國 立 交 通 大 學

電機工程研究所

碩士論文

CRAT:行動程式碼審閱輔助工具

CRAT: Code Review Assistant Tool for Mobile Operating Systems - FxOS as an Example

研 究 生：張瀚中

Student: Hang-Chung Chung

指導教授：黃育綸　博士

Advisor: Dr. Yu-Lun Huang

中華民國 104 年 6 月

JUNE, 2015

CRAT:行動程式碼審閱輔助工具

CRAT: Code Review Assistant Tool for Mobile Operating Systems - FxOS as an Example

研 究 生：張瀚中　　Student: Hang-Chung Chung

指導教授：黃育綸　博士　　Advisor: Dr. Yu-Lun Huang

國 立 交 通 大 學

電機工程研究所

碩士論文

A Thesis

Submitted to Institute of Electrical and Computer Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfill of the Requirements

for the Degree of

Master

in

Institute of Electrical and Computer Engineering

JUNE, 2015

Hsinchu, Taiwan, Republic of China

中華民國 104 年 6 月

CRAT:行動程式碼審閱輔助工具

學生：張瀚中　　指導教授：黃育綸　博士

國 立 交 通 大 學 電機工程研究所（研究所）碩士班

摘　　要

近年來，由於行動裝置的功能和便利性的提升，使用者日趨增加，愈來愈多有價值的資料也儲存在行動裝置上。相對地，導致行動裝置被攻擊的風險也因此而提高許多。因此如何提升行動裝置的安全性已是不可避免的重要課題，其中主要的攻擊來源之一就是行動裝置的軟體（app）。攻擊者上傳異常app到app商店供使用者下載;一旦使用者啓動該app，行動裝置就可能會遭到洩漏個資或功能癱瘓等攻擊。因此，提升行動裝置安全性的直接作法，就是在每個app上線前都經過審慎的審閱。但是，我們曾提交一支異常app給審閱者，審閱後，卻獲得上架許可。這件事顯示惡意碼不易在審閱過程中被偵測出來。為了提升行動裝置的安全性，以及減輕app審閱者的負擔，我們提出CRAT想協助審閱者更安全、更有效率的審閱app。我們在CRAT中重新設計K-means classification（取名為K'-means classification）來偵測異常app。K'-means classification分為兩階段:Training和Testing。在Training階段，我們把已知的正常app分成數個類別，並建立分類模型。在Testing階段，利用分類模型審閱待審app（即app under review）。假如此app不屬於任何一個正常類別，則此app會被認定為異常app。最後，我們在FxOS的手機上實作CRAT，並評估CRAT的效能。我們的實驗數據顯示，CRAT可以準確地分辨出正常或異常的app（準確率高達9成以上），並且在極短時間內完成app的分類和審閱。60000支app的分類模型的建置時間僅需約0.2秒。實驗結果顯示CRAT可以有效地提升行動裝置的安全性和app審閱效率。

CRAT: Code Review Assistant Tool for Mobile Operating Systems - FxOS as an Example

Student: Hang-Chung Chung Advisor: Dr. Yu-Lun Huang

Institute of Electrical and Computer Engineering

National Chiao Tung University

Abstract

Recently, mobile technologies grow rapidly, more and more valuable personal information is stored on the mobile devices. This leads to a raising risk of mobile devices. One of the major attacks is from the anomalous mobile apps. Attackers exploit the vulnerabilities of mobile apps and launch attacks to the mobile devices. These attacks may cause the system crash or leakage of personal information. To improve the security of a mobile device, mobile apps must be carefully reviewed before they can be pushed to a marketplace. However, when we submitted an anomalous app, which uses up extremely high cycles, to an app reviewer, the reviewer approved the app after couple days. This means it is not easy for an app reviewer to review every line of an app. In the thesis, we propose CRAT to help an app reviewer vetting mobile app. We revise the K-means classification (called K'-means classification) to better detect anomaly apps which can cause DoS attacks (CPU, memory, network I/O). We classify normal apps into groups using K'-means classification and test the app under review with these groups. Three experiments are designed for evaluating the accuracy and performance of CRAT. The results show CRAT can detect anomalies with an accuracy of 90% or above. And the classification and detection process can be done in a short time; 60000 normal apps can be classified within 0.2 seconds.

Chapter 1

Introduction

In this chapter, we introduce the security issues in a mobile network and its importance, then we show the motivation and the contributions of the thesis. The synopsis of the thesis is shown in the last section.

1.1 Mobile Security

Nowadays, mobile OS grows rapidly, mobile users even surpass 68 billions in 2014. And more and more valuable personal information is stored on the mobile device. The risk of mobile devices being attacked is raised. Consequently, the mobile security becomes more important in the present day.

Attackers exploit weakness of communication interface (ex. SMS, GSM...) or software vulnerabilities (malicious applications) to perform attacks to mobile OS. These attacks cause the system crash, leakage of personal information or even paying the bill unconsciously. Hence, we need security mechanisms to protect mobile OS from being attacked. However, existing security mechanisms such as authentication and access control are not enough to protect system from being attacked. A need for more intelligent and sophisticated security like intrusion detection systems (IDS) is necessary.

Basically, the security mechanisms can be classified into two types (**Online** and **Offline**) according to their implementation. In the **Online** implementation, security mechanisms can be implemented on a mobile device. In the **Offline** implementation, security mechanisms can be

implemented on other device such as a computer and the security analysis can be performed through a connection interface. The advantage of the **Online** implementation is more direct and instantaneous. But the **Online** implementation may cause heavy load to mobile OS because of the hardware limitation. The advantage of the **Offline** implementation is that it is not limited to the hardware of mobile device. But the **Offline** implementation cannot real-time detect the anomalies.

To enhanced the mobile security, many works have proposed **Online** solutions for mobile OS. But due to the limitation resource of mobile device, it is hard to widely implement on mobile OS. To propose a practicable solution, we focus on offline implementation in our thesis. The application review process is offline and the first line of defense of protecting mobile OS from being attacked. The application reviewers vet the submitted applications before they are released to marketplace. The application is refused by application reviewer if it is found anomalous. Recently, National Institute of Standards and Technology (NIST) also publishes a standard of vetting the security of mobile applications [**?**]. The standard mentions the importance of mobile security and establish an application review process for companies to follow. Consequently, it is importance and necessary to enhance the mobile security by having a secure application review process.

1.2 Motivation

We submit a malware application which may cause a high CPU load to the mobile OS to the marketplace and test whether the existing security mechanisms used in application review process can detect anomalies. But the application is approved by the application reviewer. Also, we find out it takes couple of days for application reviewer to approve our application. Since under the open-source circumstance, lots of third-party developer submit their applications to

the application reviewer. It is a heavy load for application reviewers to vet any amount of applications in a day.

Consequently, we look for a way to improve the security of application review process. We proposed CRAT to assist an application reviewer in vetting applications before the applications released. CRAT can both improve the performance of application review process and enhance the security of mobile OS (high detection rate). We verify the accuracy and the performance of CRAT in our evaluation.

1.3 Contribution

The contributions of our thesis is that we design a revised version of K-means classification which is named as K'-means classification for anomaly detection (used in CRAT). K-means classification constructs several classifications by training data (ex. classifications of normal applications and malicious applications). Then classify the reviewing applications into one of the classifications. The advantage of K-means classification is that it is simple (low complexity) and has acceptable accuracy (detection rate). The objective of our thesis is to propose a malware detection system that can detect anomalies for using in application review process. So we modify K-means classification to make it available to detect anomalies by only constructing the classification of normal applications. Also, the K'-means classification retains good performance that can detect anomalies in few seconds. The good performance and high detection rate of K'-means algorithm show that K'-means algorithm is suitable to implement on application review process. CRAT can not only improve the security of mobile OS but also reduce the loads of application reviewer.

1.4 Synopsis

The rest of the thesis is organized as follows. In Chapter 2, we introduce background that used in our thesis. And we discuss the related work in Chapter 3. In Chapter 4, we illustrate the K'-means classification we used for anomaly detection. Our proposed tool - CRAT is presented in Chapter 5. In Chapter 6, we state the result of evaluation. The conclusion is shown in Chapter 7.

Chapter 2

Related Work

To improve the security of mobile OS, many works have implemented intrusion detection system on mobile OS. We discussed these works in this chapter.

2.1 Shabtai's work

In 2012, Shabtai et. al. present Andromaly - a framework for detecting malware on Android mobile device [**?**]. The architecture of Andromaly is shown in Figure 2.1. The components of Andromaly are clustered into four main groups: Feature Extractors, Processors, Main Service and the Graphical User Interface (GUI).

Feature Extractors is responsible for communicating with android components and collects the feature metrics of the Android system. And it is triggered by Feature Manager in every pre-defined time. A Processor is an analysis and detection unit. Its role is to receive feature vectors from Main Service, perform analysis and output the results to the Threat Weighting Unit (TWU). Processors can be rule-based, knowledge-based or anomaly detectors employing Machine Learning methods. TWU obtains all results from every processor and apply ensemble algorithm to derive a final coherent decision regarding a device's infection level. The Alert Manager receives the final ranking from TWU and sends an alert to the Main Service. The Main Service synchronizes feature collection, malware detection and alert process. The Main Service decides when to collects new data, performs analysis, handles alert message, configures parameters and outputs results to user. And the last component is the Graphical User Interface

which provides user to configure the Andromaly's parameters and see the result.

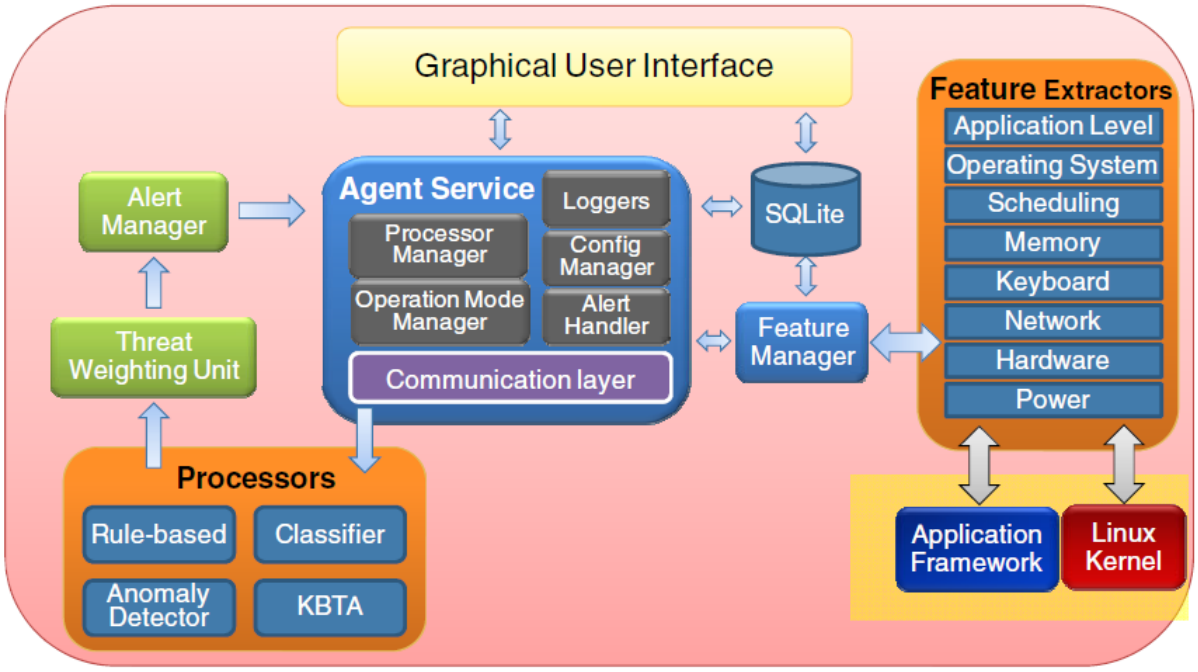

Figure 2.1: The Andromaly architecture.

In this thesis, Andromaly uses Machine Learning algorithms to realize Malware Detection System. The Machine Learning algorithms use the features collected by extractor from the Android system to construct the classifications of normal applications and malware applications. Then classified the observed application to normal or abnormal by the classifier. The thesis evaluates following Machine Learning algorithms (classifiers): k-Means, Logistic Regression, Histograms, Decision Tree (DT), Bayesian Networks (BN) and Naïve Bayes (NB).

The thesis designs four experiments to evaluate average FPR, TPR, AUC and accuracy of every classifiers. The usage of datasets (including 20 normal games, normal tools and 4 malicious applications developed by themselves) in four experiments is shown in Figure **??**. The purpose of experiment I is to evaluate the ability to differentiate between normal and malicious applications when training sets includes all normal and malicious applications and training/testing are performed on the same device. The purpose of experiment II is to evaluate the

ability to differentiate between normal and malicious applications not included in the training set, when training and testing are performed on the same device. The purpose of experiment III is to evaluate the ability to differentiate between normal and malicious applications when training set includes all normal and malicious applications but training/testing are performed on different devices. The purpose of experiment IV is to evaluate the ability to differentiate between normal and malicious applications which are not included in the training set when training and testing are performed on different devices. The result of four experiments are

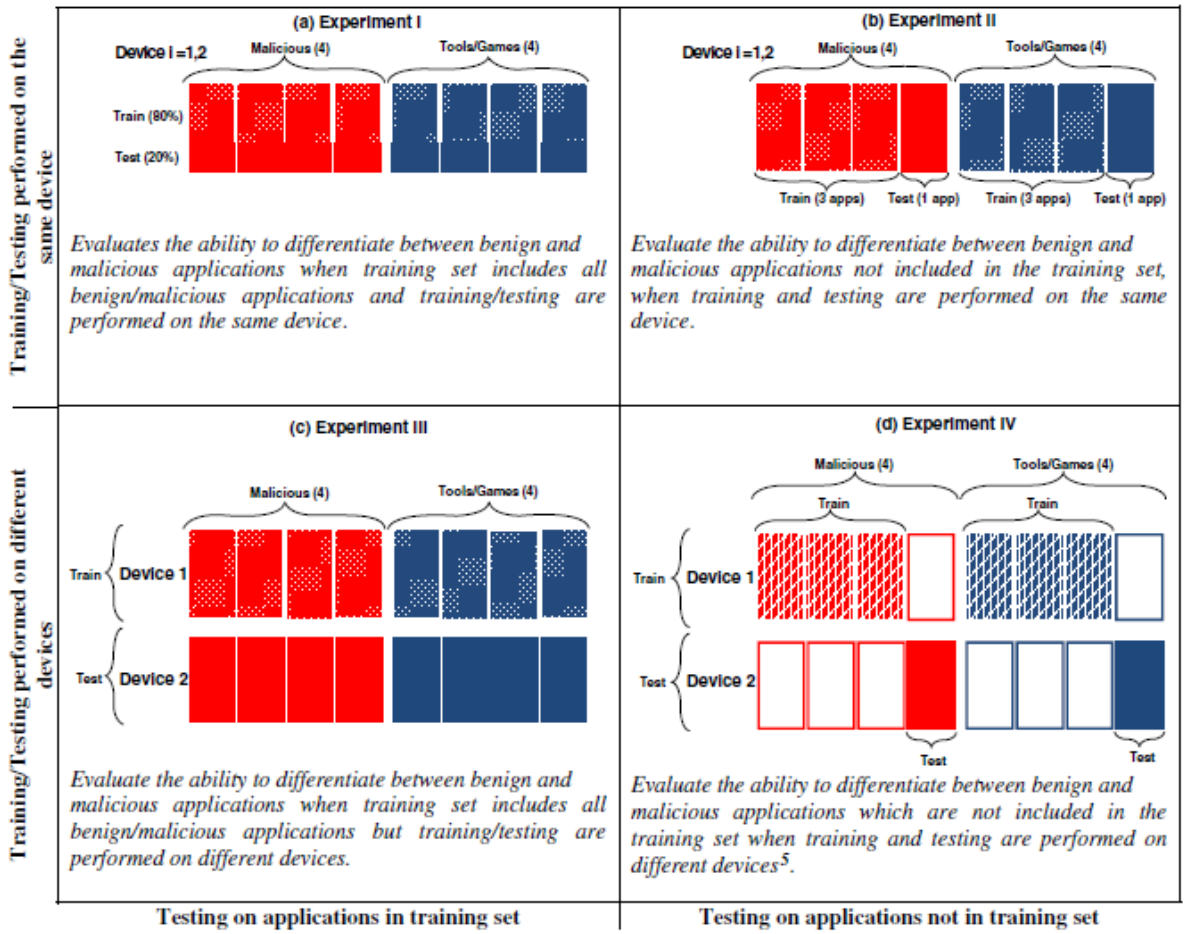

Figure 2.2: Illustration of one dataset set in each experimental configuration.

shown in Figure 2.3. We can see that the best classifiers in experiments I and II are DT, Logistic Regression and NB, while in experiment III and IV the NB out performed other classifiers. Also, the Andromaly can differentiate between normal and malicious applications with high

accuracy and low FPR.

Experiment	Detectors	Games				Tools			
		FPR	TPR	AUC	Accuracy	FPR	TPR	AUC	Accuracy
I	BN	0.000	0.985	1.000	0.993	0.000	0.983	1.000	0.992
	DTJ48	**0.000**	**0.999**	**1.000**	**1.000**	**0.001**	**0.999**	**0.999**	**0.999**
	Histogram	0.081	0.958	0.984	0.934	0.082	0.940	0.982	0.928
	Kmeans	0.115	0.697	0.791	0.813	0.175	0.681	0.753	0.774
	Logistic regression	0.001	0.999	1.000	0.999	0.001	0.998	1.000	0.999
	NB	0.007	0.926	0.995	0.962	0.009	0.901	0.991	0.947
II	BN	0.057	0.436	0.912	0.609	0.044	0.458	0.911	0.658
	DTJ48	**0.113**	**0.907**	**0.895**	**0.908**	**0.122**	**0.796**	**0.837**	**0.860**
	Histogram	0.322	0.630	0.748	0.691	0.279	0.606	0.768	0.709
	Kmeans	0.191	0.549	0.679	0.744	0.166	0.483	0.658	0.710
	Logistic regression	0.118	0.816	0.920	0.874	0.115	0.622	0.789	0.794
	NB	0.103	0.837	0.925	0.891	0.169	0.777	0.838	0.819
III	BN	0.063	0.392	0.874	0.579	0.120	0.464	0.806	0.560
	DTJ48	0.208	0.360	0.572	0.623	0.318	0.509	0.596	0.638
	Histogram	0.367	0.708	0.723	0.660	0.439	0.700	0.719	0.610
	Kmeans	0.154	0.457	0.652	0.692	0.216	0.411	0.597	0.600
	Logistic regression	0.227	0.798	0.916	0.809	0.179	0.451	0.720	0.673
	NB	**0.147**	**0.913**	**0.918**	**0.880**	**0.231**	**0.878**	**0.877**	**0.828**
IV	BN	0.056	0.171	0.735	0.441	0.121	0.216	0.678	0.403
	DTJ48	0.166	0.312	0.573	0.643	0.312	0.343	0.515	0.581
	Histogram	0.425	0.550	0.644	0.527	0.415	0.561	0.682	0.532
	Kmeans	0.167	0.374	0.603	0.669	0.202	0.374	0.586	0.615
	Logistic regression	0.210	0.608	0.803	0.759	0.257	0.311	0.563	0.595
	NB	**0.178**	**0.825**	**0.898**	**0.857**	**0.297**	**0.747**	**0.760**	**0.761**

Figure 2.3: Average FPR, TPR, AUC and accuracy for each one of the detectors.

The contributions of Shabtai's works are proposing a malware detection system - Andromaly and evaluating six classifiers to find out the most suitable classifier under different circumstance. Also Andromaly can differentiate between normal and malicious applications with high accuracy and low FPR. But Andromaly is an application online executed on the mobile device, the thesis doesn't mention the performance of Andromaly. Andromaly may not be practicable because of the complex algorithm running on limited hardware resource of mobile device. Also, it is not reasonable to detect malware applications by constructing a classification of malware

applications. Because the behavior of malicious applications is unknown and may be different between each other.

2.2 Shih's work

In 2012, Shih et. al. proposed a anomaly detection framework which can detect anomalous behavior of the mobile OS caused by an application [**?**]. The architecture of Shih's framework is shown in Figure 2.4. Shih's framework includes two phases (training and analyziing) and each phase has four stages.

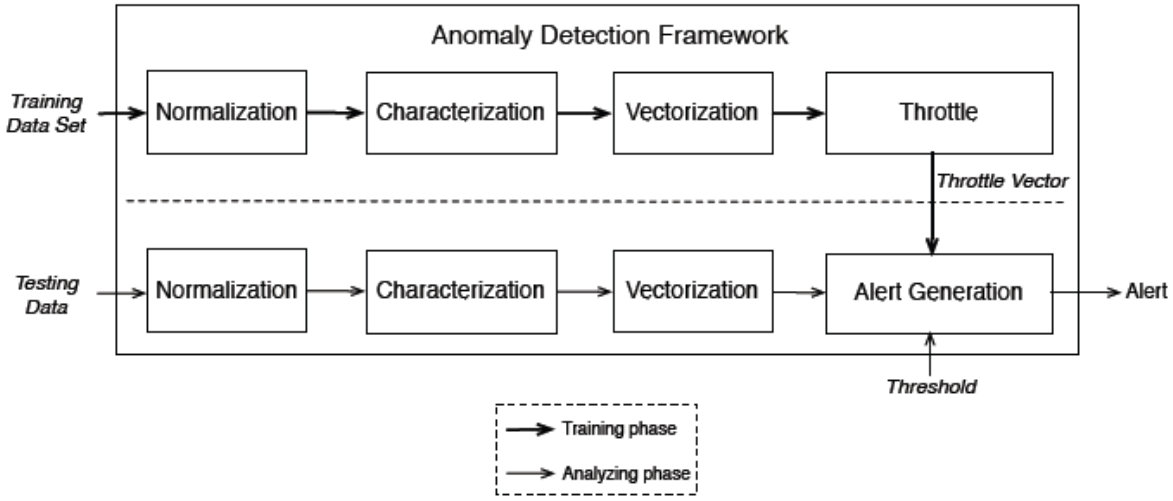

Figure 2.4: Architecture of Shih's framework.

Shih's framework collects multiple parameters for a period of time when an application is in use, and these parameters are the input of the framework. Input data is handled in the same way in both two phases as follows:

1) Normalizeation Stage: an input data sequence of different parameters is normalized with different scales.

2) Characterization Stage: find a *characteristic value* for each normalized data sequence. A *characteristic value* is used to represent the behavior of a data sequence.

3) Vectorization Stage: *characteristic values* of each inspected parameter compose a *characteristic vector*.

After the input data handling, a *characteristic vector* is used in different way corresponding to different phases as follows:

- *Training phase*

  In the training phase, *characteristic vectors* are used to generate a *throttle vector* in *Throttle Stage*. The *throttle vector* is used to represent the behavior of all the training data.

- *Analyzing phase*

  In the analyzing phase, the *characteristic vector* is compared to the *throttle vector* which is gotten from the training phase in Alert *Generation Stage*. There is a *threshold* which is set for the difference between the *characteristic vector* and the *throttle vector*. An alert is generated when the difference is higher than the *threshold*. Note that the *threshold* can be set by the user according to the differ security preference.

Shih's framework is implemented on Firefox OS. And they also design experiment to evaluate the performance of their framework. In the experiment, Test Set 1 is used for the training phase and Test Set 2 is used for analyzing phase.

- TS1

  8 normal applications are used to get the *throttle vector*. The 8 normal applications are: a website with plain text only, a website with graphics only a website with SODUKU game, a website with TIC-TAC-TOE game, WIKI, TWITTER, SOLITAIRE and JEWELS.

- TS2

  8 normal applications and 3 anomalous applications that can cause extreme CPU usage, memory usage and IPC traffic are used to compute the *diff* with the *throttle vecotr* generated in the training phase. The 8 normal applications are: GOOGLE, YAHOO, BAIDU, AMAZON, ACCUWEATER.COM, AIRBNB.COM, GOOSY FARMER and PENGUIN POP.

The *characteristic vector* of each application in TS1 and the *throttle vector* is shown in Figure 2.5. The *throttle vecotr* is obtained by computing the average of all characteristic vectors. The results of the experiment is shown in Figure 2.6. The *diff* of each *characteristic vector* are computed referring to the *throttle vector*. The *diff* is then compared with the threshold set by the user. The three levels of security preference are denoted as SP_1, SP_2, SP_3, each with different *threshold*. Note that, o indicates an application is considered as anomalous and an alert is generated, while x indicates an application is considered as normal.

	CPU	Memory	IPC
Web Page (Plain Text) Avg.	0.637	0.749	0.228
Web Page (Pictures) Avg.	0.600	0.773	0.195
Web Game (Soduku) Avg.	0.680	0.792	0.073
Web Game (Tic-tac-toe) Avg.	0.630	0.617	0.099
Wiki Avg.	0.530	0.809	0.116
Twitter Avg.	0.407	0.629	0.185
Game (Solitaire) Avg.	0.680	0.800	0.112
Game (Jewels) Avg.	0.550	0.737	0.522
Throttle Vector	0.5892	0.7385	0.1912

Figure 2.5: 8 *characteristic vectors* of applications in TS1 and the *throttle vector*.

The results show that the TPR and FPR is highly depended on the *threshold*. If the *threshold* is set too high, both the TPR and FPR raised. In contrast, both the TPR and FPR decreased if the *threshold* is set too low. In the Shih's case, the most suitable *threshold* is $\zeta = 0.35$ which has high TPR and low FPR. The contribution of Shih's work is that they propose an anomaly

	CPU	Memory	IPC	Diff	SP_1 $\zeta = 0.5$	SP_2 $\zeta = 0.35$	SP_3 $\zeta = 0.2$
Google	0.800	0.602	0.259	0.2785	×	×	○
Yahoo	0.780	0.749	0.268	0.2785	×	×	○
Amazon	0.850	0.754	0.750	0.8355	○	○	○
Baidu	0.700	0.745	0.295	0.2213	×	×	○
AccuWeather	0.600	0.812	0.326	0.2190	×	×	○
Airbnb	0.660	0.863	0.210	0.2143	×	×	○
Goosy Farmer	0.770	0.745	0.261	0.2574	×	×	○
Penguin Pop	0.760	0.809	0.071	0.2412	×	×	○
Extreme CPU	0.950	0.6579	0.009	0.3608	×	○	○
Extreme MEM	0.750	0.990	0.076	0.4124	×	○	○
Extreme IPC	0.820	0.776	1.000	1.0773	○	○	○
				TPR	87.5%	87.5%	0%
				FPR	66%	0%	0%

Figure 2.6: Testing Result.

detection framework which can detect anomalies without knowing the behavior of malware applications. But as we mentioned that Shih's framework is highly depended on the *threshold* which is set by the user. Although the *threshold* is tunable, it is impracticable because under different circumstance may need different suitable *threshold*. Also, Shih's framework does not consider that different applications may have different behavior. In spite of all the training applications are noraml, they may still have different behavior.