

# Deepnet Framework For Malware Analysis in Android Apps

Akshat Agarwal: 17ucs014  
Akshit Devpura: 17ucs016  
Ritik Mishra: 17ucs131

Faculty Mentor: Dr. Shweta Bhandari

# What is Malware:

Malware refers to malicious software perpetrators dispatch to infect individual computers or an entire organization's network. It exploits target system vulnerabilities, such as a bug in legitimate software that can be hijacked.

# What is Android Malware:

Android **malware** is **malicious** software that targets android mobile phones, causing the collapse of the system and loss or leakage of confidential information.



# Necessity to Detect and Remove Android Malware:

Malware is one of the most serious security threats on the Internet today. In fact, most Internet problems such as spam e-mails and denial of service attacks have malware as their underlying cause.



# Why Deep Learning?

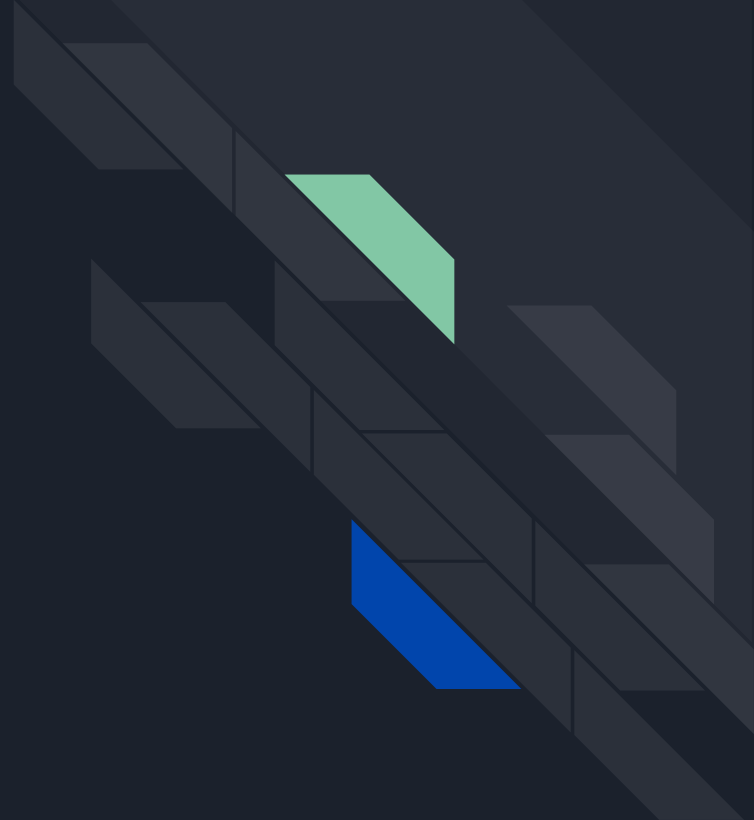
Unlike more traditional methods of machine learning techniques, deep learning classifiers are trained through feature learning rather than task-specific algorithms. it can automatically extract features and classify data into various classes.



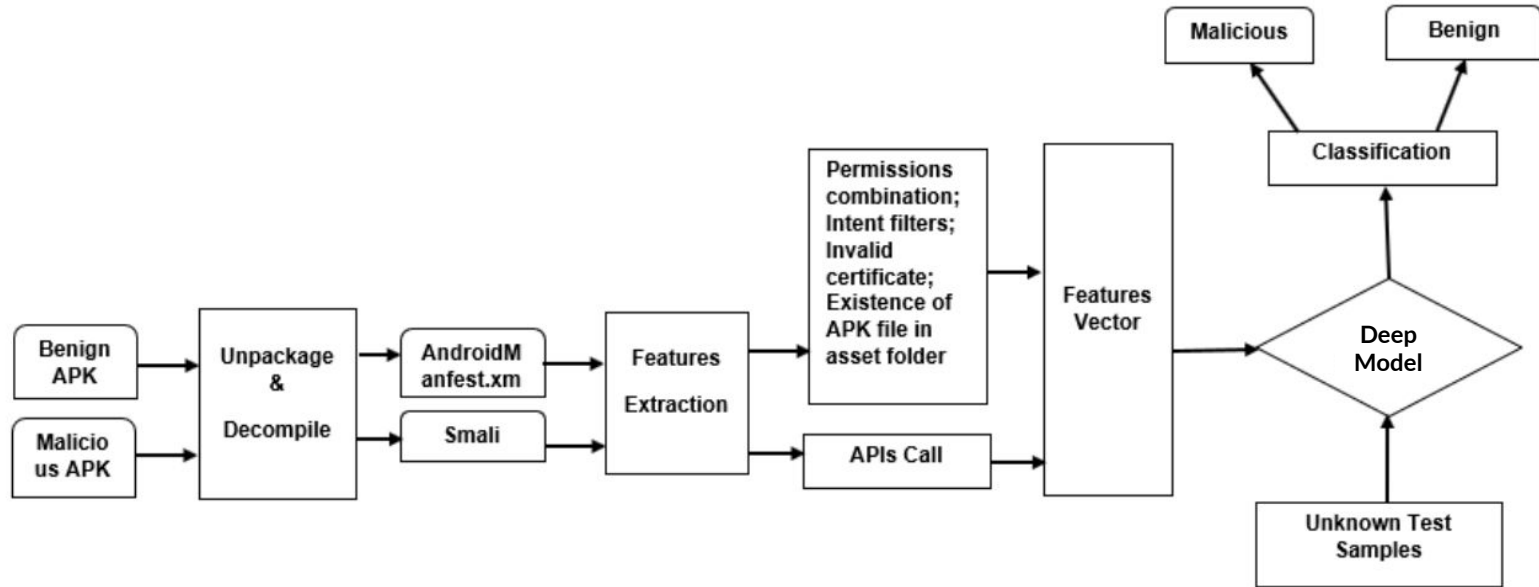


## 3-Phase:

1. Feature Extraction
2. Forming Vectors
3. Detection using Deep Learning



# How does malware detection works?





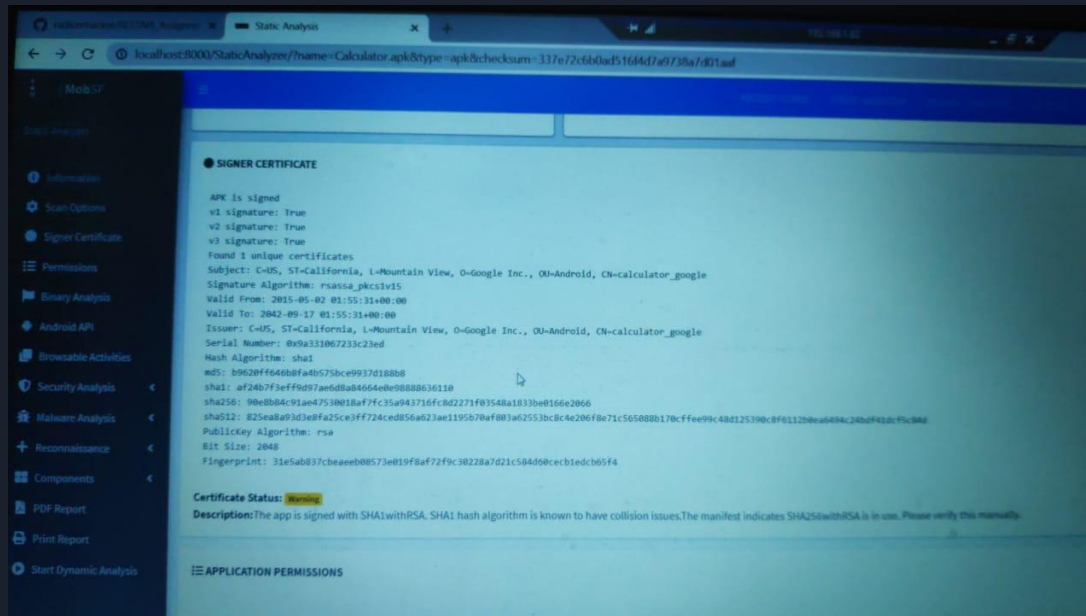
## 1. Feature Extraction:

Features include: permissions, intent filters, intents, api, hardware, services, uri, etc.

We have used AAPT(Android Asset Packaging Tool) and MobSF(Mobile Security Framework) to extract features from an apk file.

Permissions and intent-filters are extracted using aapt(can use MobSF for the same too).

The other useful features are extracted using MobSF.



```
E: intent-filter (line=19)
E: action (line=20)
  A: android:name(0x01010003)="android.intent.action.MAIN" (Raw: "android.intent.action.MAIN")
E: category (line=22)
  A: android:name(0x01010003)="android.intent.category.LAUNCHER" (Raw: "android.intent.category.LAUNCHER")
E: category (line=23)
  A: android:name(0x01010003)="android.intent.category.APP_CALCULATOR" (Raw: "android.intent.category.APP_CALCULATOR")
E: category (line=24)
  A: android:name(0x01010003)="android.intent.category.DEFAULT" (Raw: "android.intent.category.DEFAULT")
```

```
package: com.google.android.calculator
uses-permission: name='android.permission.INTERNET'
uses-permission: name='android.permission.ACCESS_NETWORK_STATE'
uses-permission: name='android.permission.WAKE_LOCK'
uses-permission: name='android.permission.GET_PACKAGE_SIZE'
uses-permission: name='com.google.android.providers.gsf.permission.READ_GSERVICES'
```





# Description of Drebin Dataset.

The dataset contains 5,560 applications from 179 different malware families in csv file named sha-256\_family.

Then there is a folder called feature vectors which has 1,29,013(including those 5,560 malware files also) text files which represented different apps and contained features of the apps.

From this sample of 1,29,013 apps we used a sample of 6000 apps and applied the Deep Learning models on those.



# Categories of Features given in Drebin Dataset

Prefix	SET
feature	S1: Hardware components
permission	S2: Requested permission
activity service_receiver provider service	S3: App Components
intent	S4: Filtered Intents
api_call	S5: Restricted API calls
real_permission	S6: Used permission
call	S7: Suspicious API calls
url	S8: Network addresses



## 2. Forming Vectors for Binary Classification

There will be 2 types of vectors which will be feeded into the deep model.

First vector will be the feature vector which will contain information about the application's features.

Second will be the truth vector which will contain 1 if the app has malware and 0 if it is clean.



# Forming Feature Vectors

For each application, the Drebin dataset contains a text file which describes all the properties of the application.

Each property belongs to one of 8 categories and in each category there are number of individual features.

In each category there are number of features such as for example s1 (hardware components) has many features like -

- "android.hardware.audio.low\_latency"
- "android.hardware.audio.output"
- "android.hardware.audio.pro"
- "android.hardware.microphone"



Permission has these type of features :

"android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS"

"android.permission.ACCESS\_NETWORK\_STATE"

"android.permission.ACCESS\_NOTIFICATION\_POLICY"

"android.permission.ACCESS\_WIFI\_STATE"

"android.permission.BLUETOOTH"

So these amounts to a total of 208 different features.

Feature vector contains 1 if the exact feature is found in the app and 0 if not found.

For example, one application can have a feature vector of [1-0-0-0-1-0-0-0.....]. It means that feature 1 and 5 is present this app and others which have 0 are not.




# Examples of feature vector and truth vector

```
[1000000000000000000100000000000001000000  
00000100000000000010000001010000000000  
0000000001000000000000000100000100000  
000101000010100000001000100000000000  
0000001000000000000000000000000000001  
0011000111100000000001000000]
```

This is snapshot showing feature vectors of 1 app i.e. a single row of the vector.

There are 208 columns having different features from those 8 categories having 1 and 0 if present our 0 if absent.



```
1 1 1 1 0 0 0
```

This is truth vector of 7 apps .

The value is 1 if the app has malware and 0 if the app is clean.

These 2 types of vectors are created for all the apps which are then used in the Deep Learning Model.



# Forming Vectors for Family Classification

For this we formed 2 pairs of vector -

1. First pair contained feature vector (which is already discussed) and family vector.
2. Second pair contained count vector and family vector.
3. Count vector shows how many times a feature of a particular category comes in an app.
4. Family vector shows which family the app belongs to.
5. We then applied deep model on both pairs.





# Ex. of count vector and Family vector

[	2	10	6	5	7	6	7	0]
[	2	1	2	2	1	1	2	0]
[	2	9	4	4	4	3	3	4]
[	5	11	4	4	6	5	6	3]
[	4	17	12	9	9	8	4	4]

This is count vector which shows app in rows and categories in columns.

First app has 2 features from s1 category and 10 from s2 and so on.

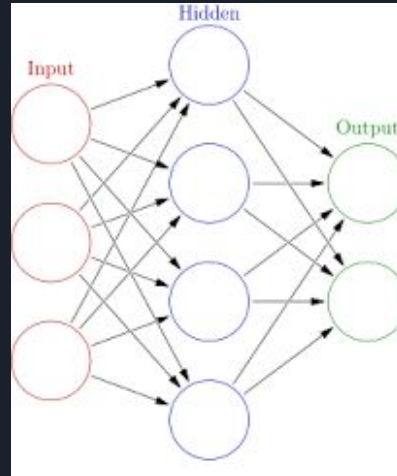
```
[ 31 117 78 44 137 71 71 33 78 12]
```

This is family vector which shows that app 1 belong to family number 31 and second to family 117 and so on.

# What is a Neural Network?

A Neural Network is a circuit of neurons. A neuron is a biological term which in our case takes an input 'x' and gives an output 'y'. Where 'x' is the linear part of the neuron and 'y' is the non linear part of the neuron.

A Neural Network contains an input layer, hidden layers and an output layer





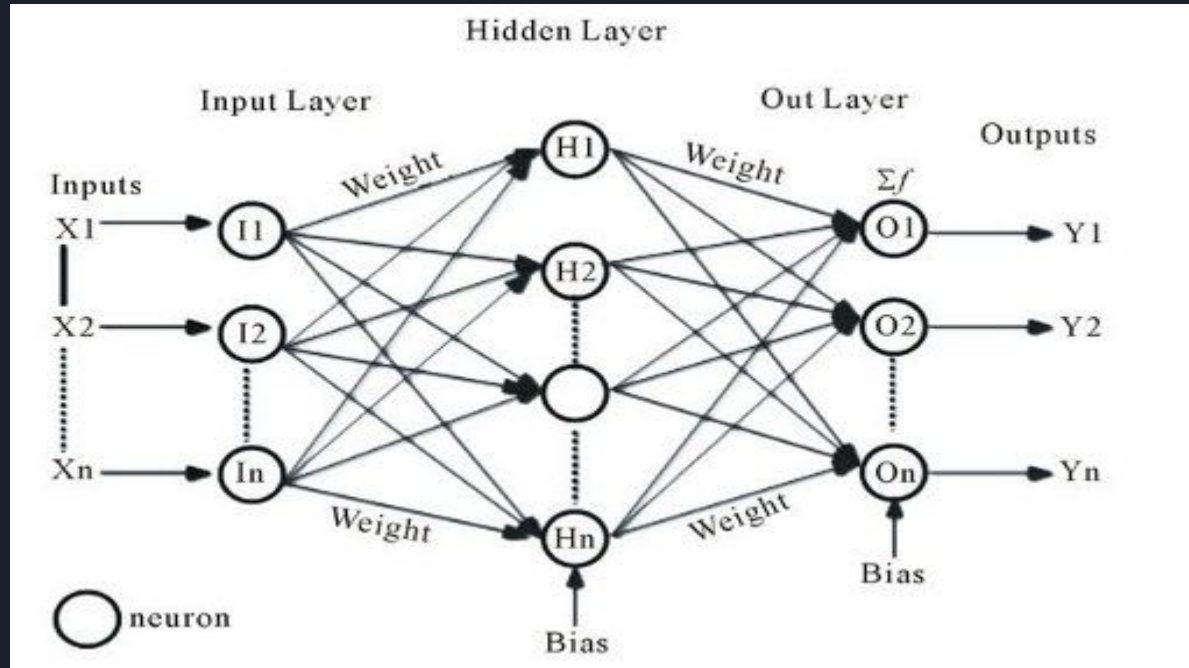
### 3. Deep Model:


In our model we used Multi-Layer Perceptron NN.

#### Multi-Layer Perceptron (MLP)

- It is a class of feed-forward Artificial NN.
- Each neuron in one layer has directed connections to neurons of subsequent layer.
- It uses supervised learning algorithm when trained on a dataset.
- Given a set of features  $X = \{x_1, x_2, \dots, x_m\}$  and a target  $y$ , it can learn a non-linear function approximator for either classification or regression.
- Between the input and the output layer, there can be one or more non-linear layers, called hidden layers.

# MLP Network



- 
- The leftmost layer, known as the input layer, consists of a set of neurons  $\{x_i \mid x_1, x_2, \dots, x_m\}$  representing the input features.
  - Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation  
$$w_1x_1 + w_2x_2 + \dots + w_mx_m.$$
  - followed by a non-linear activation function - like the hyperbolic tan function, sigmoid, ReLu etc .
  - The output layer receives the values from the last hidden layer and transforms them into output values.



# Classification of MLP

- Class MLPClassifier implements a multi-layer perceptron (MLP) algorithm that trains using Backpropagation.
- MLP trains on two arrays:-  
Array X of size (n\_samples, n\_features) :- Holds the training samples represented as feature vectors  
Array y of size (n\_samples, ), which holds the target values (class labels) for the training samples
- It uses fit( X,y) function to train on training set and predict( X\_test) function to give the output for the test set.



# Parameters of MLP

- `Hidden_layer_sizes`: tuple, length = `n_layers - 2`, default=(100,)

The `i`th element represents the number of neurons in the `i`th hidden layer.

- `Solver`: {'lbfgs', 'sgd', 'adam'}, default='adam'  
The solver for weight optimization.
- `Random_state`: int, default=None  
Determines random number generation for weights and bias initialization
- `Alpha`: float, default = 1e-5  
Its for regularization i.e combating overfitting.

```
import read
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
```

```
x_all, y_all = read.read(Load_Data=True)
x_train, x_test, y_train, y_test = train_test_split(x_all, y_all, test_size=0.3, random_state=42)
```

```
models = [MLPClassifier(solver='adam', alpha=1e-5,
                        hidden_layer_sizes=(64, 128, 256, 128, 64, 8), random_state=42)
          ]
for model in models:
    print("Fitting MLP ...")
    model.fit(x_train, y_train)

    print("Evaluating ...")
    y_pred = model.predict(x_test)
    print(y_pred[:15])

    print("Accuracy is %f." % accuracy_score(y_test, y_pred))
    print("-----")
```



# Conclusions

Accuracy in Binary Classification - 96.78%

Accuracy in Family Classification - 81.2%

```
Reading data from CSV file...
Found (5560) malwares in csv file.
Reading dataset files...
Found (6000) files to classify.
Found (271) malware files.
Found (5729) safe files.

Features & Labels arrays' shapes, respectively: (6000, 208) (6000,)
(4200, 208) (4200,)
(1800, 208) (1800,)
Fitting MLP ...
Evaluating ...
[0 0 1 0 0 0 0 1 0 0 0 0 0 0]
Accuracy is 0.967778.
```

```
Reading data ...
Previous data not loaded. Attempt to read data ...
Reading csv file for ground truth ...
Reading positive texts ...
Extracting features ...
Data is read successfully:
(5560, 8) (5560,)
Saving data under data_numpy directory ...
(3892, 8) (3892,)
(1668, 8) (1668,)
Fitting MLP ...
Evaluating ...
[ 31 117  78 ... 137 110  69]
Accuracy is 0.811751.
```



# Goals for next Semester :

1. Preparing Research Paper
2. Understanding more concepts of Deep Learning.
3. Applying other methods to improve the accuracies.
4. Find Better and efficient solutions for Binary classification and family classification.
5. Find more ways to do feature extraction from apk files.



# References :

1. Daniel Arp, Michael Spreitzenbarth, Malte Huebner, Hugo Gascon, and Konrad Rieck "Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket", 21th Annual Network and Distributed System Security Symposium (NDSS), February 2014.
2. <https://scikit-learn.org>
3. <https://en.wikipedia.org>
4. <https://towardsdatascience.com/>
5. Abdelmonim Naway, Yuancheng LI "Using Deep Neural Network for Android Malware Detection" North China Electric Power University, School of Control and Computer Engineering, Beinong Road, Chanping District, Beijing, China, 102206
6. <https://towardsdatascience.com/malware-detection-using-deep-learning-6c95dd235432>
7. eLinux. Android AAPT, June 2010. [http://www.elinux.org/android\\_aapt](http://www.elinux.org/android_aapt). Accessed 13 Apr 2018
8. Schmicker, Robert; Breitingner, Frank; and Baggili, Ibrahim, "AndroParse - An Android Feature Extraction Framework & Dataset" (2018). *Electrical & Computer Engineering and Computer Science Faculty Publications*.
9. <https://github.com/MobSF/Mobile-Security-Framework-MobSF>