Scientific Journal of Impact Factor (SJIF): 4.72
e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 01, January -2018

Android Malicious Application: Crawling, Analyzing, Detecting

Malware Integration Management System(MIMS)

Sungtaek OH[1], Woong GO[2], Junhyung Park[3]

[1,2,3]*KISA(Korea Internet & Security Agency)*

Abstract —*As smartphone are becoming more common, services using smartphones are becoming more pervasive too. Among them, as mobile banking transactions are increasing, payment fraud is also rapidly increasing. These services handle sensitive information, such as users' personal information and payment information, but as they have several security vulnerabilities, they are attacked by malicious apps. This paper proposes a method of deriving malicious app detection signatures based on the behavior information, obtained by analyzing malicious apps collected through several application distribution channels, and these signatures will be used for analysis of variants of malicious apps and development of rule-based malicious app detection systems.*

Keywords*-Android; Malware; Analysis; MachineLearning; Similarity*

I. INTRODUCTION

Recently the domestic penetration rate of smartphones sharply increased from 65% to 86.4% (2012 to 2015). Also, the number of mobile banking transactions using smart devices like smartphones and tablets was 41.01 million a day on average in the second quarter of 2015. Mobile customers account for all Internet banking transactions. Mobile banking transactions, such as mobile micropayment and banking, are quite pervasive. Along with the growth of the mobile banking market, however, mobile payment fraud like Smishing is sharply increasing, and 68% of all mobile malicious apps circulated around the world include payment-related malicious behavior. To prevent damages due to these payment fraud malicious apps, it is necessary to analyze the malicious apps in circulation, check whether they are malicious apps or not, and block them. Currently, these malicious apps can be analyzed either by statically analyzing the manifest information and the DEX file, which are obtained by deconstructing the APK file, or by installing and executing the APK file in the analysis device and dynamically analyzing. In case of static analysis, however, if code obfuscation was applied to the source code, normal analysis is difficult. In case of dynamic analysis, there are several analysis methods, including activity-based analysis and user interaction-based analysis. This paper proposes Malware integration management system. The system consists of several modules. First, Crawling Module. The crawling module collects apps from public market, black market, etc. Second, Analysis Module. The analysis module analyzes the collected APK files statically or dynamically. Last, Detecting Module. The detecting module determines malicious behavior by using analysis information. It also detects mutable malicious apps by classifying them as similar.

II. CRAWLING

Mobile malware are distributed through a variety of channels, including Smishing SMS, black market and public markets (Google Play, One Store). To detect malicious apps quickly, we need ongoing and proactive crawler that each market-specific. We proposed system for automatically collecting applications from the public market, black market and blog or homepage.

2.1. Public Market

A market in which authenticated apps are provided to users such as Google Play and One Store. APK files can be collected via OpenAPI, available in each market. Public markets detect malicious apps with their own detection system. But, recently, auto-clicking adware "Judy" was registered in the app store to avoid the Bouncer. To bypass Bouncer, the hackers create a seemingly benign bridgehead app and insert it into app store. The malicious apps reached an astonishing spread between 4.5 million and 18.5 million downloads. Our proposed system collects, analyzes, and detects all apps that are uploaded to the public market.

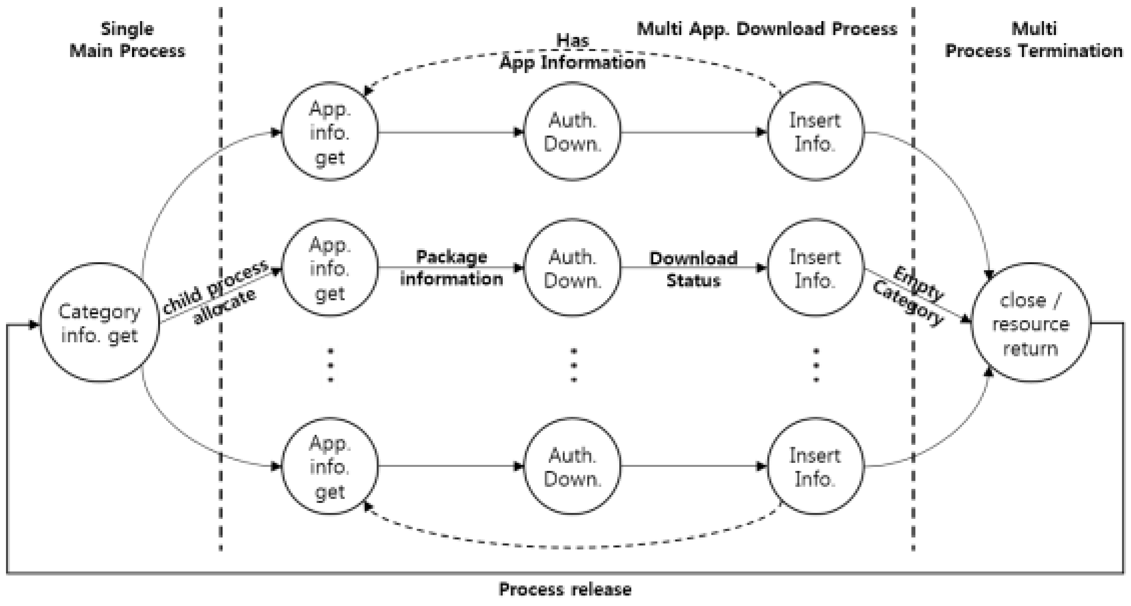


Figure 1. GooglePlay Crawling Flow

2.2. Search Engine

Crawling APK files using search engine such as Google, Naver and Daum. Search for real-time popular keywords through search engines to collect APK files attached to personal blogs or homepages.

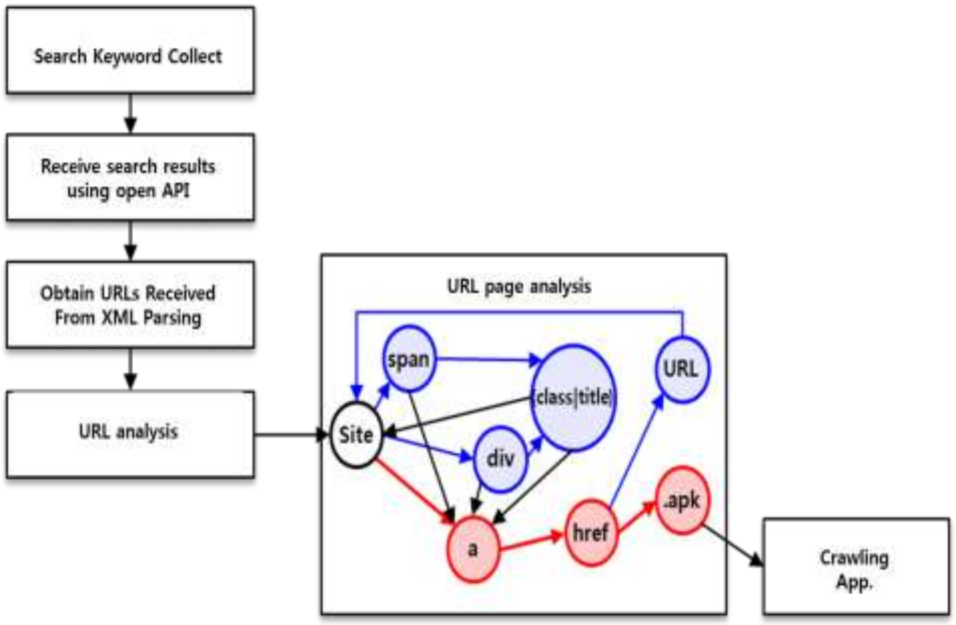


Figure 2. Search Engine Crawling flow diagram

2.3. Black Market

When using black market there is no need for an account or any kind of registration and all apps are free, even those that normally require payment. However, applications uploaded to the black market can be targeted by inserting malicious codes during repackaging. The mobile malware app, 'SMSZombie', distributed by GFAN, China's largest black market, infected approximately 500,000 smartphones in china alone, according to security guard TrustGo. In addition, black markets are not easy to monitor as they are frequently created and their URLs are changed. In this paper, we suggest automatically multi-channel apps crawling system. And black market tracking technology.

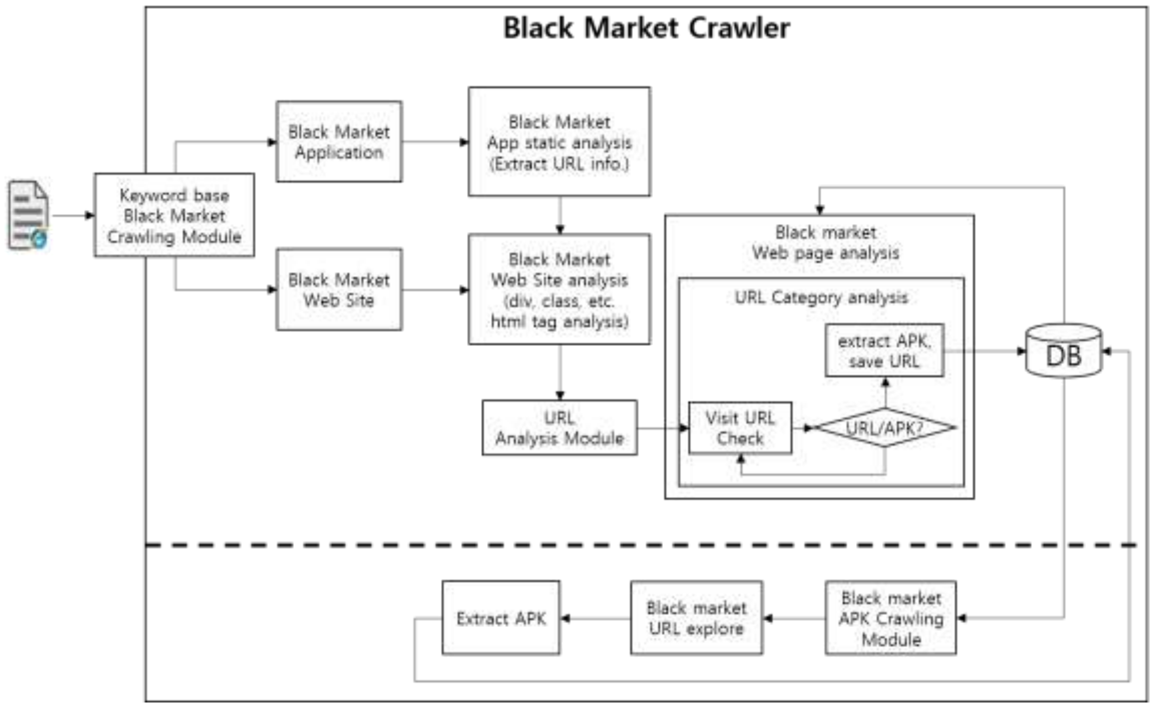

Black Market Crawler
Keyword base Black Market Crawling Module
Black Market Application
Black Market Web Site
Black Market App static analysis (Extract URL info.)
Black Market Web Site analysis (div, class, etc. html tag analysis)
URL Analysis Module
Black market Web page analysis
URL Category analysis
extract APK, save URL
Visit URL Check
URL/APK?
DB
Extract APK
Black market URL explore
Black market APK Crawling Module

Figure 3. Black Market Crawling flow diagram

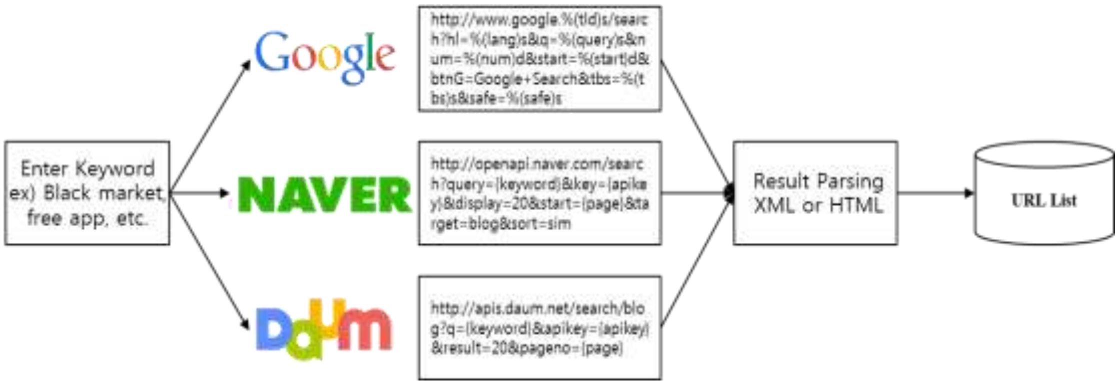
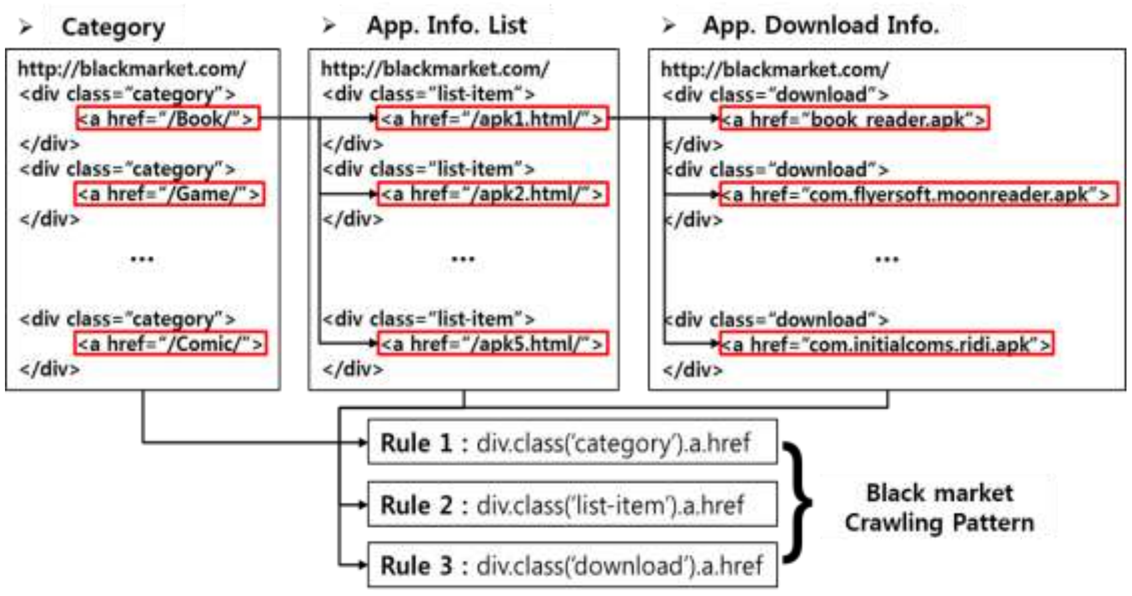

Enter Keyword ex) Black market, free app, etc.
Google
NAVER
Daum
http://www.google.%(tld)s/search?hl=%(lang)s&q=%(query)s&num=%(num)d&start=%(start)d&btnG=Google+Search&tbs=%(tbs)s&safe=%(safe)s
http://openapi.naver.com/search?query={keyword}&key={apikey}&display=20&start={page}&target=blog&sort=sim
http://apis.daum.net/search/blog?q={keyword}&apikey={apikey}&result=20&pageno={page}
Result Parsing XML or HTML
URL List

Figure 4. Black Market Tracking Flow



```
http://blackmarket.com/
<div class="category">
    <a href="/Book/">
</div>
<div class="category">
    <a href="/Game/">
</div>
...
<div class="category">
    <a href="/Comic/">
</div>
```

```
http://blackmarket.com/
<div class="list-item">
    <a href="/apk1.html/">
</div>
<div class="list-item">
    <a href="/apk2.html/">
</div>
...
<div class="list-item">
    <a href="/apk5.html/">
</div>
```

```
http://blackmarket.com/
<div class="download">
    <a href="book reader.apk">
</div>
<div class="download">
    <a href="com.flyersoft.moonreader.apk">
</div>
...
<div class="download">
    <a href="com.initialcoms.ridi.apk">
</div>
```

```
Rule 1 : div.class('category').a.href
Rule 2 : div.class('list-item').a.href
Rule 3 : div.class('download').a.href
```

Figure 5. Black Market Pattern Analysis

III. ANALYZING

3.1. Static Analysis

Static analysis refers to analyzing the manifest information and DEX file obtained by deconstructing the APK file, and Figure 1 illustrates the analysis flow chart for the static analysis system developed in this study. The manifest information in the APK file contains a lot of information about apps, and can decompile the DEX file for static analysis at the source code level.

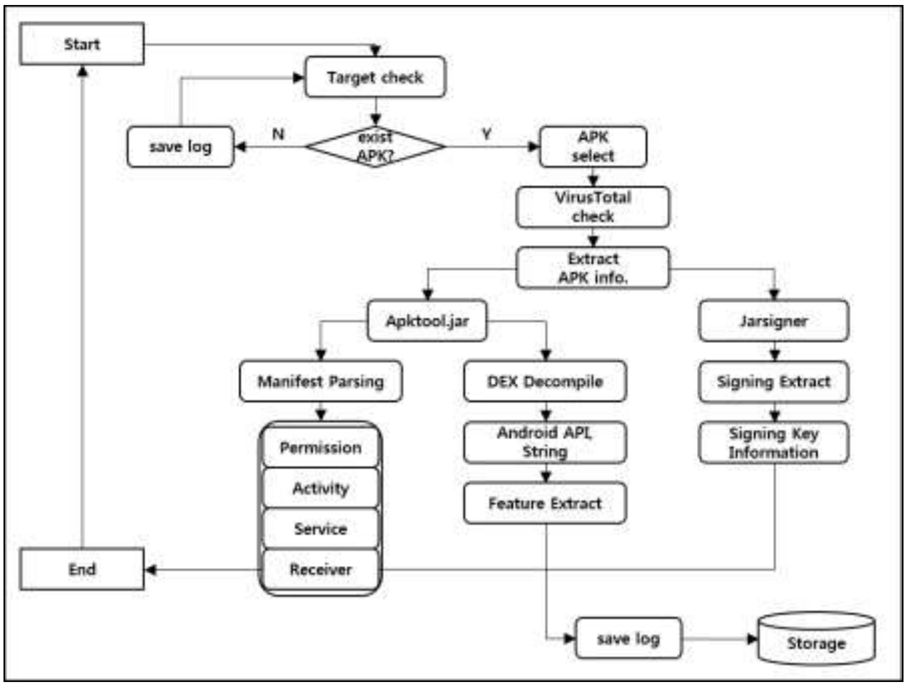


Figure 6. Static Analysis flow diagram

3.2. Dynamic Analysis

Dynamic analysis refers to installing and executing the APK file in the actual device or emulator and tracking and recording how it behaves as it is difficult to accurately know whether it is malicious or not simply by statically analyzing the APK file, and Figure 2 illustrates the analysis flow chart for the dynamic analysis system developed in this study. It tracks Android APIs, System Calls and Network information.

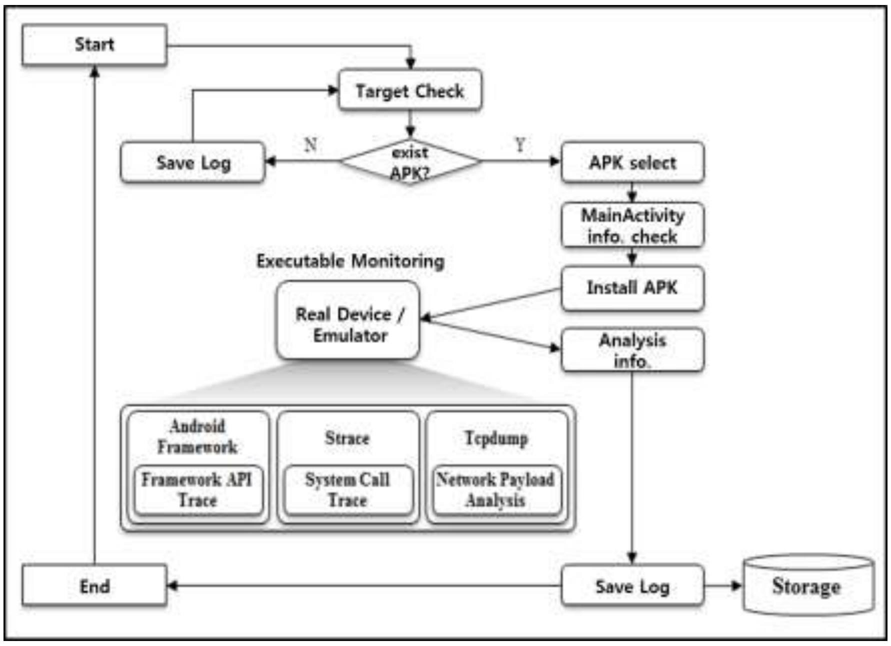


Figure 7. Dynamic Analysis flow diagram

IV. DETECTING

Time to analyze and detect malware before it is widely distributed is crucial to be able to respond quickly. As a result, malware detection using static analysis is needed rather than dynamic analysis that takes a long time to analyze. In the future, we will conduct research on malware detection using dynamic analysis to detect malicious behavior.

4.1. Risk Score

A static analysis provides a measure of the risk score by extracting feature information that can classify the malicious and normal behavior. If the risk score is higher than boundary score, determine it as a malware. The detecting techniques proposed in this paper calculate a risk score using a total of five machine learning algorithms, with the weight per algorithm proportional to the detection rate.

4.1.1. Genetic Algorithm

Features are extracted by comparing malicious and normal apps API, String, System Call, Network information. Genetic algorithm obtain weight by repeating two processes: heredity and verification. The genetic process uses random seed-based genetic algorithms to make a list of the best results for the currently created generation of genes, using selection, crossover, and mutation in the current gene. Genetic feature's weighting process are as shown in Figure 8[1].

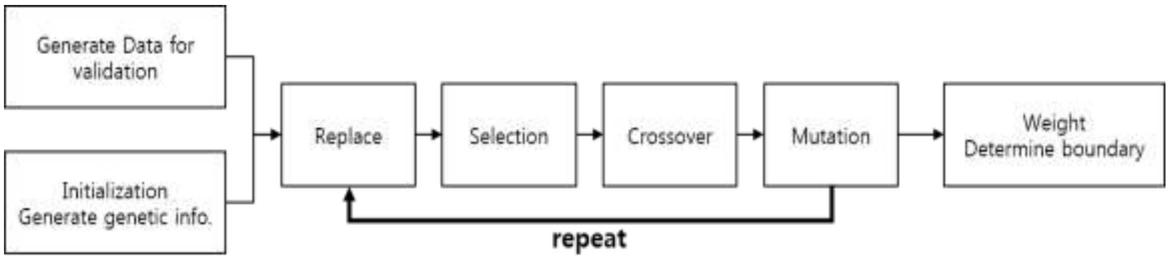
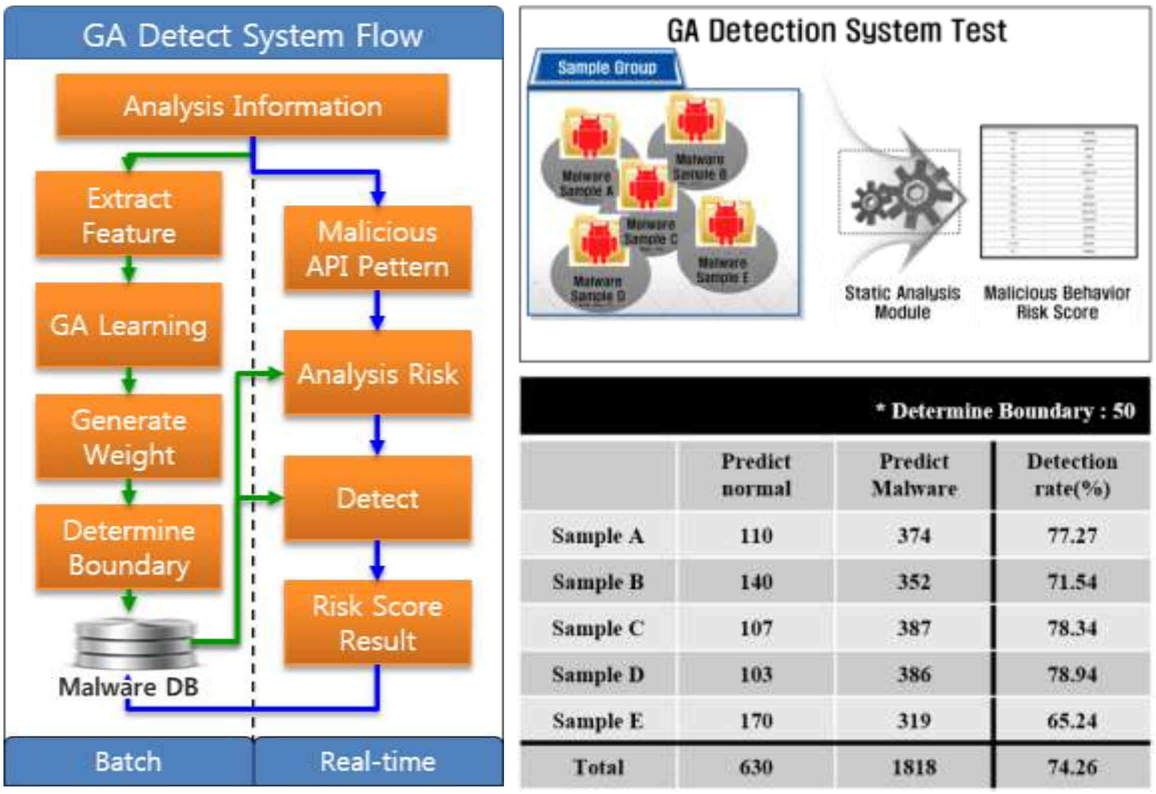

Generate Data for validation
Initialization Generate genetic info.
Replace
Selection
Crossover
Mutation
Weight Determine boundary
repeat

Figure 8. Weighting process



* Determine Boundary : 50

	Predict normal	Predict Malware	Detection rate(%)
Sample A	110	374	77.27
Sample B	140	352	71.54
Sample C	107	387	78.34
Sample D	103	386	78.94
Sample E	170	319	65.24
Total	630	1818	74.26

Figure 9. Flow diagram, Test result

4.1.2. Supervised Learning Algorithms

4.1.2.1. Decision Tree Algorithm

An algorithm commonly used for prediction and classification, which can be used to describe and understand results, is a 'White Box' type of algorithm. It can be used directly for decision making, is quick to process and handles 'Missing

Value' effectively. When we configure a tree, we branch off the node using 'Entropy' and 'Information Gain', and most often with 'Information Gain'.

4.1.2.1.1. Entropy

'Entropy' means the congestion of a given set of data when divided into specific attributes. When a given set of data is composed of different kinds of data, Entropy is higher, and when it is composed of the same type of data, Entropy is lower. The formula for finding Entropy is as shown in Equation 1. S is the given set of data, C is set of classes, Pi is the number of specific classes that fulfil the conditions in the data set, and P'i is the number of specific classes that do not.

Equation 1. Entropy formula

$$p_i = \frac{freq(C_i, S)}{|S|}$$

$$\begin{aligned} Entropy(S) &= -\sum_{i=1}^{m} p_i \log_2(p_i) \\ &= -p_i \times \log_2(p_i) - p'_i \times \log_2(p'_i) \end{aligned}$$

4.1.2.1.2. Information Gain

'Information Gain' means the degree to which data are better distinguished when any feature is selected. Computing Information Gain: information before splitting – information after splitting. A higher value means better discrimination.

4.1.2.2. Random Forest

It is the most typical and most predictable algorithm in the bagging family and is used for classification and regression. Nth Decision Tree algorithms are constructed and results are obtained by voting when making predictions. Bagging algorithms are also used to solve the 'overfitting' problem of decision tress.

4.1.2.2.1. Bagging Algorithm

It is a method of generating multiple Bootstrap data for a given data, modeling and combining each Bootstrap data to produce a final predictive model. Create sub-data sets randomly from a modeled data set of equal size and a learning model for each subset of data. Different algorithms can then be applied for each subset, using the average value for sequential data and voting for categorical data.

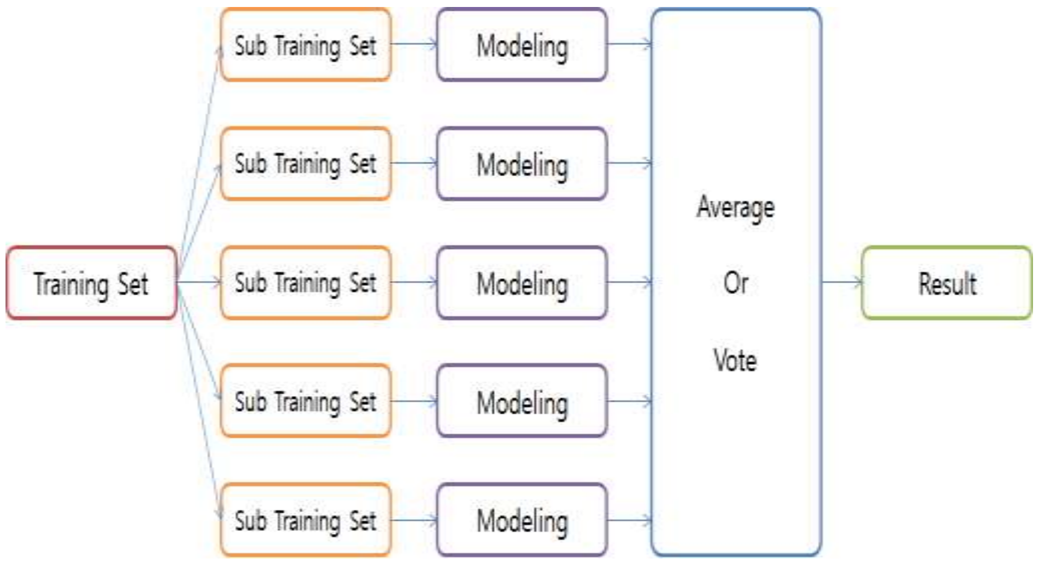


Figure 10. Bagging Algorithm

4.1.2.3. Support Vector Machine

A typical binary classifier used for classification and regression. An algorithm that maximized margins to maximize generalization capabilities. Locate a hyperplane that splits as much of the given data as possible into two groups. The closest point to the hyperplane is called the support vector. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick. Since the kernel draws a different outcome from each type of dataset, it must be based on experience.

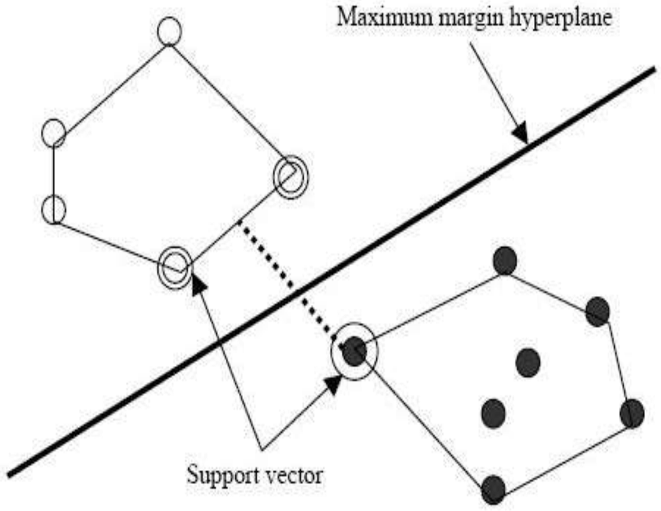


Figure 11. Example of hyperplane & support vector

4.1.2.4. Supervised machine learning detecting system Test

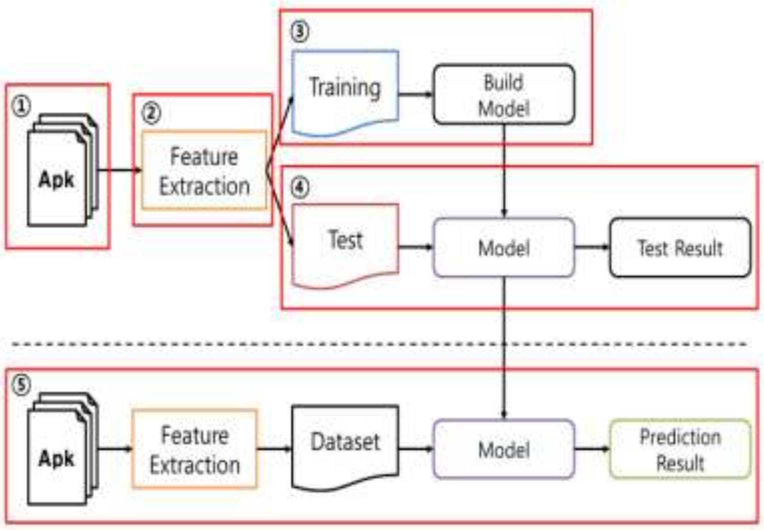
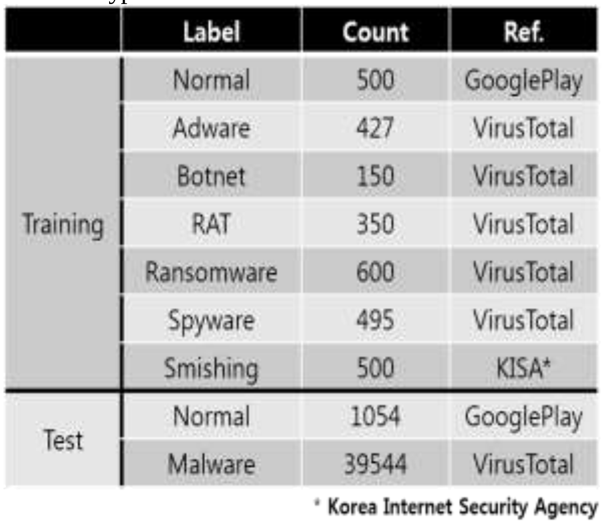


Figure 12. Supervised ML Detecting system process diagram

Learning proceeded via a group of seven types.

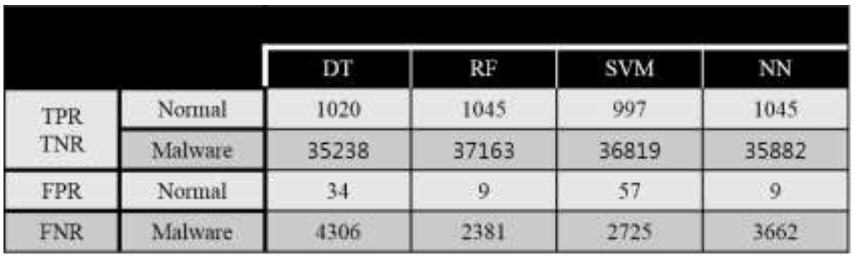


	Label	Count	Ref.
Training	Normal	500	GooglePlay
	Adware	427	VirusTotal
	Botnet	150	VirusTotal
	RAT	350	VirusTotal
	Ransomware	600	VirusTotal
	Spyware	495	VirusTotal
	Smishing	500	KISA*
Test	Normal	1054	GooglePlay
	Malware	39544	VirusTotal

* Korea Internet Security Agency

Figure 13. Training & Test Data Summary

Algorithm test result as shown Figure 14.



		DT	RF	SVM	NN
TPR TNR	Normal	1020	1045	997	1045
	Malware	35238	37163	36819	35882
FPR	Normal	34	9	57	9
FNR	Malware	4306	2381	2725	3662

Figure 14. Each algorithm test result

V. CONCLUSION

This paper proposes an integrated response system for automated malicious applications. Malware Integration Management System (MIMS) to reduce damage caused by malicious application [Figure 15.].

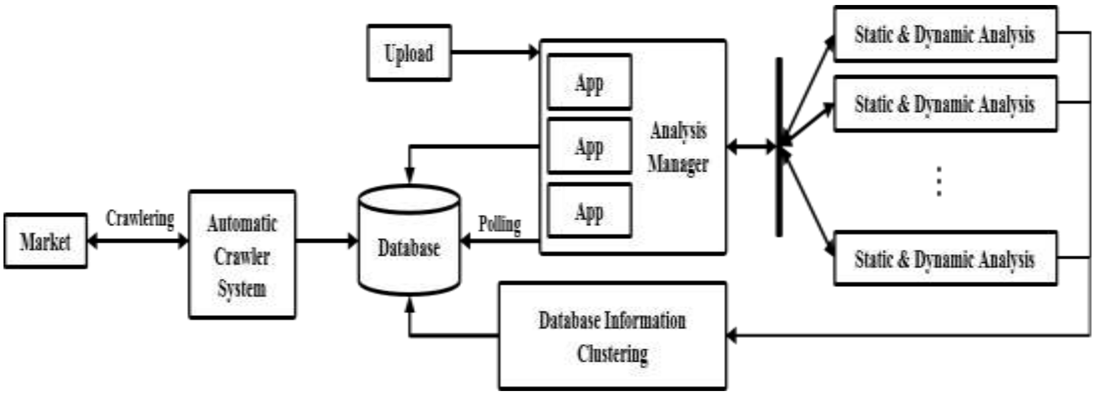

Upload
App
App
App
Analysis Manager
Static & Dynamic Analysis
Static & Dynamic Analysis
Static & Dynamic Analysis
Market
Crawlering
Automatic Crawler System
Database
Polling
Database Information Clustering

Figure 15. Purpose System

In this paper, we proposed MIMS as a method to respond quickly to the spread of malicious apps and infection. First, the system collects malicious apps that are spread across multi-channels. It automatically tracks the black market without user intervention and uses a search engine to search for related keywords. Secondly, analyze (static, dynamic) these collected apps to extract information to determine if they are malicious or not. Finally, based on the information extracted, the system determines if it is malicious. All actions, from collection to detection, are performed automatically to provide 24 hours of malicious app surveillance.

ACKNOWLEDGEMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT)
(No.2015-0-00003, Development of Profiling-based Techniques for Detecting and Preventing Mobile Billing Fraud Attacks)

REFERENCES

[1] CHAN, Patrick PK; SONG, Wen-Kai. Static detection of Android malware by using permissions and API calls. In: Machine Learning and Cybernetics (ICMLC), 2014 International Conference on. IEEE, 2014. p. 82-87.

[2] PEIRAVIAN, Naser; ZHU, Xingquan. Machine learning for android malware detection using permission and api calls. In: *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*. IEEE, 2013. p. 300-305.

[3] YUAN, Zhenlong, et al. Droid-sec: deep learning in android malware detection. In: *ACM SIGCOMM Computer Communication Review*. ACM, 2014. p. 371-372.

[4] SANZ, Borja, et al. MAMA: Manifest analysis for malware detection in Android. *Cybernetics and Systems*, 2013, 44.6-7: 469-488.

[5] SANZ, Borja, et al. Puma: Permission usage to detect malware in android. In: *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions*. Springer, Berlin, Heidelberg, 2013. p. 289-298.