ARTICLE IN PRESS

+ MODEL

HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Karbala International Journal of Modern Science xx (2017) 1–7

http://www.journals.elsevier.com/karbala-international-journal-of-modern-science/

Designing a smartphone honeypot system using performance counters

Hanaa Mohsin Ahmed*, Nidaa Flaih Hassan, Assmaa A. Fahad

Computer Science Department, University of Technology, Baghdad, Iraq

Received 19 October 2016; revised 3 January 2017; accepted 13 February 2017

Abstract

This paper presents a design for a honeypot smartphone system. The smartphone honeypot system has to perform several complex functions, the basic important three functions are: design and construct the system database, malware detection, and system reactions. During the construction of the system database various information, about the behaviour of various well known malicious applications is captured and saved in a database files using the hardware performance counters. Three features are used for this purpose: number of instructions, number of branches, and number of cache misses. A data set with 1260 malicious program is used to collect these features. One-dimensional Euclidian distance and multi-dimensional Euclidian distance are used to classify the samples from the data set to identify the family they belong to. Although the classification results were as low for some families, the algorithm is fully classified other families with 100% accuracy. The results indicate that the performance counters are good tools for detecting malware.

© 2017 The Authors. Production and hosting by Elsevier B.V. on behalf of University of Kerbala. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Keywords: Smartphone; Honeypot; Design; K-means

1. Introduction

The wide range of services that smartphones can provide users today, such as connecting via social networks, regularly checking email, surfing the internet, and taking pictures, make these devices the most valuable computers. Their massive usage has led to significant growth in the number and complexity of malicious applications targeting them. Security information tools, such as firewalls, intrusion detection systems, and anti-virus applications, are gaining popularity. However, they generally detect well-known malicious application families and protect against vulnerabilities. The dynamic evolving nature of these attacks makes zero-day attacks difficult to detect for some time. Thus, there is a need for new techniques to recognise malware attacks and deceive them. The honeypot technique is expected to achieve these requirements. Several factors go into Honeypot design process. Good Honeypot design requires daunting, and necessary skills that make Honeypot design a challenge for the most experienced software architects.

* Corresponding author.

E-mail addresses: salmanhanna2007@yahoo.com (H.M. Ahmed), nidaaalalousi_5@yahoo.com (N.F. Hassan), assmaa_fahad@yahoo.com (A.A. Fahad).

Peer review under responsibility of University of Kerbala.

http://dx.doi.org/10.1016/j.kijoms.2017.02.004

2405-609X/© 2017 The Authors. Production and hosting by Elsevier B.V. on behalf of University of Kerbala. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Please cite this article in press as: H.M. Ahmed et al., Designing a smartphone honeypot system using performance counters, Karbala International Journal of Modern Science (2017), http://dx.doi.org/10.1016/j.kijoms.2017.02.004

Therefore, this problem must be divided into its smallest parts to start in the right direction.

Based on previous works, see ([1], and [2]), and the designed honeypot in this paper, the following functions are deduced as the main functions that the honeypot must perform. The main function is to design and construct the system database that can be used by the K-means classification algorithm to identify the family to which the malicious program belongs. After detecting the malware program, the designed honeypot must react in some manner or technique, such as shutting down the system or simply reporting the threat to the user.

The honeypot system can analyse data using static techniques or dynamic techniques. Static analysis is a quick, inexpensive approach to find malicious characteristics or bad code segments in an application without executing it [7]. In contrast to static analysis, where the binary to be analysed is disassembled or debugged, the actual code of the file is completely ignored under dynamic analysis. The binary to be analysed is seen as a black box and is executed in a controlled environment, such as a virtual machine or emulator, so that the researcher can monitor the application behaviour.

The idea of detecting malicious activities is based on examining the variations within a family of malware on a device platform. Modern microprocessors produce different components for monitoring operations. For example, advanced reduced instruction set computing (RISC) machine (ARM) microprocessors provide hardware performance counters, registers provided by the majority of modern microprocessors, which can be used to capture a wide range of hardware-related activities in the system. Each of these activities represents a program feature [3]. The data collected from these performance counters can be used to analyse the behaviour of the malware program.

Demme et al. (2014), study the feasibility of using the performance counters to build a detector in hardware. They proposed a hardware modifications, that allow the detector to run securely and less buggy than the software detectors. They provide a hardware Antivirus that use external performance counters which can be used in combination with the permissions requested at the installation time of the application program and they reached an 83.1% classification result [4].

2. General system design

In this paper, the Android smartphone operating system (OS) is chosen due to its popularity among mobile users and the extremely high ratio of malware targeting it. The practical parts and designed algorithms are implemented using a mini tablet, the Nexus 7. This mini tablet works with Android version 4.3.1 under Linux kernel version 3.1.10-g30c5396. In order to work with Nexus 7, the device must be unlocked and a new version of Linux OS must be downloaded, compiled, and flashed to this device. There are two phases to the development of the proposed smartphone honeypot: a training phase and testing phase. In the training phase, the training data are analysed by a classification algorithm to generate classification rules. In the testing phase, the test data are used to estimate the accuracy of the classification rules.

The designed honeypot performs the following functions: Design and construct the system database, Data analysis, and System reactions. The first step in constructing the system's database is to download and collect malware application programs from the application market. This paper uses the malware programs supported by Android Malware Genome Project [8]. These malware samples are categorizes 1260 malicious application, presented in 49 different malware families, covering the majority of Android malware existing, which ranging from their debut in August 2010 to October 2011.

3. Manipulating the performance counters

Performance counters are event-orientation observation tools which can be used to stamp different types of events that covering the kernel, the processor, and the application behaviour. Before programming the performance counters and reading its data, it is needed to fix the CPU clock frequency to ensure the correctness of the collected data. The modern CPUs have a feature which let the system/user to scale their operating frequency according to the system/user needs. This function can be achieved by changing a voltage power supply input. With this feature the processor can reduces the overall power consumption, lowering the temperatures and fan speed when the entire processor resources are not needed. This changing in the CPU frequency in terms of work load, will effects on the correctness of the performance counters counting operation. Therefore the CPU frequency needs to fix at the beginning of the data collection function.

4. Features selection

In machine-learning applications, a large number of redundant or irrelevant features present a number of

Please cite this article in press as: H.M. Ahmed et al., Designing a smartphone honeypot system using performance counters, Karbala International Journal of Modern Science (2017), http://dx.doi.org/10.1016/j.kijoms.2017.02.004

problems, such as increasing the model complexity and runtime, misleading the learning algorithm, and overfitting. These can be more adverse effects and more crucial when implemented on mobile devices because of the restricted processing and storage capability as well as the battery life.

Applying fine features selected from a large number of available features enables more efficient use of the malware detector and a faster detection time. In this paper, the performance counters are programmed to monitor the application program for four events: the number of instructions, number of cache misses number of branches, and number of cycles.

Unlike the other features, the cycles feature does not produce good results in recognising the malware programs. This feature is excluded from the system because of its weak results.

Figs. 1 and 2 show the four features of the BaseBridge and Zsone malware families, respectively. The *x*-axis represents the samples within the family, and the *y*-axis represents the values of the features for these samples. As noted, the values of the learning-machine cycles feature diverge compared with the values of the other features (branch feature, instruction feature, and cache miss feature). Even when used with the classifier algorithm, the cycles feature produces fuzzy results that cannot be trusted to classify the application programs and distinguish the malware programs. Therefore, this feature is excluded from the system, and the classification algorithm continued with only three features rather than four.

5. Features reading and extracting

In features reading, the Android application program must be executed for a certain time period and the program must repeat different actions for different input data so that the program will take different routes and the system can capture additional and different features. Within this time, the system keeps a running count. The occurrences of events are aggregated and presented at the end of the time period. When this time period elapses, the system reads the features of the program by reading the contents of the performance counters.

The features reading algorithm generates massive data that may take a long time to process. The feature extracting function will produce a more concise data file that includes the feature vectors. Such a parsing operation is necessary to speed up the machine-learning algorithm used in the data analysis phase.

6. Data analysis function

K-means classification algorithm is used to estimate the potential for malware identification. A classification

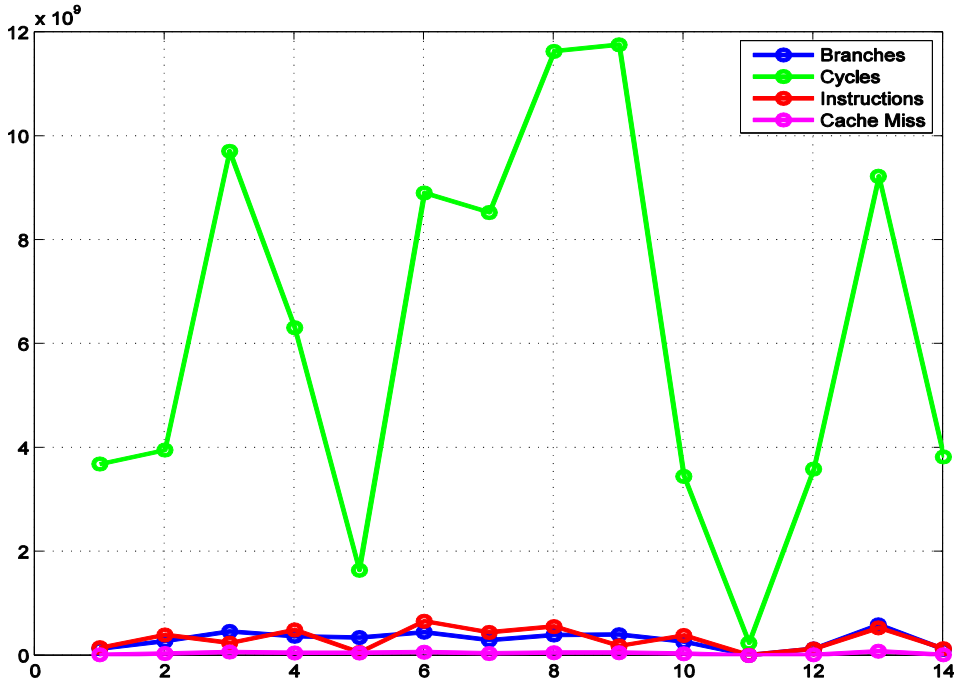

Fig. 1. BaseBridge malware family features.

Please cite this article in press as: H.M. Ahmed et al., Designing a smartphone honeypot system using performance counters, Karbala International Journal of Modern Science (2017), http://dx.doi.org/10.1016/j.kijoms.2017.02.004

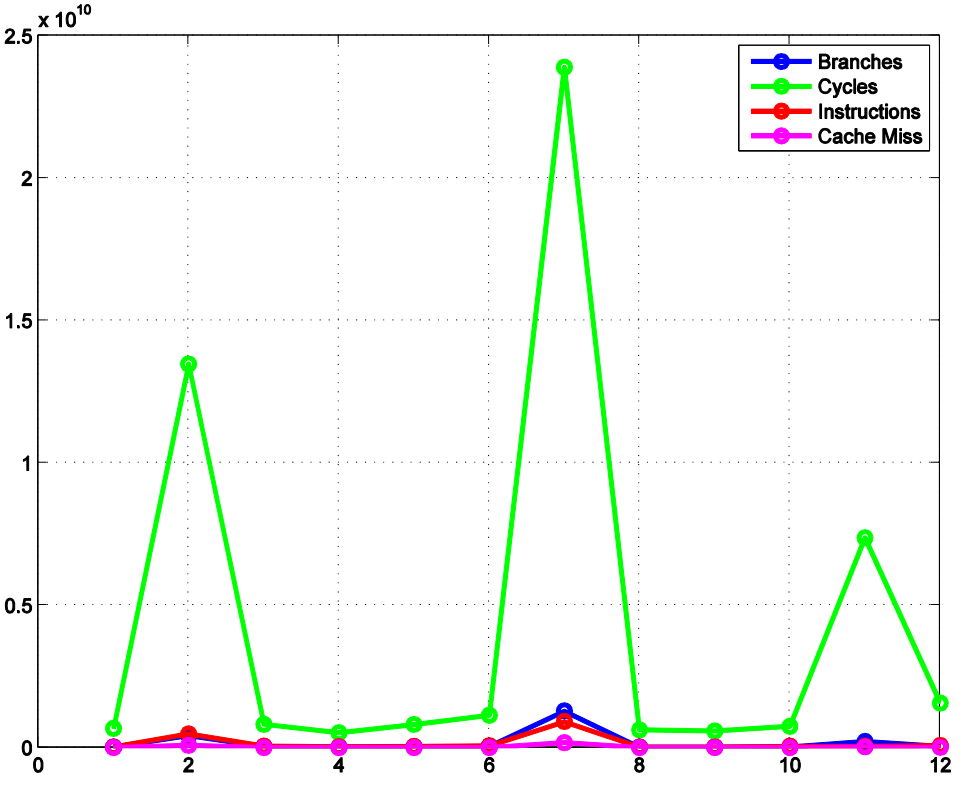

Fig. 2. Zsone malware family features.

scheme is, at best, as good as the discerning power of its features. The current performance counters offer a good number of features that can be used to produce good classification of malware.

Similarity and dissimilarity between objects is often expressed in terms of a distance measure. Euclidean distance is used to compare the features of the test-case program with the features stored on the malware database files. In this work, each feature is considered a dimension. Each malware program that is included in the database, in both the training dataset and the testing dataset, is represented as a point with three dimensions: the number of instructions, number of branches, and number of cache misses. Accordingly, the multi-dimensional Euclidean distance is calculated using Equation (1) [6]:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}, \quad (1)$$

where

p_1 is the number of instructions for the malware program in the database system;
q_1 is the number of instructions for the test-case program;
p_2 is the number of branches for the malware program in the database system;
q_2 is the number of branches for the test-case program;
p_3 is the number of cache miss for the malware program in the database system; and
q_3 is the number of cache misses for the test-case program.

7. Data pre-processing function

Two normalisation techniques are tested in this dissertation: min−max and Z-score (standard deviation) [5]. These techniques are used to normalise the database files generated by the data collection stage. Accordingly the system database consists of three different datasets:

1. A dataset collected from the ARM performance counters and saved in database files without pre-processing.

Please cite this article in press as: H.M. Ahmed et al., Designing a smartphone honeypot system using performance counters, Karbala International Journal of Modern Science (2017), http://dx.doi.org/10.1016/j.kijoms.2017.02.004

2. A dataset generated by applying the min–max normalisation rule on a copy of database files.
3. A dataset generated by applying the Z-score normalisation rule on a copy of database files.

8. Application program pattern matching

Pattern matching is a technique used to determine to which class each testing pattern belongs. The K-means algorithm is used to classify the Android application program using the features collected by the data collection function. The idea in this paper is that, regardless of how malware writers change their software, the semantics of the program does not change. In addition, in accomplishing a particular task, subtasks exist; however, the main task in the program cannot be radically modified. Therefore, the behaviour of the execution of the malware remains unchanged and invariant among variations [7]. The multi-dimensional Euclidean distance classification algorithm is implemented on the three available datasets: the dataset without pre-processing, the dataset with the min–max pre-processing algorithm, and the dataset with the Z-score per-processing algorithm. Tables 1–3 show the results of using these datasets, respectively.

The tables emphasise that using the pre-processing steps did not produce significant results compared to not using them. However, this comes with the resource costs of a smartphone, namely, a limited resource platform. A smartphone cannot afford running these steps, particularly when it does not introduce great improvement.

Additionally, the selected features – the number of cache misses, number of branches, and number of instructions – are common features not specifically attached to a particular application type, such as malware programs. Using more application-related features may produce more accurate results.

Table 1
Experimental results of implementing multi-dimensional Euclidean distance algorithm on the dataset without pre-processing.

Family name	Total number of APKs	Number of training pattern	Number of testing pattern	Results of the classifier
Zsone	12	8	4	4/4
Geinimi	20	13	7	3/7
BaseBridge	14	9	5	2/5
DroidKungFu4	6	4	2	1/5
DroidDeluxe	1	1	1	1/1
Asroot	2	1	1	0/1
ADRD	4	3	1	0/1

Table 2
Experimental results of implementing multi-dimensional Euclidean distance algorithm on dataset with min–max pre-processing.

Family name	Total number of APKs	Number of training pattern	Number of testing pattern	Results of the classifier
Zsone	12	8	4	3/4
Geinimi	20	13	7	1/7
BaseBridge	14	9	5	0/5
DroidKungFu4	6	4	2	0/2
DroidDeluxe	1	1	1	1/1
Asroot	2	1	1	0/1
ADRD	4	3	1	1/1

Table 3
Experimental results of implementing multi-dimensional Euclidean distance algorithm on dataset with Z-score pre-processing.

Family name	Total number of APKs	Number of training pattern	Number of testing pattern	Results of the classifier
Zsone	12	8	4	0/4
Geinimi	20	13	7	1/7
BaseBridge	14	9	5	2/5
DroidKungFu4	6	4	2	0/2
DroidDeluxe	1	1	1	0/1
Asroot	2	1	1	0/1
ADRD	4	3	1	1/1

9. System reaction

The final function implemented by the honeypot smartphone system is to accomplish a security policy when a malicious program attempts to infect a protected system. A wide spectrum of action may be taken

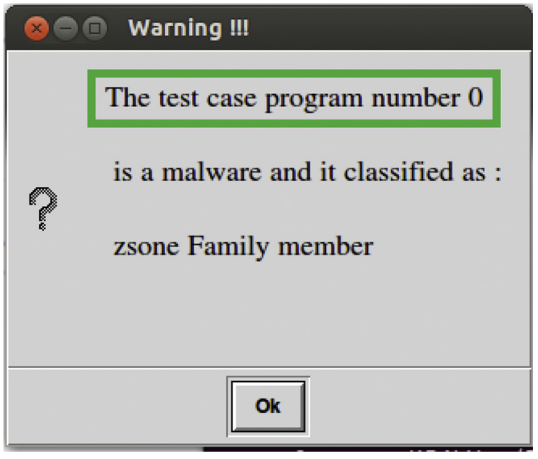

Fig. 3. One-dimensional Euclidean distance classification algorithm warning report.

Please cite this article in press as: H.M. Ahmed et al., Designing a smartphone honeypot system using performance counters, Karbala International Journal of Modern Science (2017), http://dx.doi.org/10.1016/j.kijoms.2017.02.004

Table 4
Comparison between the designed honeypot system with the previous work.

Related work	Using honeypot tools	Data analysis techniques		Implementation		Running operating system	
		Hardware	Software	Emulator	Real device	Android	Windows
[9]	✓		✓	✓			✓
[10]			✓		✓	✓	
[11]	✓		✓		✓	✓	
[12]			✓		✓	✓	
[13]			✓	✓		✓	
[14]	✓		✓	✓		✓	
[15]			✓		✓	✓	
[16]			✓		✓	✓	
[4]		✓ External			✓	✓	
The suggested smart phone		✓ Provided on mobile device			✓	✓	

based on the security system output. Some of these are the following:

1. Logging communication and interaction between the suspicious process and other processes.
2. Running more sophisticated behavioural analysis on a suspicious process.
3. Forcing the shutdown of the suspicious process.
4. Displaying a warning report to the user.

Except for the last system reaction, all of the abovementioned actions come with high resource consumption in terms of CPU and storage, which are already scarce in a smartphone device. Furthermore, these mechanisms require the security system to work at the highest privilege level, which must be supported by the Android OS.

In this paper, the designed security system is expected to display a warning report to the user. This report contains a warning message and the family name to which the test-case application program belongs. Eventually, the user may decide to kill the suspicious process or to allow the process continue if the user trusts this application program. Fig. 3 shows an example of the warning report displayed by the multi-dimensional Euclidean distance classification algorithm.

10. Comparison with the previous work

Finally and as a comparison with the previous work, Table 4 shows that some researchers, such as Freeman [9], Wahlisch [11], and Wahlisch [14], used a prebuilt tools to build their Honeypot. While others researchers used software data analysis techniques. Honeypot suggested in this dissertation as well as Demme [4], both use hardware data analysis tools. Demme use an external hardware to collect the data from the performance counters. In this dissertation the hardware provided on a mobile device is used.

11. Conclusions

From the experiments performed on using the data collected from the performance counters to build a database system for the designed smartphone honeypot, and the classification algorithms used to test the feasibility of using this database system to identify the malware programs, four conclusions are drawn:

1. Designing a smartphone honeypot comes with computational overhead that may affect the device response and battery life. Very restricted smartphone resources mean a low-interaction honeypot is recommended to design a smartphone honeypot system.
2. The honeypot could help to generate real statistics about the attack behaviour that can be shared with other users or the system security to prevent the spread of the malware.
3. With features of the performance counters, a simple classification algorithm can be used to produce acceptable classification results.
4. Although the K-means classification results were as low as 33% for some families, the algorithm fully classified other families with 100% accuracy. This indicates that the malicious application programs can be detected using the hardware performance counters provided by modern microprocessors.

Please cite this article in press as: H.M. Ahmed et al., Designing a smartphone honeypot system using performance counters, Karbala International Journal of Modern Science (2017), http://dx.doi.org/10.1016/j.kijoms.2017.02.004

References

[1] M.A. Hanaa, F.H. Nidaa, A.F. Assmaa, A survey on smartphone honeypot, Int. J. Comput. Technol. 11 (4) (Oct 2013) 2476–2480.

[2] A.F. Assmaa, M.H. Hanaa, F.H. Nidaa, Design requirements of a smart phone honeypot system, Int. J. Comput. Appl. 6 (4) (Dec 2014) 97–104.

[3] ARM-DDI-0388I, Cortex-A9 Technical Reference Manual, ARM, 2012.

[4] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, Sethumadhavan, S. Salvatore, On the feasibility of online malware detection with performance counters, in: Proceeding 40th Annual International Symposium on Computer Architecture, ACM, New York, USA, 2013, pp. 559–570.

[5] I. Witten, E. Frank, M. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, Elsevier, 2011.

[6] A.A. Inas, H. Soukaena, A Model to Detect Denial of Service Using Data Mining Classification Algorithms, Iraqi Commission for Computers and Informatics, Baghdad, Iraq, 2013.

[7] T. Isohara, K. Takemori, A. Kubota, Kernel-based behavior analysis for android malware detection, in: IEEE 7th International Conference on Computational Intelligence and Security, IEEE, Hainan, 2011, pp. 1011–1015.

[8] Y. Zhou, X. Jiang, Android Malware Genome Project, 2014. Retrieved 01 20, 2015, from Dataset Release Policy: http://www.malgenomeproject.org/policy.html.

[9] Michael Freeman and Andrew Woodward, SmartPot - creating a 1st generation smartphone honeypot, in: Proceedings of the 7th Australian Digital Forensics Conference, 1–3 December 2009.

[10] C. Mulliner, S. Liebergeld, M. Lange, Poster: HoneyDroid - creating a smartphone honeypot, in: Poster at IEEE Security & Privacy, May 2011.

[11] M. Wählisch, S. Trapp, C. Keil, J. Schönfelder, T. C.Schmidt, J. Schiller, First insights from a mobile honeypot, in: Proc. of ACM SIGCOMM, Poster Session, ACM, New York, August 2012, pp. 305–306.

[12] Matthias Wählisch, Sebastian Trapp, Jochen Schiller, Benjamin Jochheim, Theodor Nolte, Thomas C. Schmidt, Osman Ugus, Dirk Westhoff, Martin Kutscher, Matthias Küster, Christian Keil, Jochen Schönfelder, Vitamin C for your smartphone: the SKIMS approach for cooperative and lightweight security at mobiles, in: SIGCOMM'12, Finland, Helsinki, August 13–17, 2012.

[13] Yubo Song, Xiaoyun Zhu, Yelin Hong, Haoyue Zhang, Hangbo Tan, A mobile communication honeypot observing system, in: Fourth International Conference on Multimedia Information Networking and Security, IEEE, November 2012.

[14] Matthias Wählisch, André Vorbach, Christian Keil, Jochen Schönfelder, Thomas C. Schmidt, Jochen H. Schiller, Design, Implementation, and Operation of a Mobile Honeypot, arXiv:1301.7257v1, 30 Jan 2013.

[15] S. Liebergeld, M. Lange, C. Mulliner, Nomadic honeypots: anovel concept for smartphone honeypots, in: Proc. W'shop on Mobile Security Technologies (MoST'13), Together with 34th IEEE Symp. On Security and Privacy, May 2013 to appear.

[16] E. Gelenbe, G. Gorbil, D. Tzovaras, S. Liebergeld, D. Garcia, M. Baltatu, G. Lyberopoulos, Security for Smart Mobile Networks: the NEMESYS Approach, arXiv:1307.0687v1, 2 Jul 2013.

Please cite this article in press as: H.M. Ahmed et al., Designing a smartphone honeypot system using performance counters, Karbala International Journal of Modern Science (2017), http://dx.doi.org/10.1016/j.kijoms.2017.02.004