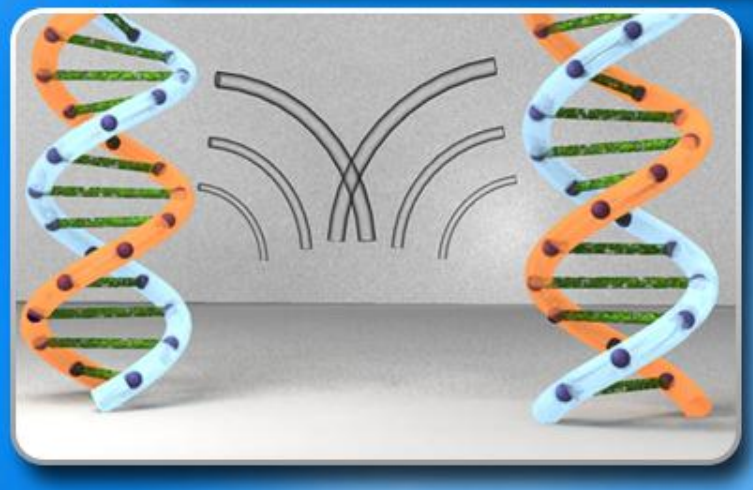
Virus Bulletin 2013



The Droid Knight: a silent guardian for the Android kernel, hunting for rogue smartphone malware applications

Next Generation Intelligent Networks Research Center (nexGIN RC) http://www.nexginrc.org/
Institute of Space Technology (IST) http://www.nexginrc.org/
National University of Computer and Emerging Sciences (NUCES) http://www.nu.edu.pk/
Islamabad, Pakistan

Authors - Introduction

M. Ali Akbar (Email:ali.akbar@nexginrc.org)

- Research Engineer, nexGIN RC, IST, Pakistan

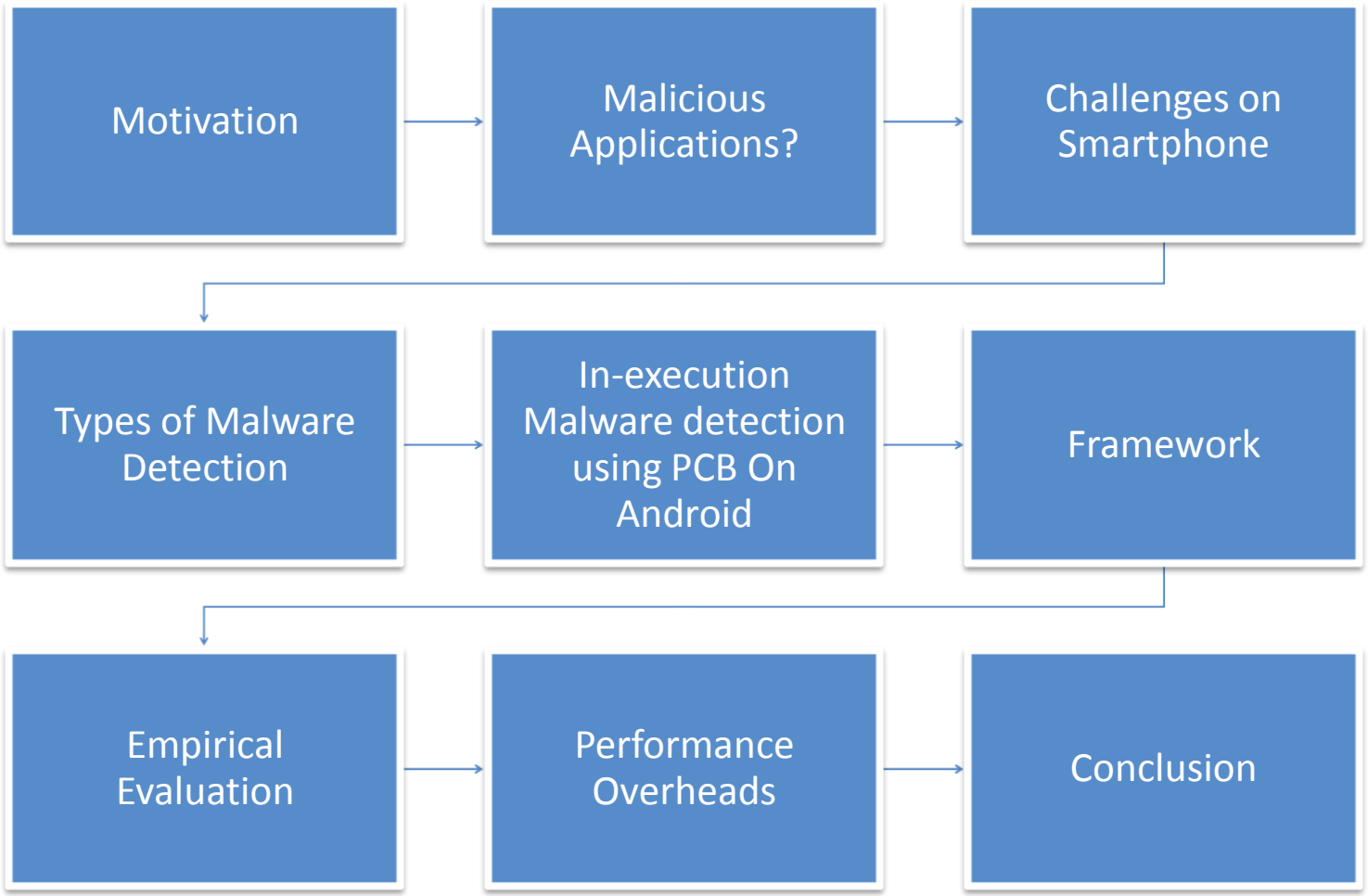
Farrukh Shahzad (Email:farrukhshahzad0@yahoo.com)

- PhD Candidate, FAST-NU, Principal Malware Researcher, Ebryx (Pvt) Ltd. Pakistan

Muddassar Farooq (Email: muddassar.farooq@nexginrc.org)

- Director nexGIN RC, Dean IST, Pakistan

Key points


Motivation
Malicious Applications?
Challenges on Smartphone
Types of Malware Detection
In-execution Malware detection using PCB On Android
Framework
Empirical Evaluation
Performance Overheads
Conclusion

Motivation

Market Share (1st Quarter 2012) - Gartner report 2012

- Android Market Share : 64.1%
- iPhone Market Share : 19%
- Symbian Market Share : 6%
- RIM Market Share : 5.9%
- Microsoft Market Share : 2.6%

Market Share (2nd Quarter 2013) - Gartner Report 2013

- Android Market Share : 74.4%
- iPhone Market Share : 18.2%
- Symbian Market Share : 0.6%
- Black Berry Market Share : 3%
- Microsoft Market Share : 2.9%

Threats in 2011-12 - Android - Malware samples

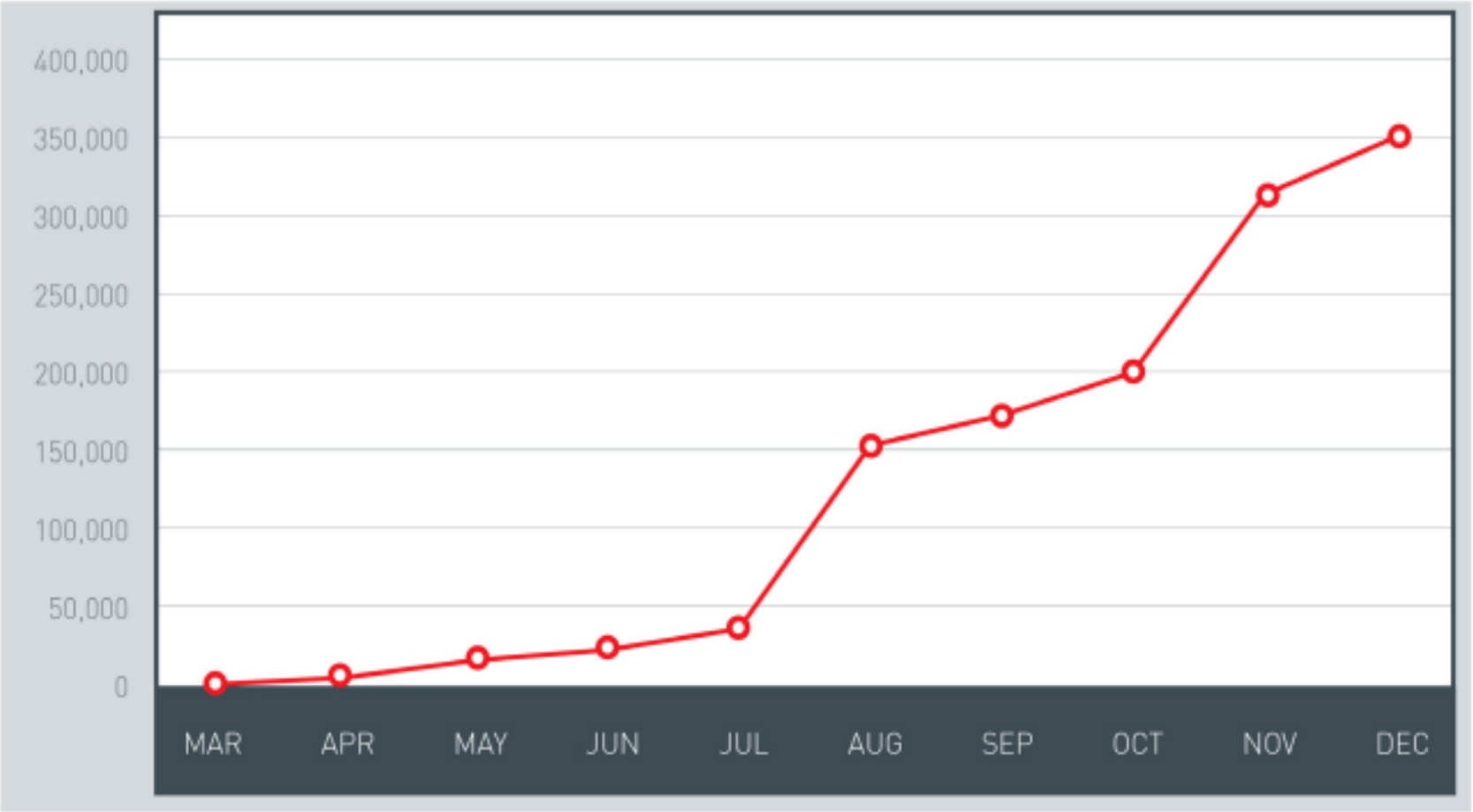
- Increased from 400 to 13302 (an increase of 3325%) – 2011 [Juniper Report 2012]
- A sixfold increase in malicious applications 30000 -- 175,000 [Trend-Micro 3rd Quarter 2012]

Malware Categories in Android [Juniper Report 2012]

- Spyware - - 63.39%
- SMS Trojan -- 36.43%
- SMS-Flooder -- 0.09%
- Worms -- 0.09%

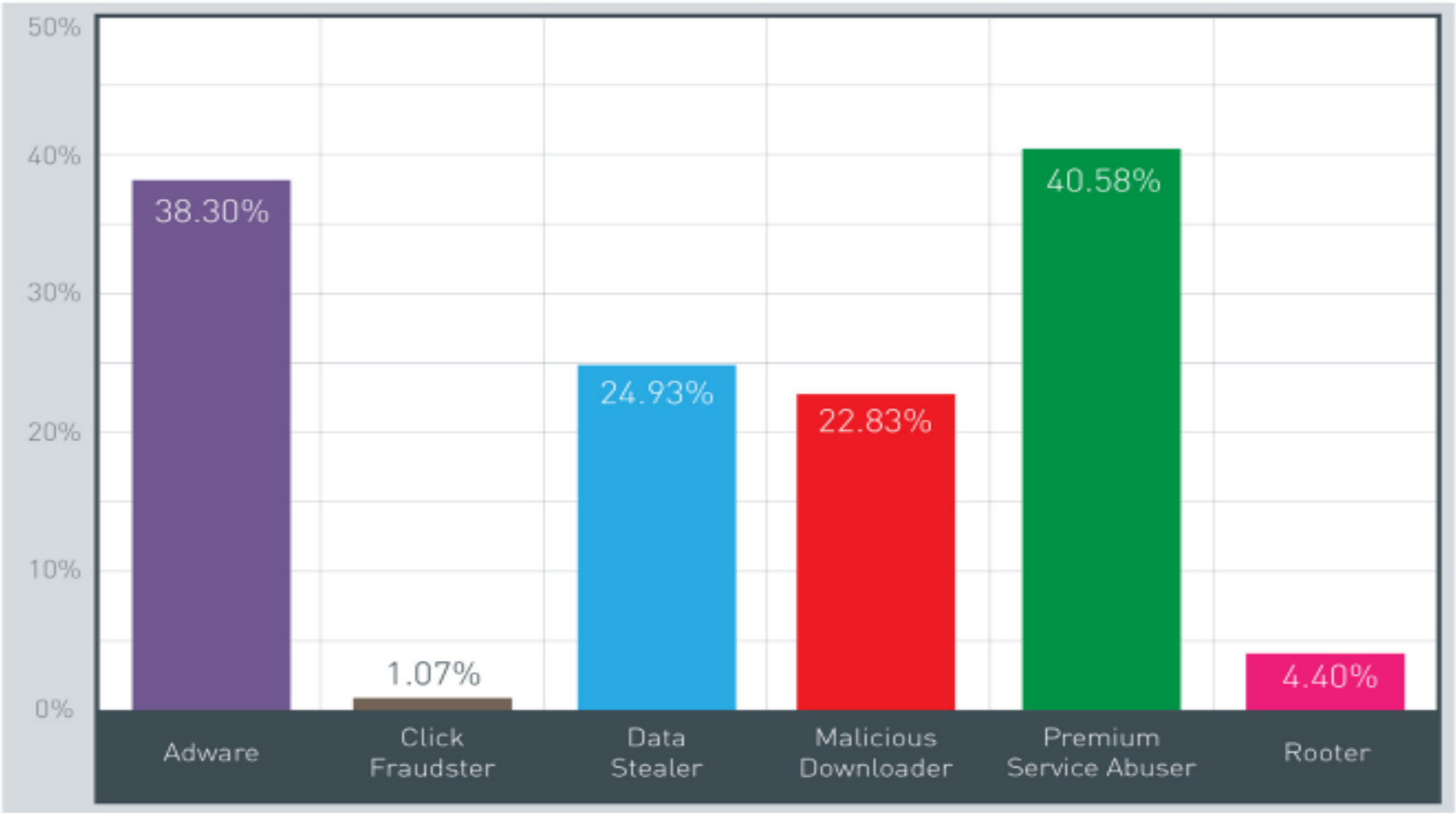
Motivation (Android threat Growth)

TREND MICRO: Mobile Threats and Security Round up Report 2013



Motivation (Distribution of threats)

TREND MICRO: Mobile Threats and Security Round up Report 2013



Malicious Applications

Worm

- A malicious code that usually spreads by exploiting vulnerabilities in the network services is termed as a worm. e.g. iOS Ikee Worm, Commwarrior etc

Virus (only seen on windows mobiles)

- Viruses infect normal processes on a system and use them to execute their malicious code. They usually spread through sharing of infected programs. Duts etc.

Trojans

- They pose as legitimate and applications, execute their malicious codes in the background without user's knowledge. e.g. ZeuS , SMS.AndroidOS.Foncy and Skull.D etc.

Spyware

- Spyware present themselves as useful programs but their primary objective is to steal sensitive information about a user such as passwords, emails etc. e.g. GPSSMSSpy and Nickyspy

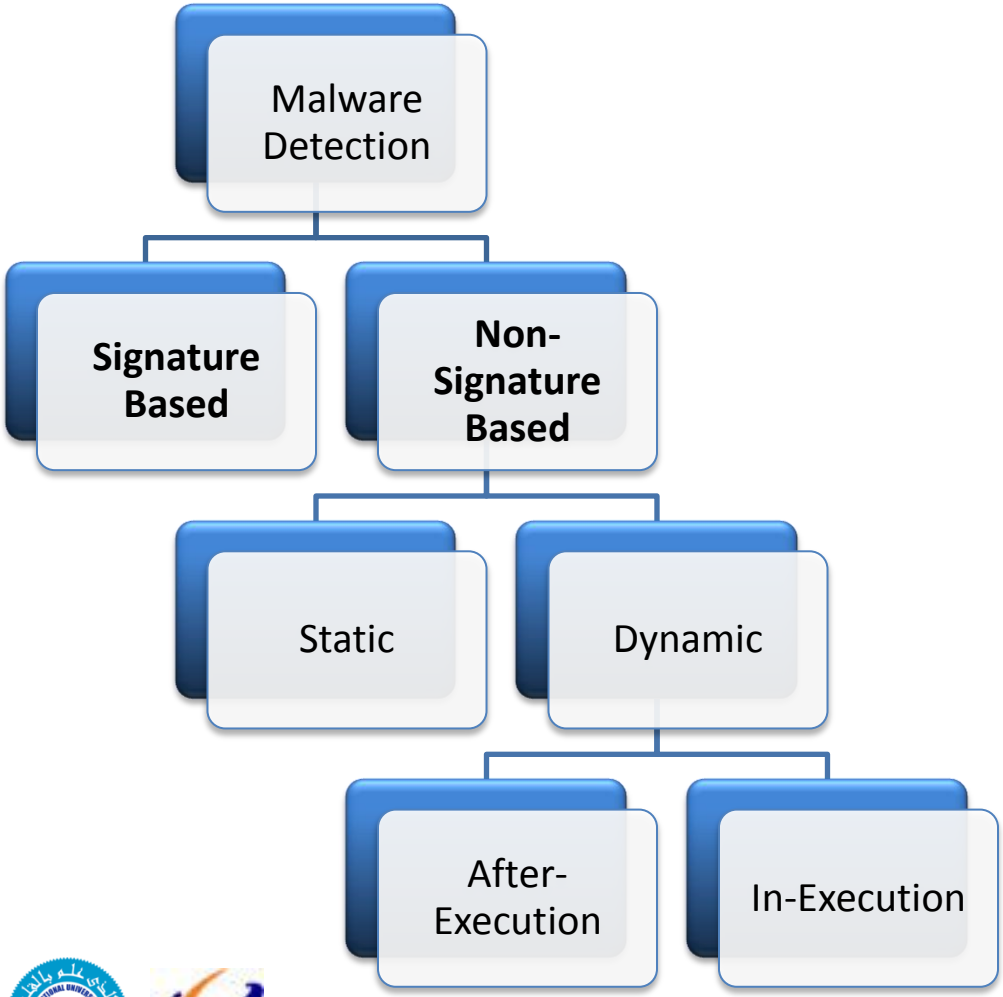
RootKits

- Rootkits hide a malicious application on a system by modifying a system's kernel by modifying system calls are instrumented. e.g. ITFUNZ and Z4Mod

Malware Detection on Smartphones
Major Challenges

- High Detection Rate
- Low False Alarm
- Detection Delay (minimal)
- Phone's Performance Degradation (minimal)
- Robustness to Evasion
- Battery Constraints
- Processing Power
- Processor Architecture Issues (desktop solutions are not directly portable)

Types of Malware Detection



Signature Based:

· Detection on the basis of known byte sequences

· Unable to detect new malware

· Regular updates required

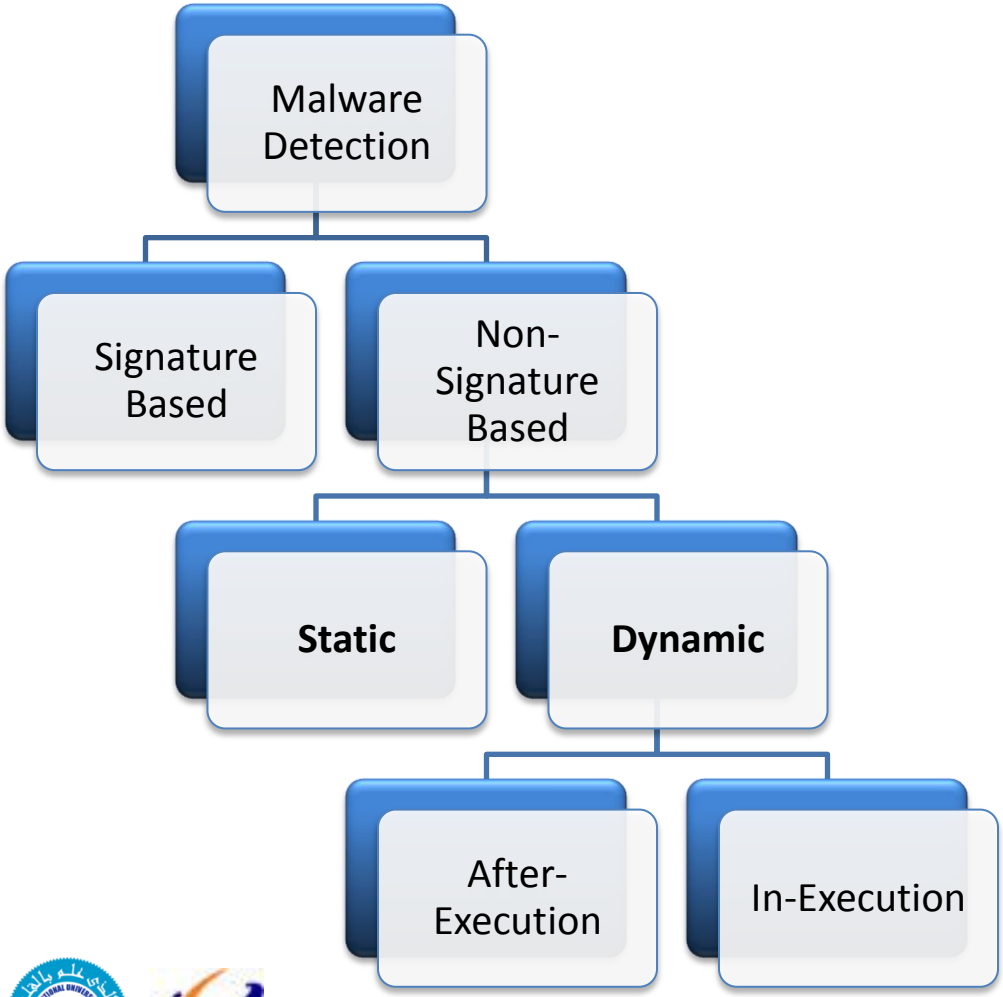
Non-Signature Based:

· Detection on the basis of smarter features

· Able to detect new malware

· Regular updates may not be necessary

Types of Malware Detection



Static Detection:

· Detection on the basis of file as residing on secondary storage

· Prone to techniques such as obfuscation and polymorphism.

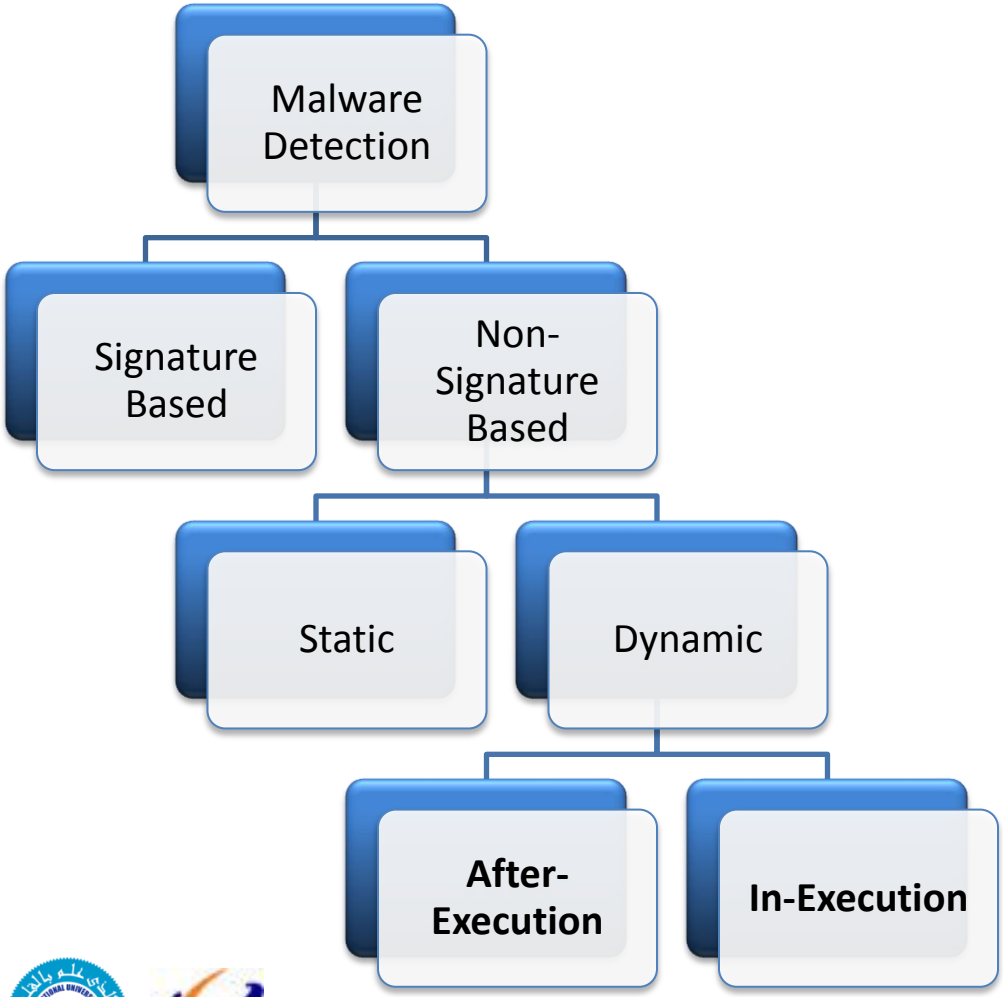
Dynamic Detection:

· Its detects malware on the basis of run-time behavior of a process (It's a more direct look)

· Resilient to techniques such as code obfuscation

· High processing overhead

Types of Malware Detection



After-Execution Detection:

· Forensic and taint Analysis

· Offline analysis

· Lower processing overhead

· Undo the Change made by it

In-Execution Detection:

- Detection during execution
- End user tool
- High processing overhead

Dynamic Analysis Techniques (on Android)

- Information Flow Tracking
- Dynamic Function Call Tracing
- Runtime Permissions Leak Detection
- Misbehavior analysis using power utilization patterns
- System Performance/Behavior based Anomaly Detection

Our Proposed Thesis

To use parameters of Process Control Blocks in kernel of OS for malware detection

- *Task_struct is Process control block in Android kernel*

Every executing process has an associated Task Structure in the kernel

- Task_struct is structure of multiple structures (along with it own native parameters)
- e.g. Memory Manager, Signal Structure, File system etc.

The parameter values of the TS fields are different for benign and malicious processes on the basis of their execution behavior

- So a collection of parameters can be use to discriminate between benign and malware processes

Multi-Threaded applications launched from user space have a common task structure for all threads.

Problem Statement

To Identify a malicious process based upon its execution patterns in its process control blocks (using machine learning algorithms) and stop it before it can complete its execution

Requirements for our system

- High detection accuracy
- Low false alarm rate
- Low processing overhead
- Real-time classification
- Detect malware during execution
- Robust to evasion attempts

Datasets (Enhanced Results)

Enhanced version of DroidKnight is presented as TStructDroid [Tech Rep]

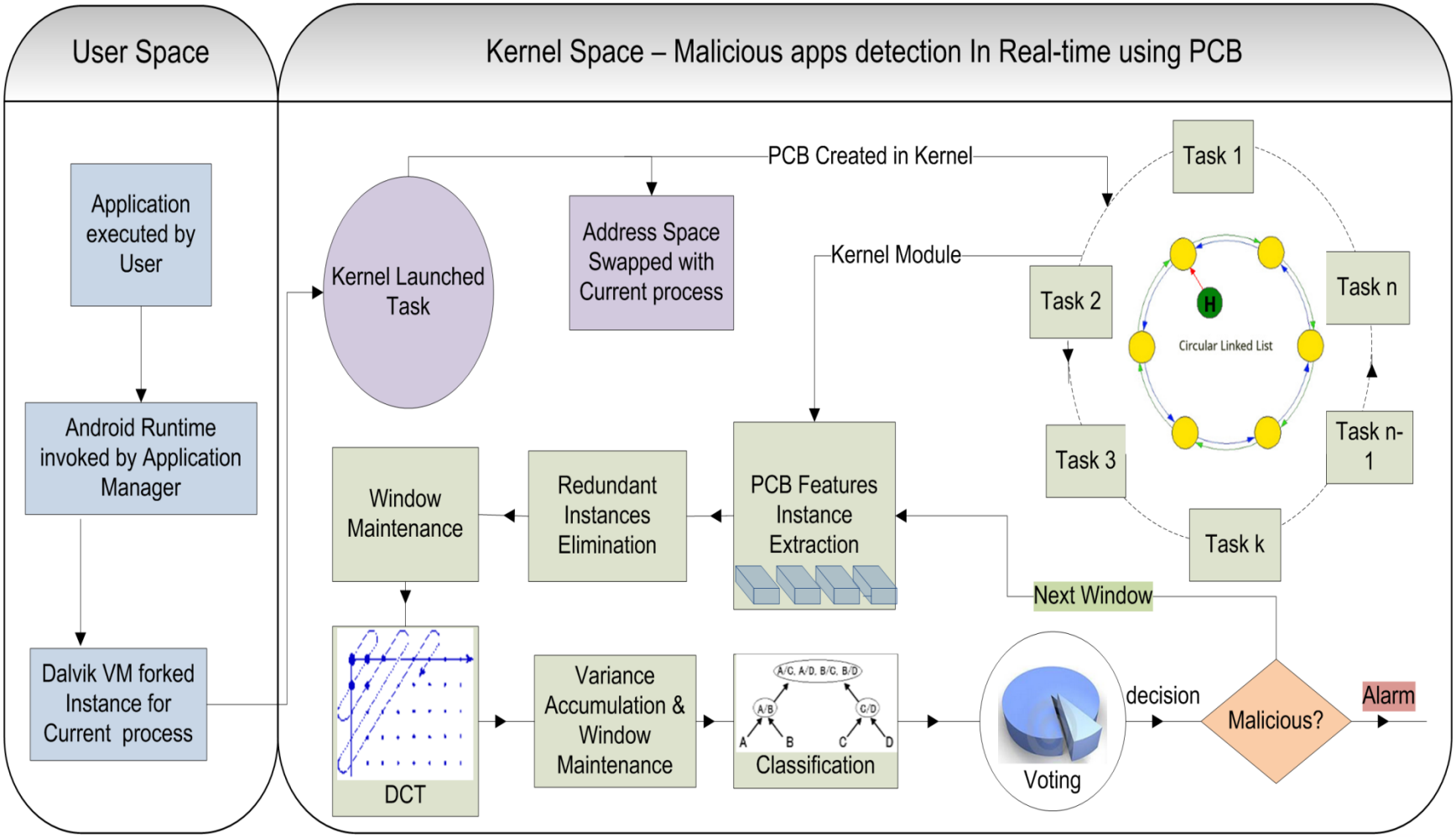
Malware Categories (110 Malware Samples)

- Trojans, SpyWare, BackDoor, Bots, RootKits

Benign Categories (110 benign sample)

- Games, Image Viewers, Sketching tools, Text Editors, Image Editors, Android Utilities

Process Flow Diagram



Architecture of Framework

Feature Extractor

- Java Apps Execution
- Kernel Module for PCB dump

Features Analyzer

- Time-series Features Short-listing

$$p_{f_i} = \begin{cases} 0 & \text{if } f_i \in \mathcal{F}_{indexers} \\ 0 & \text{if } f_i \in \mathcal{F}_{constant} \\ 0 & \text{if } f_i \in \mathcal{F}_{null} \\ 0 & \text{if } f_i \in \mathcal{F}_{ident-dist} \\ 1 & \text{Otherwise} \end{cases}$$

…contd

Redundant feature-instance elimination

Time-series Segmentation & Frequency Information Extraction

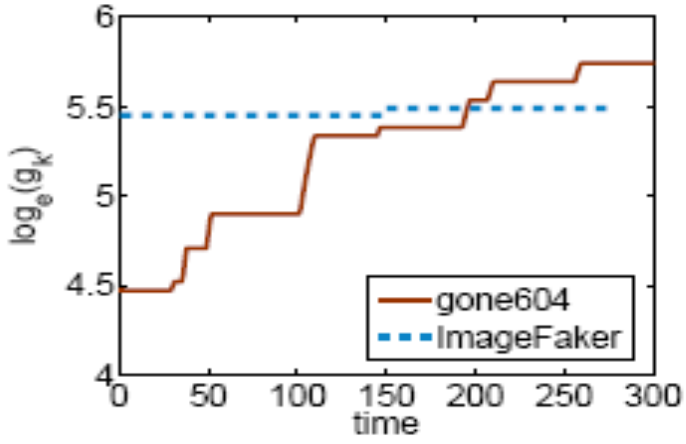
Variance Calculation over time-series Segments

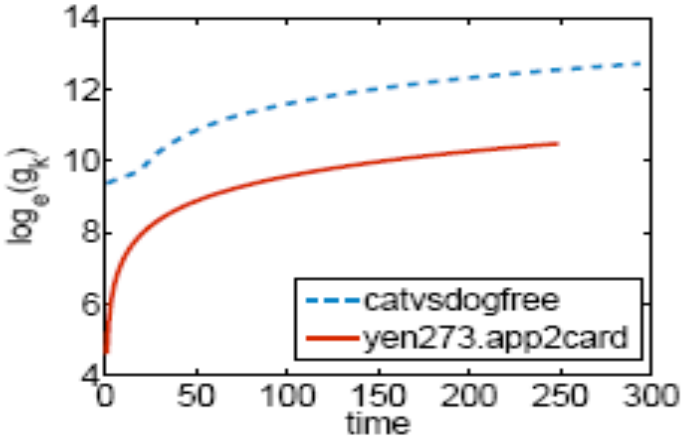
Variance Accumulation for Time-series Segments

Lemma 1. *Let g_k be the cumulative variance on window k of size S where $1 \leq k \leq K$. k is defined as $K = T/S$, then:*

$$g_k = \sum_{k=1}^{K} (\mathrm{E}[X_k^2] - \mathrm{E}^2[X_k])$$

…contd



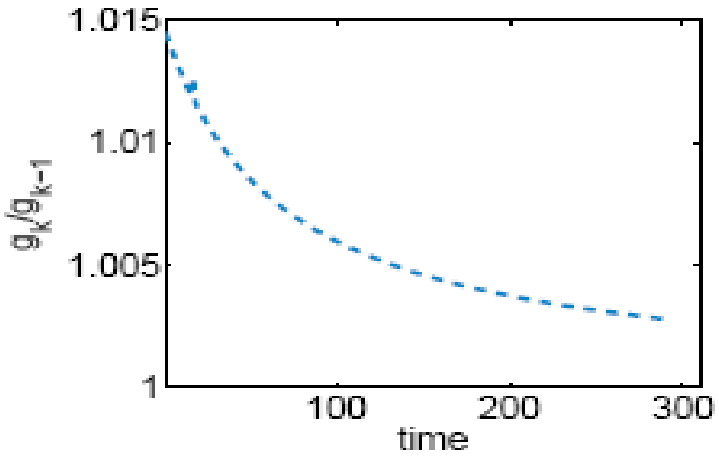


Theorem 1. *The function* $\frac{g_k}{g_{k-1}}$ *is given by*

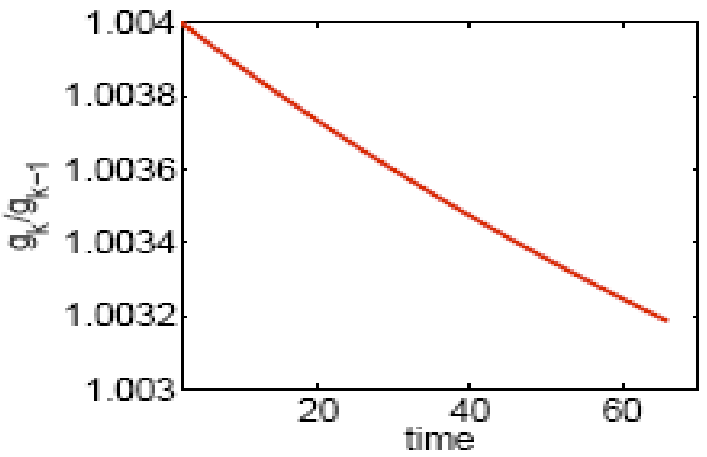
$$\frac{g_k}{g_{k-1}} = 1 + \frac{\sigma^2(X_k)}{g_{k-1}} \tag{5}$$

and it is a bounded function and converges to 1, as time $T \rightarrow \infty$

...contd

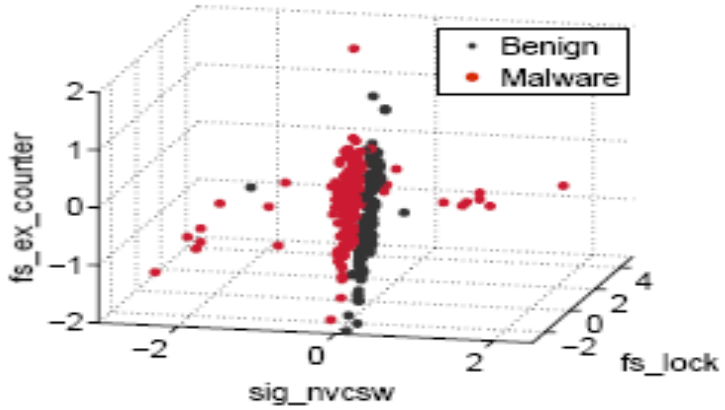


(a) Benign Processes

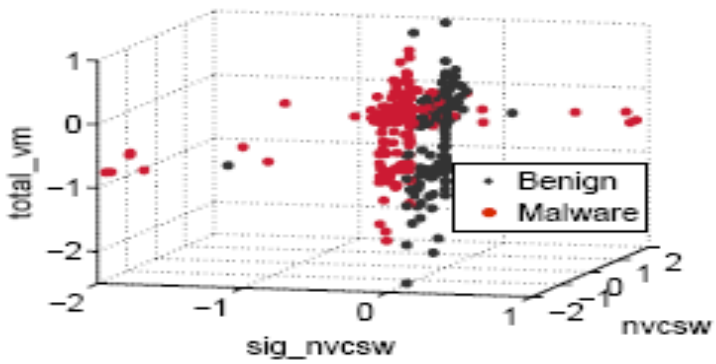


(b) Malware Processes

Theorem 2. *The presented model – variance accumulation of frequency components obtained using DCT, over the time varying windows of PCB parameters of benign and malicious processes – is linear and stable.*



(a) One set of three coefficients



(b) Another set of three coefficients

Scenario 1: RealTime Malware Detection

- (Dataset Composition 1x220 folds)

Scenario 2: Standard - Cross Validation Scenario

- (Dataset Composition 10x22 folds)

Classification Engine

- Information gain (refer TR)
- Machine Learning Classifier – Decision Tree J48

Voting Method & Alarm

Results

Time Resolution (δt ms)	Segment Size (T instances)	Voting Window Size (W_{vote} segments)	Real-Time Scenario (%)			Cross-validation Scenario (%)		
			DR	FAR	DA	DR	FAR	DA
10	5	30	99.09	0.91	99.1	90	5.45	92.27
10	10	30	99.09	0.91	99.1	90	5.45	92.27
10	20	30	99.09	0.901	99.55	90.91	5.45	92.73
10	40	30	98.18	0.901	98.64	93.64	7.27	93.18
20	5	30	99.09	0.909	99.1	90	5.45	92.27
20	10	30	99.09	0.909	99.1	90.91	5.45	92.73
20	20	30	99.09	0.909	99.1	90.91	5.45	92.73
20	40	30	98.18	0.909	98.64	90	5.45	92.27
40	5	30	99.09	0.909	99.1	90	5.45	92.27
40	10	30	99.09	0.909	99.1	90.91	5.45	92.73
40	20	30	99.09	0.91	99.55	90.91	5.45	92.73
40	40	30	100	0.91	99.55	93.64	6.36	93.64

Performance Overhead

Processing Overhead (1.685%)

- Feature Extraction + Classification

Android - Performance Degradation (3.73%)

Application	Baseline Exec. Time (s)	Exec. Time with TstructDroid (s)	Overhead (%)
Zip (Archive)	127.2	131.5	3.27
Zip (Best Compress)	67.86	69.94	2.97
Zip (Best Speed)	50.39	52.01	3.11
Zip (Deflated)	132.12	141.81	6.83
Zip (Filtered)	147.59	154.27	4.33
File Copy	190.6	196.95	3.22
File Search	131.68	134.91	2.4
Average			3.73

Conclusion

A realtime analysis of time-series PCB dataset is presented to detect Malware.

A Novel scheme is used to extract hidden information in the execution patterns of the processes

110 Malware and 110 benign application are used to test the framework.

99% DR with less than 1% Fp is achieved with realtime malware detection scenario

A 90-93% Dr is achieved with 5-6% FP using standard 10 fold cross validation scenario

Finally, 3-4% performance degradation of System is observed with framework

All these benchmark make is this scheme suitable for realtime deployment.

References

Gartner, “Gartner says asia/pacific led worldwide mobile phone sales to growth in first quarter of 2013,” http://www.gartner.com/newsroom/id/2482816 [last-viewed-June-3-2013], 2013.

Trend-Micro, “3q 2012 security roundup: Android under siege: Popularity comes at a price,” Research and Analysis, 2012.

S. HILL, “Top 3 android security apps, do they protect you?” http://www.digitaltrends.com/mobile/top-android-security-apps/(lastviewed- on-December-10-2012), 2012.

D. Ehringer, “The dalvik virtual machine architecture,” Techn. report (March 2010), 2010.

I. Witten and E. Frank, Data mining: Practical machine learning tools and techniques, second edition. Morgan Kaufmann, 2005.

G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, “Madam: a multi-level anomaly detector for android malware,” in International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, 2012, pp. 240–253.

References

T. Blasing, L. Batyuk, A. Schmidt, S. Camtepe, and S. Albayrak, “An android application sandbox system for suspicious software detection,” in International Conference on Malicious and Unwanted Software (MALWARE). IEEE, 2010, pp. 55–62.

G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, “Paranoid android: Versatile protection for smartphones,” in Proceedings of the 26th Annual Computer Security Applications Conference. ACM, 2010, pp. 347–356.

I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, “Crowdroid: behavior-based malware detection system for android,” in Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011, pp. 15–26.

A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, “Andromaly: a behavioral malware detection framework for android devices,” Journal of Intelligent Information Systems, pp. 1–30, 2011.

Farrukh Shahzad, M. Ali Akbar, Salman H, Khan, Muddassar Farooq, “TStructDroid: Realtime Malware Detection using In-execution Dynamic Analysis of Kernel Process Control Blocks on Android”, Technical Report [http://nexginrc.org/nexginrcAdmin/PublicationsFiles/TR-TStructDroid-2013-02.pdf]

Thank you. Any questions?