Maldy: Malware Detection using NLP and ML on Behavioral Reports

Henil Vyas[1], Shubhra Ranjan Srivastava[2], Zeeshan Ahmad Lone[3], Prof. S.V. Vanjire[4]

[1-3]Student VIIIth Semester, B.Tech, Pune(Maharashtra)
[4]Prof. comp. Department Sinhgad Institiues, Pune(Maharashtra)

Abstract - Android is liable to malware attacks because of its open architecture, large userbase and access to its code. The security investigation is depend upon the Dynamic analysis for the malware detection, in this the binary samples or we can say the system calls are analysed and runtime behavioural profile of the malicious apps is generated and analysed. Resulting report is then used to detect the malware and attribute threat types, using the manually chosen features. But due to the diversity of malware and execution environment it is not scalable, because for every new execution environment new feature need to be engineered manually. MalDy (mal die) is a portable malware detection and family threat attribution framework using supervised machine learning techniques. The explaination of MalDy portability is that the modelling of behavioural reports in sequence of words, together with advanced NLP and ML techniques for automatic engineering of relevant security measures to detect and attribute malware without the intervention of the investigator. More precisely the BOW- NLP is used to represent the behavioral report. On top of the BOW-NLP ML ensemble is constructed. MalDy is then evaluated on various datasets from different platforms and execution environment.

Key Words: **Android, Malware, Win32, ML, NLP, Behavioural Reports.**

1. INTRODUCTION

Malware investigation is a crucial, time consuming and because of the diversity of malware it is more tedious for security investigators. The variety of malware needs to adopt portable tools and techniques for malware detection and threat attribution. Various tools are available for malware detection like static analysis, dynamic analysis of binary code. However, the utilization of static analysis can be problematic for heavily obfuscated malware and is platform dependent. These issues however are addressed partly and covers simple evading techniques. The dynamic analysis solutions on the opposite hand show more robustness towards the evading techniques like obfuscation and packing. For this reason, dynamic analysis continues to be the default choice for malware analysis. Static and behaviour analysis are sources for security measures use to determine maliciousness of a given program. Manual inspection of those features may be a tedious task. For this reason, it can be automated using ML techniques supervised or unsupervised. The supervised ML starts by training a classification model on train-set and afterwards is use on new samples. Unsupervised learning is however more common in malware family clustering but is less common in malware detection. Here emphasis is on supervised ML techniques together with behaviour or dynamic analysis to investigate malicious software.

Malware is a malicious software program with the purpose to wrek computers and servers.

The diversity of platforms and architectures make the malware investigation tougher.

The investigator has to deal with variety of malicious applications which are intended for different platforms. Even the more challenging part being the obfuscation of the malware and the payloads generated for the legitimate applications, which makes it difficult to detect these applications.

The basic motivation being the exploitation of the general behaviour of the malicious apps at the runtime and to attribute them as malicious or as benign.

MalDy (mal die), will be a portable (plug and play) malware detection and family threat attribution framework for the cross-platform apps using supervised machine learning techniques.

2. LITERATURE SURVEY

Sgandurra, Daniele, Mu~noz- Gonz_alez, Luis, Mohsen, Rabih, Lupu, Emil C [1] Automated dynamic analysis of ransomware. It involves that ML is a viable and effective approach to detect new variants and families of ransomware for subsequent analysis and signature extraction.

Severi, Giorgio, Tim Leek, Dolan- gavitt, Brendan [2] Malrec: compact full-trace malware recording for re trospective deep analysis. In: Detection of Intrusions and Malware, and Vulnerability Assessment. This introduces a new malware sandbox system, Malrec.

Chen, Li, Zhang, Mingwei, Yang, Chih- yuan, Sahita, Ravi [3] Semi-supervised classification for dynamic android malware detection.

Alzaylaee, Mohammed K., Yerima, Suleiman Y., Sezer, Sakir [4] DynaLog: an automated dynamic analysis framework for characterizing Android applications. The framework provides the capability to analyse the behaviour of applications based on an extensive number of dynamic features.

Karbab, ElMouatez Billah, Debbabi, Mourad [5] Automatic investigation framework for android malware cyber-infrastructures. The proposed work covers an automatic investigation framework that takes the Android malware samples, as input, and produces a situation awareness about the malicious cyber infrastructure of these samples families.

Paul, Irolla, Filiol, Eric [6] Glassbox: Dynamic Analysis Platform for Malware Android Applications on Real Devices. A system that does not use virtual devices for analysing malware behavior. Glassbox is a functional prototype for the dynamic analysis of malware applications. It executes applications on real devices in a monitored and controlled environment.

3. METHODOLOGY USED IN PROPOSED SYSTEM

The purpose of the system is to detect the malicious application which can have diversity and come from the different platforms such as windows and the android, also the framework is for analysts to perform analysis of diverse, obfuscated applications easily and efficiently. In proposed system for Malware detection the security investigators relay on dynamic analysis in which the binary samples are executed and reports are generated that shows their run time behaviour. The security investigators use the reports to detect the malware with the manually chosen features.

However, the variety of malware and execution environment does not make manual approaches scalable as the investigator needs manually engineered fingerprinting features for new environments.

Therefore, in this model we propose a portable malware detection and Family Threat Attribution Framework, based on supervised machine learning techniques and also using specialized natural language processing.

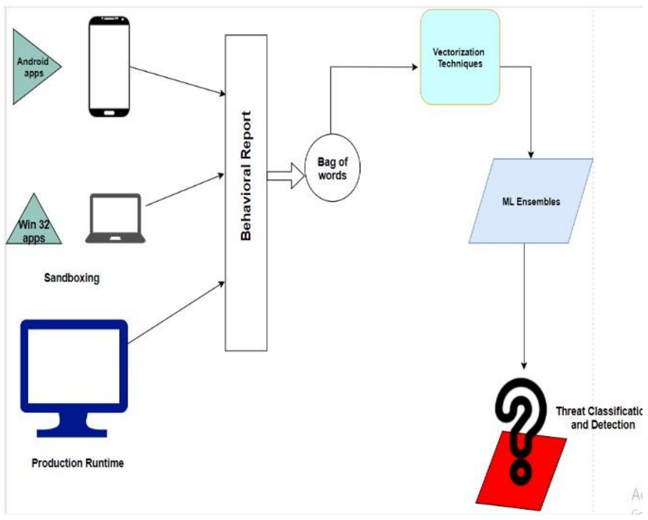


Fig1. System Architecture.

4. MATHEMATICAL MODEL

Algorithm 1. Build Models Algorithm.

```
Input  : $X_{build}$: build set
Output: $Opt$: optimum models' tuples
1  $X_{train}, X_{valid} = X_{build}$
2  for $c$ in $MLAlgorithms$ do
3      score = 0 for $params$ in $c.params\_array$ do
4          model = train($alg, X_{train}, params$) ;
5          s, th = validate(model, $X_{valid}$) ;
6          if $s > score$ then
7              ct = $< c, th, params >$ ;
8          end
9      end
10     $Opt$.add(ct)
11 end
12 return $Opt$
```

Explanation of algo1:

- The above dataset which has been vectorized is used as input to this model.
- We consider our dataset as x-build and divide it into x-train and x-valid. x- valid<<x-train.
- For loop is to iterate over all the ML algorithms.
- The params are the set of data which are unique for all the algorithms.
- Next we train the model on the x-train and param-I dataset.
- To check if the algorithm performs well we use x-

valid set to check the score and threshold.

- If the score better then this generates a tuple ct which denotes that the model is suitable to be used

Algorithm 2. Prediction Algorithm.

```
Input  : report: Report
Output: D: Decision
1  $E_{detection} = E^1_{Detection}$ ;
2  $E_{family} = \{E^2_{Family1}, .., E^T_{familyJ}\}$ ;
3  $x$ = Vectorize(report) ;
4  detection_result = $E_{detection}(x)$;
5  if detection_result < 0 then
6      return detection_result ;
7  end
8  for $E_{F_i}$ in $E_{family}$ do
9      family_result = $E_{F_i}(x)$ ;
10 end
11 return detection_result, family_result ;
```

Explanation of algo 2:

- The trained model is then used to classify the application as themalicious or benign.
- Input will be the report of the file.
- The output will be whether the app is malicious or benign and to which family it belongs.
- The report will be vectorized again.
- The vectorized report is given to our model.
- If detection result is < 0 then the app is malicious.
- Then we iterate over all the families of malwares to check the footprint match and return family if it belongs to any.

Algorithm 3. Feature Vector Computation.

```
Input  : X_seq: Report Word Sequence,
         L: Feature Vector Length
Output: FH: Feature Hashing Vector
1 ngrams = Ngram_Generator(X_seq);
2 FH = new feature_vector[L];
3 for Item in ngrams do
4     H = hash(Item) ;
5     feature_index = H mod L ;
6     FH[feature_index] += 1 ;
7 end
8 // normalization
9 FH = FH / $\|FH\|_2$ ;
```

Explanation of algo 3:

- Input is the behavior report which is collected from our sandbox.
- Output generated will be the vectorized behavioral report.
- We use the n-gram generators to generate the words with the collective meaning.
- We also use the TF-IDF to extract the features from our reports which is widely used in NLP.
- The output then can be fed into our algorithms and we can perform analysis on them. Build model algorithm.

5. EXPERIMENTATION AND RESULT

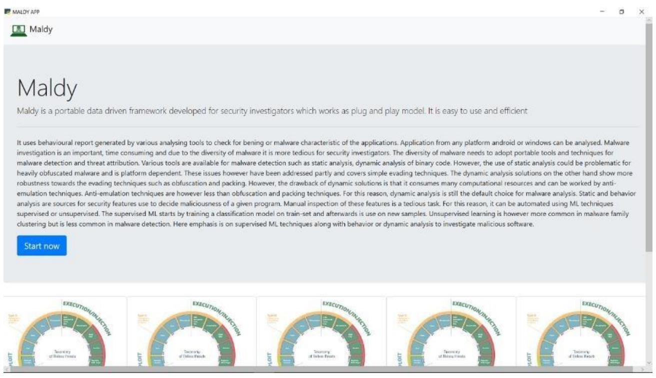
The modules built in the project are enlisted below.

Behaviour extraction: we make use of the cuckoo sandbox and pefile (python library) for the extraction of the behavioural reports and the static information (PE header). We make the use of the api provided by the virustotal for behavioural report extraction at the runtime and the extracted result is forwarded for the virus detection and classification

Behavioural module: this module is responsible for the classification of the file based on its behaviour at the runtime. The underlying syscalls , memory read/write and network activity serve our purpose.

Static module: this modue uses PEfile library to exracct the header information from the file and the classification is done on the basis of the signature of the pefile.

Family classification: in this module we exploit the apicalls. Trained on api calls this module extracts the api information from the file and then the family is attributes to any given malware env.

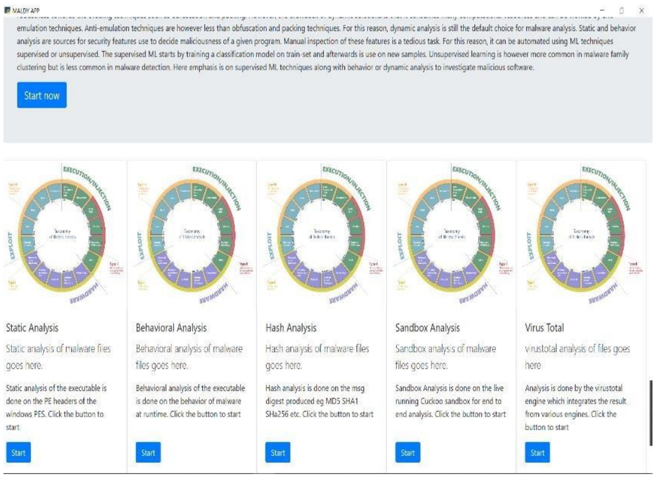


Maldy

Maldy is a portable data driven framework developed for security investigators which works as plug and play model. It is easy to use and efficient

It uses behavioural report generated by various analysing tools to check for bening or malware characteristic of the applications. Application from any platform android or windows can be analysed. Malware investigation is an important, time consuming and due to the diversity of malware it is more tedious for security investigators. The diversity of malware needs to adopt portable tools and techniques for malware detection and threat attribution. Various tools are available for malware detection such as static analysis, dynamic analysis of binary code. However, the use of static analysis could be problematic for heavily obfuscated malware and is platform dependent. These issues however have been addressed partly and covers simple evading techniques. The dynamic analysis solutions on the other hand show more robustness towards the evading techniques such as obfuscation and packing. However, the drawback of dynamic solutions is that it consumes many computational resources and can be worked by anti-emulation techniques. Anti-emulation techniques are however less than obfuscation and packing techniques. For this reason, dynamic analysis is still the default choice for malware analysis. Static and behavior analysis are sources for security features use to decide maliciousness of a given program. Manual inspection of these features is a tedious task. For this reason, it can be automated using ML techniques supervised or unsupervised. The supervised ML starts by training a classification model on train-set and afterwards is use on new samples. Unsupervised learning is however more common in malware family clustering but is less common in malware detection. Here emphasis is on supervised ML techniques along with behavior or dynamic analysis to investigate malicious software.

Figure 2. Maldy overview



emulation techniques. Anti-emulation techniques are however less than obfuscation and packing techniques. For this reason, dynamic analysis is still the default choice for malware analysis. Static and behavior analysis are sources for security features use to decide maliciousness of a given program. Manual inspection of these features is a tedious task. For this reason, it can be automated using ML techniques supervised or unsupervised. The supervised ML starts by training a classification model on train-set and afterwards is use on new samples. Unsupervised learning is however more common in malware family clustering but is less common in malware detection. Here emphasis is on supervised ML techniques along with behavior or dynamic analysis to investigate malicious software.

Static Analysis
Static analysis of malware files goes here.
Static analysis of the executable is done on the PE headers of the windows PES. Click the button to start

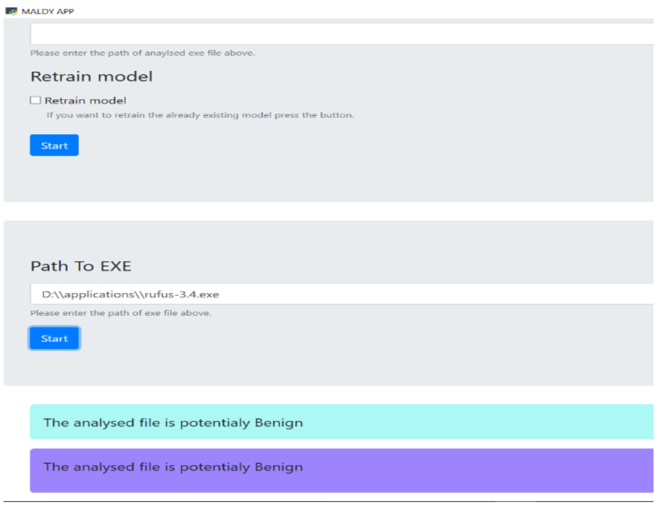
Behavioral Analysis
Behavioral analysis of malware files goes here.
Behavioral analysis of the executable is done on the behavior of malware at runtime. Click the button to start

Hash Analysis
Hash analysis of malware files goes here.
Hash analysis is done on the msg digest produced eg MD5 SHA1 SHa256 etc. Click the button to start

Sandbox Analysis
Sandbox analysis of malware files goes here.
Sandbox Analysis is done on the live running Cuckoo sandbox for end to end analysis. Click the button to start

Virus Total
virustotal analysis of files goes here
Analysis is done by the virustotal engine which integrates the result from various engines. Click the button to start

Fig2.1 Overview



Please enter the path of anaylsed exe file above.

Retrain model

☐ Retrain model

If you want to retrain the already existing model press the button.

Path To EXE

D:\\applications\\rufus-3.4.exe

Please enter the path of exe file above.

The analysed file is potentialy Benign

The analysed file is potentialy Benign

Fig3. Analyzed report

6. CONCLUSION AND FUTURE WORK

The number of daily users targeting the wellbeing of cyberspace is increasing rapidly, which overwhelms the security investigator. The diversity of targeted platforms and architectures mitigates the problem by opening the investigation to new dimensions. Behaviour analysis is an investigative tool for analysing binary samples and producing behaviour reports. In this task, we are looking for a portable, effective solution for malware detection and family retention. The key concept is to model the behaviour report using a bag of words model. We later take advantage of advanced NLP and ML techniques to construct discriminatory machine learning assemblies. On the Maldyome, Drebin, and MalDozer datasets, the Android detection task scores over 94% in the F1-score and over 90% in the detection task. We prove Malady portability by implementing the framework on Win32 malware reports, where the framework achieved 94% on the attribution task. The Maldy performance depends on the execution environment reporting system; And the quality of reporting affects its performance. In the current design, MalDy helps measure this property to help the investigator choose the optimal execution environment. We consider resolving this issue for future research

7. REFERENCES

[1] Sgandurra, Daniele, Mu~noz- Gonzalez, Luis, Mohsen, Rabih, Lupu, Emil C., 2016. Automated dynamic analysis of ransomware: benefits, limitations and use for detection. CoRR.

[2] Severi, Giorgio, Tim Leek, Dolan- gavitt, Brendan, 2018. Malrec: compact full-trace malware recording for retrospective deep analysis. In: Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA

[3] Chen, Li, Zhang, Mingwei, Yang, Chih-yuan, Sahita, Ravi, 2017. Semi- supervised classification for dynamic android malware detection. In: ACM Conference on Computer and Communications Security. CCS.

[4] Alzaylaee, Mohammed K., Yerima, Suleiman Y., Sezer, Sakir, 2016. DynaLog: an automated dynamic analysis framework for characterizing Android applications. CoRR.

[5] Karbab, ElMouatez Billah, Debbabi, Mourad, 2018. Automatic investigation framework for android malware cyber- infrastructures. CoRR.

[6] Paul, Irolla, Filiol, Eric, 2016. Glassbox: Dynamic Analysis Platform for Malware Android Applications on Real Devices. CoRR.