywhere. It requires an internet connection to upload the data files, but in today’s world of increased usage of the internet, there are hardly any phones that do not use the internet on the phone regularly. The files which are uploaded to the Google Firebase storage are then downloaded for further processing.

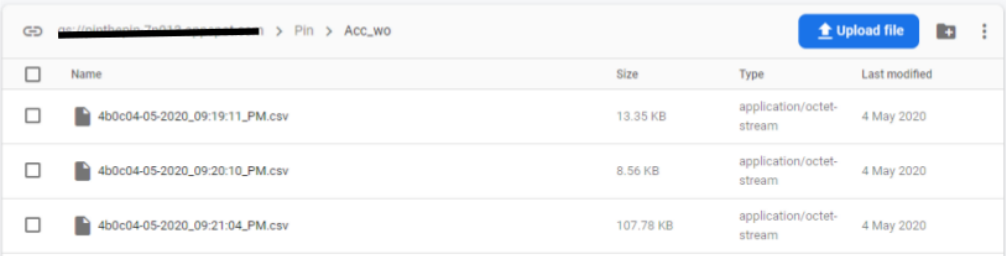

Figure 17: Google Firebase storage where data is uploaded

First, a graph is plotted for the file which contains the accelerometer without gravity sensor data as it shows a proper graph of the linear motion and hence interpretable. It could be seen

that the PIN of length four was entered by observing the five spikes in the graph. The first spike is the one when the user picks up a phone and swipes up on screen to get the PIN keypad and the other four spikes being the corresponding keys that were pressed. There is a little gap between them all, and are split in that way. Now we need to get the timestamp of the keypress by using only the graph of the file. All the four split data files having accelerometer without gravity sensor data is subjected to local minima search, and the lowest value of the Z coordinate is chosen as the timestamp of the key press and is appended in the file name.

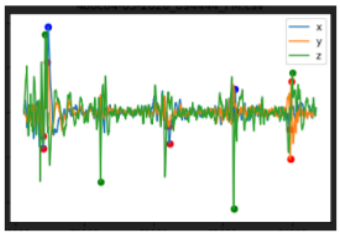

Figure 18: 5 spikes seen in the data for one unlock

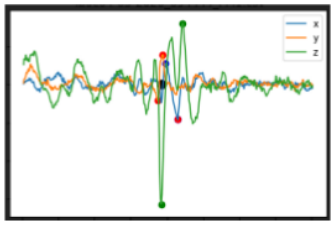

Figure 19: Split and get timestamp for processing (black dot shows timestamp)

The file is then further processed according to the model which we are using to predict the keys, using the same processing method, we get a file with four rows (4-digit PIN). Each key press gives 2-guess data for the key that might have been pressed. All the results are described in Experimental Results.

Chapter 4

4 Experimental Results

4.1 Method 1

It was observed that motion detected by the gravity sensor did not show remarkable changes in sensor data values, hence did not contribute to give accurate prediction (16%). The sensor data from the sensor type accelerometer without gravity also did not provide accurate results (12.3%). We can observe that gyroscope and accelerometer with gravity are the winners among all the other type of sensors. It can be seen in Table 3. It shows maximum accuracy among all kinds of processing methods.

Table 1: Comparison of classification models in method 1

Type of data	Random Forest		Decision Tree		Logistic		Naïve Bayes		KNN		SVM	
	Acc	Gyro	Acc	Gyro	Acc	Gyro	Acc	Gyro	Acc	Gyro	Acc	Gyro
On-table	37.5	42.5	20	34	19	19	22.5	23.75	35	28.75	17.5	35
In-hand	38.93	40.7	27.65	31.2	36.87	41.13	26.54	33.62	32.74	42.47	33.62	42.47
Hybrid	34.71	31.6	29.87	29.46	19.91	26.14	12.43	22.27	36.78	39.89	26.94	27.46

Table 2: Multioutput classification results

	Multioutput			
Type of data	Acc (%)		Gyro (%)	
	First 2	List of 4	First 2	List of 4
On-table	41	62	44	61
In-hand	49.64	72.34	59.57	73.75
Hybrid	43.56	61.82	51.45	67.63

In Table 3, it is seen that gyroscope is giving best results up to 42.47% accuracy of a single-digit prediction when the phone is in hand showing that gyroscope is a sensor that should be given more weight if it is to be merged with another sensor type.

In Table 4, we can see that from the list of 4 output, the actual key matching either first or the second output can be maximum of 59.57% accurate by keeping the phone in hand and using only gyroscope sensor. From the list of 4 output, the actual key present in that list, the maximum accuracy achieved is 73.75 using phone in-hand and gyroscope is the sensor being used.

4.2 Method 2

In the second method, we have used different Classifiers and discarded the previously used models that were not giving good results. The following table shows the maximum accuracies after comparison (the complete data is too big to be displayed). Using 2700 files in total and splitting it to get 25% of test data for validation.

Table 3: Method 2 test accuracies

Model	Type of processing			Accuracy (%)
	Acc w/o grav (size)	Gyro (size)	Type	
Logistic Regression	200	200	5.2.4	64.29
SVM	400	400	5.2.5	58.55
MLP	200	400	5.2.4	66.38
Deep learning model	200	200	5.2.4	64.99

Here, table 5 shows the window sizes used and the type of processing that shows the best accuracies from the complete comparison. All the other processing file types were eliminated. Then the single file that was giving good accuracies for 2 of the models was taken up for exploitation, i.e. accelerometer without gravity and Gyroscope sensor data with 200 window size of each and combined in a row to form feature vectors.

4.3 Pin the PIN

Taking 50 random 4-digit PINs to see how well the prediction works for the actual exploitation. The models are trained with the complete dataset of 2700 files and used for predictions. The following table shows the accuracy of predicting a single digit from all the keys pressed for those 50 PINs (50x4=200 digits) and the count of the correct digits predicted from a 4-digit PIN. The second guess of the digits surely boosts up the accuracy by at least 10% for each model.

(G1 → Guess 1) (G2 → Guess 2).

Table 4: Count of correct digits and accuracy

Model	1 digit		2 digits		3 digits		4 digits		Accuracies	
	G1	G2	G1	G2	G1	G2	G1	G2	G1	G2
Logistic	20	18	15	17	3	8	0	2	29.5	42
SVM	21	19	11	16	1	9	0	0	22	39
MLP	24	18	11	21	1	6	0	1	24.5	41
Deep Learning	11	23	2	4	0	3	0	0	7.5	20

After extending the dataset to 6450 files, the count of the correct digits predicted is as follows along with the accuracies.

Table 5: Effect of extension of dataset

Model	1 digit		2 digits		3 digits		4 digits		Accuracies	
	G1	G2	G1	G2	G1	G2	G1	G2	G1	G2
Logistic	26	12	13	29	1	6	1	1	29.5	46
SVM	19	17	9	16	7	7	0	4	29	43
MLP	19	19	13	20	2	8	0	0	25.5	41.5
Deep Learning	20	21	3	8	0	4	0	0	13	24.5

4.4 Conclusion

Exploring the sensors used on smartphones and exploiting the vulnerability, it is possible to develop a single-digit classification methodology to recover PIN from maliciously captured sensor data. Using different types of pre-processing methods and different classification models to classify a single digit, the comparison of all such models used is shown in Table 3 and Table 4 where maximum accuracy achieved for on-hand data collection method is 42.5% with random forest, in-hand data collection method is 42.47% with SVM and merged data is 39.89% with KNN.

Method 2 includes new pre-processing methods and classifiers. We can observe that the combination of both sensor type, accelerometer without gravity and gyroscope with a window

of 200 each gives a good accuracy for Logistic Regression (64.29%) and Deep learning model (64.99%). Getting a window of 400ms for a digit in the real-world scenario is not possible every time as the speed of typing may be high for a PIN as it is entered very frequently. So, the processing of file with larger window size is eliminated.

It will be safe to say that the PIN lock of a phone is crackable using these motion sensors. Out of 50 4-digit PINs, we can predict at least one digit correct for 38 4-digit PINs. The prediction accuracy of 28.36% for 200 digits is significantly higher than the primary cracking method of trying out every combination. Comparing our results with [15] and [17] the model accuracies are not coming up to the same mark because of the different methods of data collection. Actual exploitation of data and their results are absent in [15] and [17], so the comparison cannot be made for that part. The further exploitation of PIN from the phone and processing the file using only the graph with no proper time stamp reduces the accuracy of the already trained model of ours. Deep learning model did not perform well with the fresh new data because of the small size of the dataset. We can improve the model by adding more data. It is also observed from Table 6 and Table 7 that by increasing the size of the dataset for training, we can improve predicting power of a model.

References

[1] Sensors Overview | Android Developers. https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/sensors/sensors_overview.html

[2] Motion sensors | Android Developers. https://developer.android.com/guide/topics/sensors/sensors_motion

[3] Position sensors | Android Developers. https://developer.android.com/guide/topics/sensors/sensors_position

[4] Tinder, Richard F. (2007). Relativistic Flight Mechanics and Space Travel: A Primer for Students, Engineers and Scientists. Morgan & Claypool Publishers. p. 33. ISBN 978-1-59829-130-8. Extract of page 33

[5] Rindler, W. (2013). Essential Relativity: Special, General, and Cosmological (illustrated ed.). Springer. p. 61. ISBN 978-1-4757-1135-6. Extract of page 61

[6] Corke, Peter (2017). Robotics, Vision and Control: Fundamental Algorithms In MATLAB (second, completely revised, extended and updated ed.). Springer. p. 83. ISBN 978-3-319-54413-7. Extract of page 83

[7] 3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit .... https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[8] https://en.wikipedia.org/wiki/Logistic_regression

[9] Biopython - Machine Learning - Tutorialspoint. https://www.tutorialspoint.com/biopython/biopython_machine_learning.htm

[10] https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222

[11] Decision Tree - GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree/

[12] Our objective is to find a plane that has the maximum https://www.coursehero.com/file/p27f3qtq/Our-objective-is-to-find-a-plane-that-has-the-maximum-margin-ie-the-maximum/

[13] 1.12. Multiclass and multilabel algorithms — scikit-learn .... https://scikit-learn.org/stable/modules/multiclass.html

[14] https://developer.android.com/docs

[15] Mehrnezhad, M., Toreini, E., Shahandashti, S.F. et al. Stealing PINs via mobile sensors: actual risk versus user perception. Int. J. Inf. Secur. 17, 291–313 (2018). https://doi.org/10.1007/s10207-017-0369-x

[16] Narain, S., Sanatinia, A., Noubir, G.: Single-stroke language-agnostic keylogging using stereo-microphones and domain-specific machine learning. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec'14, pp. 201–212. ACM, New York (2014)

[17] Berend, David & Bhasin, Shivam & Jungk, Bernhard. (2018). There Goes Your PIN: Exploiting Smartphone Sensor Fusion Under Single and Cross User Setting. ARES 2018: Proceedings of the 13th International Conference on Availability, Reliability and Security. 1-10. 10.1145/3230833.3232826.

b.tech. project

ORIGINALITY REPORT

15%	12%	8%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	link.springer.com Internet Source	2%
2	eprint.iacr.org Internet Source	1%
3	en.wikipedia.org Internet Source	1%
4	hh.diva-portal.org Internet Source	1%
5	homepages.cs.ncl.ac.uk Internet Source	1%
6	developer.android.com Internet Source	<1%
7	Submitted to IIT Delhi Student Paper	<1%
8	Submitted to Bilkent University Student Paper	<1%
9	Submitted to University of Florida Student Paper	<1%

10	Submitted to Universiti Tenaga Nasional Student Paper	<1%
11	David Berend, Shivam Bhasin, Bernhard Jungk. "There Goes Your PIN", Proceedings of the 13th International Conference on Availability, Reliability and Security - ARES 2018, 2018 Publication	<1%
12	Submitted to University of New South Wales Student Paper	<1%
13	Submitted to University of Strathclyde Student Paper	<1%
14	openaccess.city.ac.uk Internet Source	<1%
15	docplayer.com.br Internet Source	<1%
16	Submitted to University of Bradford Student Paper	<1%
17	Submitted to Dundalk Institute of Technology Student Paper	<1%
18	ethesis.nitrkl.ac.in Internet Source	<1%
19	halshs.archives-ouvertes.fr Internet Source	<1%
20	Submitted to University of Northumbria at	

	Newcastle Student Paper	<1%
21	Submitted to University of Westminster Student Paper	<1%
22	Submitted to Coventry University Student Paper	<1%
23	pypi.org Internet Source	<1%
24	Monica Ionita Ciolacu, Leon Binder, Heribert Popp. "Enabling IoT in Education 4.0 with BioSensors from Wearables and Artificial Intelligence", 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), 2019 Publication	<1%
25	Maryam Mehrnezhad, Ehsan Toreini, Siamak F. Shahandashti, Feng Hao. "Stealing PINs via mobile sensors: actual risk versus user perception", International Journal of Information Security, 2017 Publication	<1%
26	confcats_isif.s3.amazonaws.com Internet Source	<1%
27	www.timtailieu.vn Internet Source	<1%

28	Submitted to National Institute of Technology, Silchar Student Paper	<1%
29	"Advanced Intelligent Systems for Sustainable Development (AI2SD'2019)", Springer Science and Business Media LLC, 2020 Publication	<1%
30	docand.com Internet Source	<1%
31	Submitted to Nanyang Technological University, Singapore Student Paper	<1%
32	gurugovt.com Internet Source	<1%
33	Submitted to CSU, San Jose State University Student Paper	<1%
34	scholar.sun.ac.za Internet Source	<1%
35	Submitted to Nottingham Trent University Student Paper	<1%
36	arxiv.org Internet Source	<1%
37	"ICDSMLA 2019", Springer Science and Business Media LLC, 2020 Publication	<1%

38	Submitted to Imperial College of Science, Technology and Medicine Student Paper	<1%
39	eprints.whiterose.ac.uk Internet Source	<1%
40	machinelearningmastery.com Internet Source	<1%
41	Submitted to Kyungpook National University Student Paper	<1%
42	Submitted to Laureate Education Inc. Student Paper	<1%
43	Submitted to Universiti Kebangsaan Malaysia Student Paper	<1%
44	IFMBE Proceedings, 2015. Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		