Workshop Proceedings of

ATIS 2012

Melbourne, November 7th, 2012.

Third Applications and Techniques in Information Security Workshop

Edited by:
Matthew Warren
ISBN 978-0-9872298-2-3

Proceedings of

ATIS 2012

Edited by

Matthew Warren
ISBN 978-0-9872298-2-3

Published by the School of Information Systems, Deakin University, Melbourne, Victoria, 3125, Australia.

All formal papers published in the conference proceedings have been blind refereed by at least two of the ATIS 2012 **Organising** committee.

Welcome

The ATIS 2012 workshop is the third workshop of its series and is being held at Deakin University, Burwood Campus, Melbourne, Australia. This workshop looks at the continued development of Information Security and related technologies taking into account the issues that impact everyone in a global context.

Members of the workshop organising committee accepted each paper in the proceedings after a careful review; this took the form of a **blind review** by at least **two** members of the workshop organising committee. The papers were subsequently reviewed and developed where appropriate; taking into accounts the comments of the reviewers. The proceedings also includes a number of poster / short papers. These were submitted as non-refereed items and were discussed by the participants at the workshop itself.

The aim of this conference is to further the work already achieved within Australia and bring together researchers in the field to discuss the latest security issues.

We commend the authors for their hard work and sharing their results, and the reviewers of the workshop for producing an excellent program.

ATIS 2012 Organising Committee

- Associate Professor Jemal Abawajy, Deakin University, Australia.
- Dr Leijla Batina, Radboud University, The Netherlands and KU Leuven, Belgium.
- Professor Lynn Batten, Deakin University, Australia.
- Associate Professor Gleb Beliakov, Deakin University, Australia.
- Dr Morshed Choudhury, Deakin University, Australia.
- Dr Bernard Colbert, Deakin University, Australia.
- Dr Honghua Dai, Deakin University, Australia.
- Dr Robin Doss, Deakin University, Australia.
- Dr Rafiq Islam, Charles Sturt University, Australia.
- Dr Andrei Kelarev, Ballarat University, Australia.
- Dr Gang Li, Deakin University, Australia.
- Dr Vicky Mak, Deakin University, Australia.
- Dr Wenjia Niu, Institute of Acoustics, Chinese Academy of Sciences, Beijing, China.
- Dr Lei Pan, Deakin University, Australia.
- Dr Udaya Parampalli, Melbourne University, Australia.
- Professor Rei Safavi-Naini, University of Calgary, Canada.
- Dr Wolfgang Schott, IBM, Switzerland.
- Dr Steve Versteeg, CA Labs, Australia.
- Associate Professor Jinlong Wang, Qingdao Technological University, Qingdao, China.
- Dr Xiaofeng Wang, Wireless Sensor Network Laboratory, Institute of Computing Technology, Chinese Academy of Sciences.
- Professor Matthew Warren, Deakin University, Australia.
- Professor John Yearwood , Ballarat University, Australia.
- Associate Professor Xun Yi, Victoria University, Australia.

Contents

Paper 1

Zero Permission Android Applications - Attacks and Defenses.
Veelasha Moonsamy and Lynn Batten, Deakin University, Australia.

Zero Permission Android Applications - Attacks and Defenses

Veelasha Moonsamy, Lynn Batten
School of Information Technology
Deakin University
Melbourne, Australia
v.moonsamy@research.deakin.edu.au, lynn.batten@deakin.edu.au

Abstract

Google advertises the Android permission framework as one of the core security features present on its innovative and flexible mobile platform. The permissions are a means to control access to restricted APIs and system resources. However, there are Android applications which do not request permissions at all.

In this paper, we analyze the repercussions of installing an Android application that does not include any permission and the types of sensitive information that can be accessed by such an application. We found that even applications with no permissions are able to access sensitive information (such the device ID) and transmit it to third-parties.

Keywords: android, application, permission.

1. Introduction

Permission systems were introduced in the early days of computing when desktop computers were regarded as an emerging technology [18]. Traditionally, permissions were used to assign file access rights (for example: read, write) to users and also to regulate access to lower-levels of the operating system (for example as superuser). The operating system keeps a list – *Access Control List* (ACL) – in order to document the permission rights for each user who has access to the machine.

Google implemented a similar idea in its Android mobile operating system. Android includes a Linux kernel (and its libraries) which serves as its base operating system, a Dalvik virtual machine, an application middleware layer and lastly, a set of system applications. Each application is assigned its own user ID (UID) and is executed in individual sandboxes. Even though the applications' executions are separated, Android allows inter-application communication, provided that the correct levels of permissions have been assigned. Permissions are also required to access restricted system resources, such as the contact list. Whilst most applications do contain permissions, some might not necessarily make use of them; it depends on their functionalities.

This work investigates the consequences of installing an application that does not request any permission and determines the sensitive information such an application can have access to. The rest of the paper is organized as follows: In Section 2, we describe Android permissions in further detail and also present some relevant work in the Android permission area; we describe the potential attacks by a zero permission application on a device and possible defenses against such attacks in Section 3. In Section 4, we provide a discussion around zero permission applications and lastly, we give our conclusions and ideas for future work in Section 5.

2. Android Permissions

Android applications are available for download/purchase from Google's official market, Google Play [12], as well as from multiple third-party markets. The applications are available in .APK (application package file) format, which is essentially an optimized version of Java code. Each APK file includes the application's code (.dex files), multimedia resources, certificates, and the AndroidManifest.xml where the permissions are defined. The .xml file includes *<uses-permission>* and *<permission>* tags to allow developers to request permissions. The *<uses-permission>* tag is used if the developer needs to request any permission that has been predefined by Google. Currently, there are 130 official Android permissions running on Android 4.0. The other *<permission>* tag allows developers to define customized permissions in their applications.

Permissions can be classified into four types [11]: *Normal*, *Dangerous*, *Signature* and *SignatureorSystem*. Normal permissions do not require the user's approval but they can be viewed after the application has been installed. Dangerous permissions are displayed to the user and require confirmation in order to proceed with the installation process; these permissions have access to restricted resources and can have a negative impact if used incorrectly. Permissions classified under the

Signature category are granted automatically, provided that the requesting application is signed with the same certificate as the application that declared the permission. Finally, SignatureorSystem permissions are granted to those applications that have the same certificates as the Android system image.

Android adopts an 'install-time' permission granting policy [2]. In this case, once the application is downloaded the user will have to acknowledge the permission request in order to be able to use the application. Once the permissions are granted, they cannot be revoked unless the application is completely uninstalled. The authors in [1, 6, 15, 20] have developed methodologies that can provide users with more flexibility over the permissions once they are granted.

In [7], Enck et al. provided some insight into the Android security model by demonstrating the use of permissions during install-time and through inter-component communication. In their work, the authors explained that whilst application developers are allowed to control access to restricted resources by defining permissions in the AndroidManifest.xml file, they can also regulate inter-component communications. In regard to the latter, Enck et al. mentioned that developers can prevent other applications from accessing an application's components by either explicitly assigning permissions to those components or declaring the components to be private instead of public.

The work of Chaudhuri et al. [4] focused on designing semantics that can be used to formulate abstract representation of a particular application. The authors argued that their work can reveal the integrity of an application before the user installs it. This is useful in cases where third-party applications are required to interact with the components of those applications that come pre-installed on an Android phone.

The Android Permission framework allows developers to use Google's predefined set of permissions or generate their own, depending on the requirements of the applications being developed. Shin et al. [21] found a flaw in the customized permission scheme: since developers are not required to follow any naming conventions while defining their own permissions, the authors pointed out that this can introduce conflicted permissions. They implemented a legitimate banking application and a malicious application, both sharing a customized permission of the same name, to demonstrate how the privileges for the rogue application escalated by exploiting the vulnerability.

Felt et al. [8] and Bartel et al. [3] developed tools that can assist both users and developers to assess the integrity and reliability of applications before they are installed or uploaded on the market.

Nevertheless, it should be noted that if an application does not require access to any of the restricted system resources, the developer can choose not to include any permission in the AndroidManifest.xml file. In the next section, we will elaborate further on this particular case.

3. Potential Attacks and Defenses

In this section, we investigate the possible negative ramifications of installing an application with no permission by conducting a manual investigation of the application and also the defense mechanisms that are present in the literature to counter this issue.

3.1 Attacks

In one of his online posts, Brodeur [16] pointed out that an application that does not request any permission can scan the external storage directory and return a list of applications and files that are stored on the SDcard. This information can be accessed at /sdcard/. This vulnerability is rather predictable as Google grants read-only access to the SDcard on any device irrespective of the OS version that it is running on.

Additionally, a similar exploit can be carried out on the internal memory by scanning the /data/system/packages.list folder to retrieve the list of package names of those applications that are installed on the device. Furthermore, in the same folder, the file packages.xml stores the shared permissions of all applications on the device along with their corresponding UIDs.

The aforementioned vulnerabilities can be exploited by enticing the user to download a decoy application which can give the attacker access to a backdoor on the victim's device, as explained by Cannon in [22]. Cannon used an active remote shell to interact with the device and therefore had access to the data on the device. Furthermore, Brodeur demonstrated that the absence of INTERNET permission does not stop an attacker from exporting the captured data onto an external server. In fact, the URI_ACTION_VIEW Intent can initiate a Web browser call and thus allow the attacker to transfer data via the Internet. This is a common technique used by advertising libraries designed to leak device ID or subscriber ID in order to carry out targeted advertising, as observed in [14].

3.2 Analysis of the Brodeur Application

We downloaded and installed Brodeur's zero permission application on two versions of Android – 2.3 and 4.0. We set up an Android emulator, which is provided by the Android SDK and executed the application in question. At first glance, the installed

application behaves as any other clean applications. We then start a logcat filter, running in parallel with the emulator, to monitor the execution behaviors of the zero permission application.

The application itself includes three buttons – which can be customized depending on the nature of the attack. Each button carries out an attack and sends the information to an external party via the URI_ACTION_VIEW Intent, as explained in sub-section 3.1. When a user clicks on the first button, the application will read any information stored on the SD card and transfer it to the attacker. The second button allows the attacker to retrieve the application package names that are installed on the user's device. Finally, the third button returns two important unique identifiers, which are the device's SIM and GSM operator identifier.

Once the aforementioned information is gathered, an attacker can then carry out a targeted attack to exploit a particular user's device. Moreover, a zero permission application can be used to identify vulnerable Android users which will ensure the longevity of the attack, and afford a lower risk of being discovered or reported to the authorities.

3.3 Defenses

It is well-known [9] that the best defense against any security attack is achieved by raising user awareness. In the case of Android permissions, users need to be educated on how to interpret them. A study conducted by Felt et al. [9] showed Android users do not fully comprehend the permissions requests presented to them during install-time. Even more alarming is the fact that some users do not even take the trouble to read the permission descriptions and simply proceed to install the application. (This type of user behavior is very common when dealing with End-User License Agreements). Therefore, since smartphone users do not value the importance of permission requests, their absence is hardly noticed.

In addition to training, users may rely on metrics before proceeding to download and install applications. User ratings, reviews, number of downloads are some of the available resources that can help users to assess whether or not the application is popular or has caused other users any problems.

The solution proposed by Chin et al. [5] is also an effective way to quickly assess the integrity of an application before it is installed on the device. The authors developed a tool, ComDroid, that can detect potential vulnerabilities in Android applications. In order to do so, ComDroid first disassembles the APK file and then parses through the code to track the communication flow between components. However, it should be noted that this method is valid only for applications that are downloaded from third-party markets and not the ones from Google Play.

4. Discussion

In this section, we give our viewpoints on zero permission Android applications.

In its official documentation [13], Google states that *"a basic Android application has no permissions associated with it, meaning it cannot do anything that would adversely impact the user experience or any data on the device"*. For instance, an example of such an application could be a customized calculator application to calculate tips to be given for good service at an eatery. Based on the description provided by Google, the aforementioned application should not be able to access sensitive/restricted information on a device. However, as demonstrated in Section 3, the current security vulnerabilities within the Android platform can allow a zero permission application to access restricted information.

Furthermore, we believe that imposing rules stipulating that all applications must request at least one permission will not necessarily solve the problem. The literature shows that application that request unnecessary permissions cause far more damage than a zero permission application [17, 19]. Moreover, an over-privileged application can also compromise the functionalities of other applications stored on the device; the impact of this attack is far beyond the scope than the ones mentioned in Section 3.

Additionally, even though Google considers the Android permission system as one of the key security features of the platform, it should be noted that users' responses to understanding and approving permission requests have compromised the security standards of Android. The work by Felt et al. [10] demonstrates the lack of comprehension from Android users when it comes to understanding and interpreting permission requests upon installing an application. In fact, a majority of users do not even read the list of permissions and simply proceed to download and install the application. This shows that users regard the permission system as a hassle and therefore will not be concerned even if an application does not request any permission.

5. Future Work

In this paper, we have given an overview on Android permissions and analyzed and elaborated on the implications of zero permission applications. We also presented some measures that users can apply to identify vulnerable applications. In summary, user understanding

and user behavior are the key aspects that can mitigate the propagation of rogue applications.

As future work, we will provide a third-party service where Android users can perform a quick scan of applications that have been downloaded from Google Play and return the user a report, assessing the integrity of the applications before installation.

6. References

[1] H. Banuri, M. Alam, S. Khan, J. Manzoor, B. Ali, Y. Khan, M. Yaseen, M. Tahir, T. Ali, Q. Alam, and X. Zhang, "An android runtime security policy enforcement framework," *Personal and Ubiquitous Computing*, pp. 1–11, 2010, 10.1007/s00779-011-0437-6. [Online]. Available: http://dx.doi.org/10.1007/s00779-011-0437-6

[2] D. Barrera, J. Clark, D. McCarney, and P. van Oorschot, "Understanding and improving app installation security mechanisms through empirical analysis of android," in *Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)*. ACM, 2012.

[3] A. Bartel, J. Klein, M. Monperrus, and Y. Traon, "Automatically securing permission-based software by reducing the attack surface - an application to android," *Arxiv preprint arXiv:1206.5829*, 2012.

[4] A. Chaudhuri, "Language-based security on android," in *Proceedings of the ACM SIGPLAN Fourth Workshop on Programming Languages and Analysis for Security*, ser. PLAS '09. New York, NY, USA: ACM, 2009, pp. 1–7. [Online]. Available: http://doi.acm.org/-10.1145/1554339.1554341

[5] E. Chin, A. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communication in android," in *Procs. of the 9th Annual International Conference on Mobile Systems, Applications, and Services*, 2011.

[6] M. Conti, V. Nguyen, and B. Crispo, "Crepe: Context-related policy enforcement for android," in *Information Security*, ser. Lecture Notes in Computer Science, M. Burmester, G. Tsudik, S. Magliveras, and I. Ilic, Eds. Springer Berlin / Heidelberg, 2011, vol. 6531, pp. 331–345, 10.1007/978-3-642-18178-8_29. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-18178-8_29

[7] W. Enck, M. Ongtang, and P. McDaniel, "Understanding android security," *Security Privacy, IEEE*, vol. 7, no. 1, pp. 50 –57, jan.-feb. 2009.

[8] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 627–638. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046779

[9] A. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," UC Berkeley, Tech. Rep. UCB/EECS-2012-26, 2012.

[10] A. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," 2012.

[11] Google. Android Permissions, available at http://developer.android.com/guide/topics/manifest/permission-element.html.

[12] Google. Google play, available at https://play.google.com/.

[13] Google. (2012, Oct) Android Security - Using Permissions, http://developer.android.com/guide/topics/security/permissions.html.

[14] V. Moonsamy, M. Alazab, and L. Batten, "Towards an understanding of the impact of advertising on data leaks," *to appear in 'International Journal of Security and Networks (IJSN)'*, 2012.

[15] M. Nauman and S. Khan, "Design and implementation of a fine-grained resource usage model for the android platform," 2010.

[16] Paul Brodeur (Leviathan Security Group), "Zero-permission android applications, available at http://leviathansecurity.com/blog/archives/17-zero-permission-android-applications.html," April 2012.

[17] P. Pearce, A. Felt, G. Nunez, and D. Wagner, "Addroid: Privilege separation for applications and advertisers in android," in *Proceedings of AsiaCCS*, 2012.

[18] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38 –47, Feb 1996.

[19] S. Shekhar, M. Dietz, and D. Wallach, "Adsplit: Separating smartphone advertising from applications," *Arxiv preprint arXiv:1202.4030*, 2012.

[20] W. Shin, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A formal model to analyze the permission authorization and enforcement in the android framework," in *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, aug. 2010, pp. 944 –951.

[21] W. Shin, S. Kwak, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A small but non-negligible flaw in the android permission scheme," in *Policies for Distributed Systems and Networks (POLICY), 2010 IEEE International Symposium on*, july 2010, pp. 107 –110.

[22] T. Cannon (from Via Forensics), "No permission android app gives remote shell, available at https://viaforensics.com/security/nopermission-android-app-remote-shell.html," December 2011.

Paper 2

Performance Evaluation of Multi-Tier Ensemble Classifiers for Phishing Websites

Jemal Abawajy, Gleb Beliakov and School of Information Technology, Deakin University, Australia.

Andrei Kelarev, John Yearwood, School of Science, Information Technology and Engineering, University of Ballarat, Australia.

Performance Evaluation of Multi-Tier Ensemble Classifiers for Phishing Websites

Jemal Abawajy, Gleb Beliakov, Andrei Kelarev
School of Information Technology
Deakin University, 221 Burwood Hwy,
Burwood 3125, Australia
Email:{jemal.abawajy, gleb, kelarev}@deakin.edu.au

John Yearwood
School of Science, Information Technology
and Engineering, University of Ballarat,
P.O. Box 663, Ballarat, Victoria 3353, Australia
Email:j.yearwood@ballarat.edu.au

***Abstract*—This article is devoted to large multi-tier ensemble classifiers generated as ensembles of ensembles and applied to phishing websites. Our new ensemble construction is a special case of the general and productive multi-tier approach well known in information security. Many efficient multi-tier classifiers have been considered in the literature. Our new contribution is in generating new large systems as ensembles of ensembles by linking a top-tier ensemble to another middle-tier ensemble instead of a base classifier so that the top-tier ensemble can generate the whole system. This automatic generation capability includes many large ensemble classifiers in two tiers simultaneously and automatically combines them into one hierarchical unified system so that one ensemble is an integral part of another one. This new construction makes it easy to set up and run such large systems. The present article concentrates on the investigation of performance of these new multi-tier ensembles for the example of detection of phishing websites. We carried out systematic experiments evaluating several essential ensemble techniques as well as more recent approaches and studying their performance as parts of multi-level ensembles with three tiers. The results presented here demonstrate that new three-tier ensemble classifiers performed better than the base classifiers and standard ensembles included in the system. This example of application to the classification of phishing websites shows that the new method of combining diverse ensemble techniques into a unified hierarchical three-tier ensemble can be applied to increase the performance of classifiers in situations where data can be processed on a large computer.**

***Keywords*-phishing websites; ensemble classifiers; multi-tier ensembles; Random Forest**

I. Introduction

Experiments evaluating classifiers applied to particular areas are important, since their outcomes can be used in order to improve the performance of future applications and can contribute to choosing directions of future research. For any given algorithm that produces very good outcomes in certain applications, there always exist examples of data sets in other domains where different algorithms are more effective. This is also confirmed by the so-called "no-free-lunch" theorems, which imply that there does not exist one algorithm, which is best for all problems [45]. The performance of every category of algorithms depends on the dimension of a data set and the number of instances, types of attributes, the nature of functional relations and dependencies among the attributes and other parameters.

We introduce a new unified multi-tier construction of ensemble classifiers combining diverse ensembles into one integrated hierarchical system. This construction is illustrated in Figure 1. More explanations are given in Section II. Figure 2 shows how to aggregate the classifiers at different levels to obtain the multi-tier construction.

Every ensemble classifier at the middle tier of this construction is an integral part of the ensemble classifier at the top tier, and in turn every base classifier at the bottom tier is included as a part of the ensemble classifier of the middle tier, see Section II for more details. Using one ensemble as an integral part of another ensemble makes it easy to set up and run such ensembles, even though they can be very large.

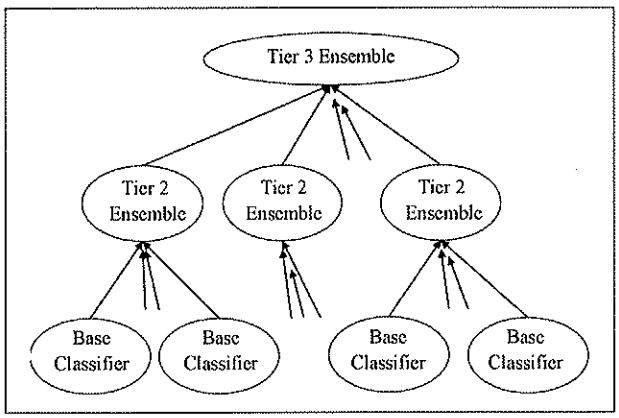

Figure 1. Data flow in three-tier ensemble classifiers generated as ensembles of ensembles by WEKA

The present article is devoted to experiments comparing the performance of new three-tier classifiers, their base classifiers and standard ensemble classifiers in the special case of an application to the detection of phishing websites. While phishing is an important direction that has been actively investigated recently, the aim of our paper is to develop a general technique that may be useful for various applications in information security. Let us refer to the Anti-Phishing Working Group [1], OECD Task Force on

Spam [34] and recent papers [4], [8], [14], [18], [19], [28], [47] for background information and preliminaries on phishing. The authors hope that the outcomes of this example of application prove helpful for the future development of classifiers in other branches of information security too.

Our new results show that novel three-tier ensemble classifiers achieved substantially better performance in comparison with the base classifiers or standard ensemble classifiers. This demonstrates that the new method of combining diverse ensemble techniques into one unified three-tier ensemble incorporating diverse ensembles as parts of other ensembles can be applied to improve classifications.

The paper is organised as follows. Section II describes new multi-tier ensemble classifiers investigated in this paper. Section III is devoted to preprocessing of data. Section IV deals with the base classifiers and ensemble classifiers. Section V contains the outcomes of experiments comparing the effectiveness of base classifiers, ensemble classifiers and three-tier ensemble classifiers. These results are discussed in Section VI. Main conclusions are presented in Section VII.

For consistency, in writing the paper an attempt was made to use present simple tense throughout to describe what is done in this article as well as to refer to background information. Past simple and present perfect tenses were reserved to the discussion of articles published previously and to the description of our experiments, since all our tests had been completed before we started writing the paper.

II. Three-tier Ensemble Classifiers

Ensemble classifiers combine a collection of base classifiers into a common classification system. Here we introduce and explain our new multi-tier ensemble construction inspired by previous research in the literature. Our experiments evaluate performance of such large three-tier ensemble classifiers combining diverse ensemble classifiers on two tiers into one unified system. Several efficient multi-tier classifiers and more general multi-classifier systems have been explored, for example, in the previous publications [19], [20], [23], [24], [25].

Several techniques for the design of ensemble classifiers are well known in artificial intelligence and data mining. This paper introduces a new three-tier construction, which makes it easy to combine diverse ensemble methods into one scheme. Our experiments are devoted to performance evaluation of new large three-tier ensemble classifiers for phishing websites.

This paper deals with large three-tier ensemble classifiers, illustrated in Figure 1. The direction of arrows in the diagram indicates the flow of data. All base classifiers pass their output on to Tier 2 ensemble classifiers. The Tier 2 ensemble classifiers combine the output of base classifiers. Their output in turn in analysed by the Tier 3 ensemble classifier that makes the final decision for the whole multi-tier classification system. Arcs not connected to classifiers indicate the direction of possible data flow from additional classifiers. The whole system may involve thousands of base classifiers, but it is easy to set it up, since in most cases the Tier 2 classifiers generate the whole collection of their base classifiers automatically given just one instance of a base classifier. Likewise, all Tier 2 ensemble classifiers are generated by the Tier 3 ensemble classifier automatically given only one instance of a Tier 2 ensemble classifier. This means that the Tier 3 ensemble classifier generates its Tier 2 classifiers and executes them in exactly the same way as it usually handles base classifiers. Similarly, each Tier 2 ensemble applies its method to combine its base classifiers as usual. The whole system is generated automatically in SimpleCLI, as illustrated in Figure 2.

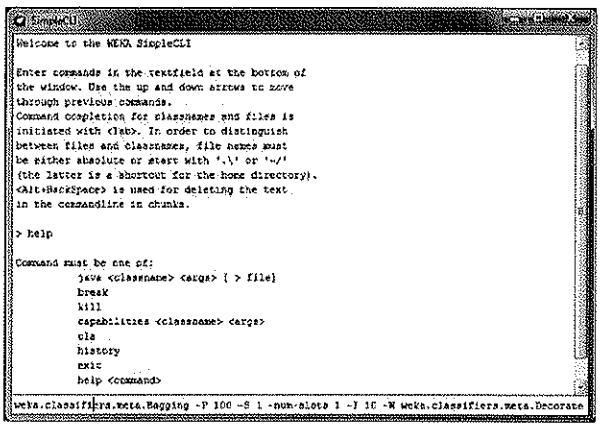

Figure 2. A part of command line generating three-tier ensemble in SimpleCLI

Thus, in this paper we introduce and investigate a three-tier ensemble construction originating as a contribution to the general approach introduced by previous authors. We obtain new results evaluating performance of such large three-tier ensemble classifiers. These new results show, in particular, that Random Forest performed best in this setting for our data set considered in this article, and that novel three-tier ensemble classifiers can be used to achieve further improvement of the classification outcomes. The three-tier ensemble classifiers based on Random Forest achieved better performance compared with the base classifiers or simpler ensemble classifiers.

Large three-tier ensemble classifiers require a lot of computer memory to train, especially for very large data sets, where they can be used to improve performance. If a data set is small and an ensemble classifier is larger, then it will revert to using just one base classifier and produce the same outcomes as the base classifier. As we will see in Section V below, our experiments show that such large three-tier ensemble classifiers are effective if diverse ensembles are combined at different tiers of the three-tier ensemble

classifier. The authors believe that this approach to designing ensembles of classifiers deserves further investigation for other large data sets and application directions too.

III. Feature Extraction

We used the same set of features extracted from the data set of phishing websites considered by the authors in [4], since it is suitable for this study. Our new experiments used a collection of simple features extracted during work on the paper [4]. Similar data sets are available from the downloadable databases at the PhishTank [35]. The present article investigates a novel method for improving performance of the classifiers, and we did not attempt to extract more sophisticated collections of features. The extraction of features is very important for applications, for example, see [2], [21], [22], [26], [29], [30], [31], [33], [40], [41] and [42], but it is not the main focus of the present article.

Since this paper concentrates on the contribution of multi-tier ensembles, for the purposes of this work, we applied the bag-of-words model and extracted only a simple collection of the features reflecting the content of the websites. As in [4], we used *term frequency–inverse document frequency* word weights, or TF-IDF weights, to select words as features. Features were extracted using a flexible preprocessing and feature extraction system implemented in Python by the third author.

We collected a set of words with highest TF-IDF scores in all websites of the data set. For each website, the TF-IDF scores of these words in the website were determined. These weights and additional features were assembled in a vector. In order to determine the TF-IDF scores we used Gensim, a Python and NumPy package for vector space modelling of text documents. These features were collected in a vector space model representing the data set.

IV. Base Classifiers and Ensemble Classifiers

The following classifiers available in WEKA [13] were used as base classifiers in our experiments with outcomes presented in Section V: FURIA [17], J48 [37], LibLINEAR [9], LibSVM [7], [10], [16], Random Forest [6], SMO [15], [27], [36], These robust classifiers were chosen since they represent most essential types of classifiers available in WEKA [13] and performed well for our data set.

We used SimpleCLI command line in WEKA [13] to investigate the performance of the following ensemble techniques: AdaBoost [12], Bagging [5], Dagging [39], Decorate [32], Grading [38], MultiBoost [43] and Stacking [46].

Consensus functions can also be used as a replacement for voting to combine the outputs of several classifiers. Here we use the HBGF consensus function, following the recommendations of [11] and our previous experience with consensus functions presented in [8], [47] and [48]. The HBGF consensus function is based on a bipartite graph with two sets of vertices: classes and elements of the data set.

V. Experiments Evaluating Performance

We used 10-fold cross validation to evaluate the effectiveness of classifiers in all experiments. The following measures of performance of classifiers are often used in this research direction: precision, recall, F-measure, accuracy, sensitivity, specificity and Area Under Curve also known as the Receiver Operating Characteristic or ROC area.

Notice that weighted average values of the performance metrics are usually used. This means that they are calculated for each class separately, and a weighted average is found then. In particular, our results included in this paper deal with the weighted average values of precision. In contrast, the *accuracy* is defined for the whole classifier as the percentage of all websites classified correctly, which means that this definition does not involve weighted averages in the calculation. *Precision* of a classifier, for a given class, is the ratio of true positives to combined true and false positives.

Sensitivity is the proportion of positives (phishing websites) that are identified correctly. *Specificity* is the proportion of negatives (legitimate websites) which are identified correctly. Sensitivity and specificity are measures evaluating binary classifications. For multi-class classifications they can be also used with respect to one class and its complement. Sensitivity is also called True Positive Rate. *False Positive Rate* is equal to 1 - specificity. These measures are related to recall and precision. *Recall* is the ratio of true positives to the number of all positive samples (i.e., to the combined true positives and false negatives). The recall calculated for the class of phishing websites is equal to sensitivity of the whole classifier.

All tables of outcomes in this paper include the F-measure, since it combines precision and recall into a single number evaluating performance of the whole system, [44]. The F-measure is equal to the harmonic mean of precision and recall

$$\text{F-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \qquad (1)$$

The weighted average F-measure is contained in the standard WEKA output for all classifiers.

First, we include the results of experiments comparing the performance of several base classifiers for phishing websites. The results obtained for five best classifiers are presented in Figure 3. Random Forest outperformed other base classifiers for the phishing websites data set.

Second, we include the results of experiments comparing standard ensemble classifiers in their ability to improve the outcomes. We compared AdaBoost, Bagging, Dagging, Decorate, Grading, HBGF, MultiBoost and Stacking based on RandomForest.

F-measures of the resulting ensemble classifiers are presented in Figure 4, which shows improvement as compared to the base classifiers. In these tests all ensembles were used

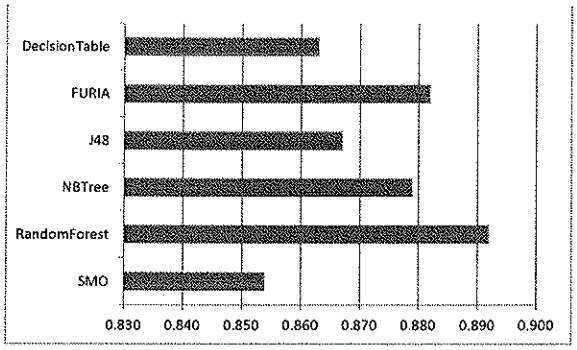

Figure 3. F-measure of base classifiers for phishing websites

with one and the same base classifier, RandomForest, in all tests.

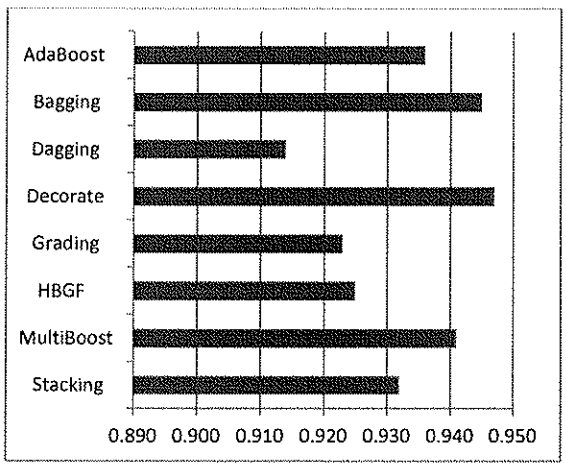

Figure 4. F-measure of ensemble classifiers for phishing websites

Finally, we include the results of experiments evaluating the 3-tier ensemble method. This is the main topic of the paper. These experiments included the all combinations of Bagging, Decorate and MultiBoost, since these ensemble methods produced better F-measures in Figure 4. Each three-tier ensemble classifier contains one ensemble in Tier 3. It generates or includes a whole set of Tier 2 ensembles and executes them in exactly the same way as it handles any other base classifiers. In turn, each Tier 2 ensemble applies its method to combine its base classifiers in Tier 1. We have not included repetitions of the same ensemble technique in both tiers, since tests have shown that they do not produce further improvement. The outcomes of the three-tier ensemble classifiers are presented in Figure 5. A part of command generating one of these multi-level ensembles in SimpleCLI is shown in Figure 2.

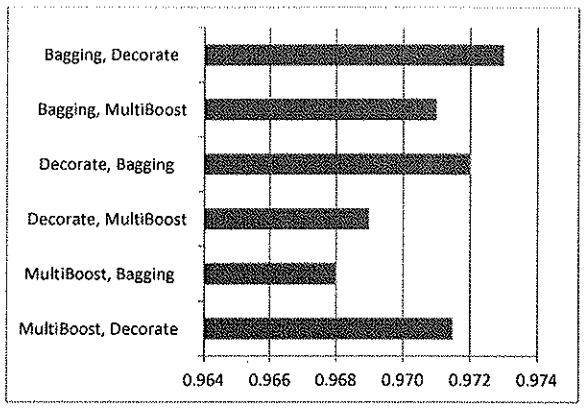

Figure 5. F-measure of three-tier ensemble classifiers for phishing websites

VI. Discussion

Our work shows that large three-tier ensamples are quite easy to use and can be applied to improve classifications, if diverse ensembles are combined at different tiers. It is an interesting question for future research to investigate three-tier ensembles for other large datasets.

Random Forest outperformed other base classifiers for the phishing websites data set, and Decorate improved its outcomes better than other ensemble meta classifiers did. The best outcomes were obtained by the new combined three-tier ensemble classifier where Bagging is used in Tier 3 and Decorate in Tier 2.

The performance of ensemble classifiers considered in this paper depends on several numerical input parameters. In all experiments we used them with the same default values of these parameters in order to have a uniform equivalent comparison of outcomes across all of these ensemble classifiers. It may be also possible to obtain further improvement to the outcomes by optimizing their parameters with optimization techniques presented in [3]. At present the ranges of parameter values remain restricted by the size of memory available on personal computers for training of large three-tier ensemble classifiers.

VII. Conclusion

We carried out a systematic investigation of new automatically generated multi-tier ensemble classifiers, where diverse ensembles are combined into a unified system by integrating different ensembles at a lower tier as a part of another ensemble at the top tier. Our experiments evaluated the performance of these large three-tier ensemble classifiers for a data set of phishing websites and have demonstrated the

feasibility and performance of the approach. The experimental outcomes show that these multi-tier ensemble classifiers can be used to improve classifications. They produced better results compared to the base classifiers or standard ensemble classifiers.

ACKNOWLEDGMENT

The authors are grateful to four referees for thorough reports with comments and corrections that have helped to improve the text of this article, and for suggesting several possible directions for future research. All authors were supported by Deakin-Ballarat collaboration grants.

REFERENCES

[1] APWG, "Anti-Phishing Working Group," http://apwg.org/, accessed 10 June 2012.

[2] L. Batten, J. Abawajy, and R. Dose, "Prevention of information harvesting in cloud service environments," in *Proceedings of the 1st International Conference on Cloud Computing and Services Science, CLOSER 2011*, 2011, pp. 66–72.

[3] G. Beliakov and J. Ugon, "Implementation of novel methods of global and non-smooth optimization: GANSO programming library," *Optimization*, vol. 56, pp. 543–546, 2007.

[4] G. Beliakov, J. Yearwood, and A. Kelarev, "Application of rank correlation, clustering and classification in information security," *Journal of Networks*, vol. 7, pp. 935–955, 2012.

[5] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[6] ——, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[7] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[8] R. Dazeley, J. Yearwood, B. Kang, and A. Kelarev, "Consensus clustering and supervised classification for profiling phishing emails in internet commerce security," in *Knowledge Management and Acquisition for Smart Systems and Services, PKAW2010*, ser. Lecture Notes in Computer Science, vol. 6232, 2010, pp. 235–246.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR – a library for large linear classification," Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear/, viewed 21 February 2012, 2012.

[10] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training SVM," *J. Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[11] X. Fern and C. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *21st International Conference on Machine Learning, ICML'04*, vol. 69. New York, NY, USA: ACM, 2004, pp. 36–43.

[12] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Internat. Conf. Machine Learning*, 1996, pp. 148–156.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations*, vol. 11, pp. 10–18, 2009.

[14] I. Hamid and J. Abawajy, "Hybrid feature selection for phishing email detection," in *International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP 2011*, ser. Lecture Notes in Computer Science, vol. 7017, 2011, pp. 266–275.

[15] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Advances in Neural Information Processing Systems*, 1998.

[16] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Dept. Computer Science, National Taiwan University, http://www.csie.ntu.edu.tw/~cjlin, Initial version: 2003, last updated: April 15, 2010.

[17] J. Huehn and E. Huellermeier, "FURIA: An algorithm for unordered fuzzy rule induction," *Data Mining and Knowledge Discovery*, vol. 19, pp. 293–319, 2009.

[18] R. Islam and J. Abawajy, "A multi-tier phishing detection and filtering approach," *Journal of Network and Computer Applications*, p. to apper soon, 2012.

[19] R. Islam, J. Abawajy, and M. Warren, "Multi-tier phishing email classification with an impact of classifier rescheduling," in *10th International Symposium on Pervasive Systems, Algorithms, and Networks, ISPAN 2009*, 2009, pp. 789–793.

[20] R. Islam, J. Singh, A. Chonka, and W. Zhou, "Multi-classifier classification of spam email on an ubiquitous multi-core architecture," in *Proceedings – 2008 IFIP International Conference on Network and Parallel Computing, NPC 2008*, 2008, pp. 210–217.

[21] R. Islam, R. Tian, L. Batten, and S. Versteeg, "Classification of malware based on string and function feature selection," in *CTC 2010: Proceedings of the Second Cybercrime and Trustworthy Computing Workshop*, 2010, pp. 9–17.

[22] R. Islam, R. Tian, V. Moonsamy, and L. Batten, "A comparison of the classification of disparate malware collected in different time periods," *Journal of Networks*, vol. 7, pp. 956–955, 2012.

[23] R. Islam and W. Zhou, "Email classification using multi-tier classification algorithms," in *Proc. 7th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2008*, 2008.

[24] R. Islam, W. Zhou, and M. Chowdhury, "Email categorization using (2+1)-tier classification algorithms," in *Proceedings – 7th IEEE/ACIS International Conference on Computer and Information Science, IEEE/ACIS ICIS 2008, In conjunction with 2nd IEEE/ACIS Int. Workshop on e-Activity, IEEE/ACIS IWEA 2008*, 2008, pp. 276–281.

[25] R. Islam, W. Zhou, M. Gao, and Y. Xiang, "An innovative analyser for multi-classifier email classification based on grey list analysis," *Journal of Network and Computer Applications*, vol. 32, pp. 357–366, 2009.

[26] B. Kang, A. Kelarev, A. Sale, and R. Williams, "A new model for classifying DNA code inspired by neural networks and FSA," in *Advances in Knowledge Acquisition and Management*, ser. Lecture Notes in Computer Science, vol. 4303, 2006, pp. 187–198.

[27] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[28] A. Kelarev, S. Brown, P. Watters, X.-W. Wu, and R. Dazeley, "Establishing reasoning communities of security experts for internet commerce security," in *Technologies for Supporting Reasoning Communities and Collaborative Decision Making: Cooperative Approaches*. IGI Global, 2011, pp. 380–396.

[29] A. Kelarev, B. Kang, and D. Steane, "Clustering algorithms for ITS sequence data with alignment metrics," in *AI 2006: Advances in Artificial Intelligence, 19th Australian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Artificial Intelligence, vol. 4304, 2006, pp. 1027–1031.

[30] A. Kelarev, R. Dazeley, A. Stranieri, J. Yearwood, and H. Jelinek, "Detection of CAN by ensemble classifiers based on Ripple Down Rules," in *Pacific Rim Knowledge Acquisition Workshop, PKAW2012*, ser. Lecture Notes in Artificial Intelligence, vol. 7457, 2012, pp. 147–159.

[31] A. Kelarev, A. Stranieri, J. Yearwood, and H. Jelinek, "Empirical study of decision trees and ensemble classifiers for monitoring of diabetes patients in pervasive healthcare," in *Network-Based Information Systems, NBiS-2012*, 2012, pp. 441–446.

[32] P. Melville and R. Mooney, "Creating diversity in ensembles using artificial data," *Information Fusion*, vol. 6, pp. 99–111, 2005.

[33] V. Moonsamy, R. Tian, and L. Batten, "Feature reduction to speed up malware classification," in *Information Security Technology for Applications*, ser. Lecture Notes in Computer Science, P. Laud, Ed. Springer Berlin / Heidelberg, 2012, vol. 7161, pp. 176–188.

[34] OECD, "Organisation for Economic Cooperation and Development, OECD task force on spam, OECD anti-spam toolkit and its annexes," http://www.oecd.org/dataoecd/63/28/36494147.pdf, accessed 20 November 2011.

[35] PhishTank, "Developer information," http://www.phishtank.com/developer_info.php, viewed 20 September 2011.

[36] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods – Support Vector Learning*, 1998.

[37] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[38] A. Seewald and J. Fuernkranz, "An evaluation of grading classifiers advances in intelligent data analysis," in *Advances in Intelligent Data Analysis*, ser. Lecture Notes in Computer Science, vol. 2189/2001, 2001, pp. 115–124.

[39] K. Ting and I. Witten, "Stacking bagged and dagged models," in *Fourteenth international Conference on Machine Learning*, 1997, pp. 367–375.

[40] H. Vu, G. Li, and G. Beliakov, "A fuzzy decision support method for customer preferences analysis based on Choquet Integral," in *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2012*, 2012, pp. 1–8.

[41] H. Vu, S. Liu, Z. Li, and G. Li, "Microphone identification using one-class classification approach," in *Applications and Techniques in Information Security, ATIS 2011*, 2011, pp. 29–37.

[42] X. Wang, W. Niu, G. Li, X. Yang, and Z. Shi, "Mining frequent agent action patterns for effective multi-agent-based web service composition," in *7th International Workshop on Agents and Data Mining Interation, ADMI 2011*, ser. Lecture Notes in Artificial Intelligence, vol. 7103, 2012, pp. 211–227.

[43] G. Webb, "Multiboosting: A technique for combining boosting and wagging," *Machine Learning*, vol. 40, pp. 159 – 196, 2000.

[44] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Amsterdam: Elsevier/Morgan Kaufman, 2011.

[45] D. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Computation*, vol. 8, pp. 1341–1390, 1996.

[46] ——, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[47] J. Yearwood, D. Webb, L. Ma, P. Vamplew, B. Ofoghi, and A. Kelarev, "Applying clustering and ensemble clustering approaches to phishing profiling," in *Data Mining and Analytics 2009, Proc. 8th Australasian Data Mining Conference, AusDM 2009*, ser. CRPIT, P. Kennedy, K. Ong, and P. Christen, Eds., vol. 101. Melbourne, Australia: ACS, 2009, pp. 25–34.

[48] J. Yearwood, B. Kang, and A. Kelarev, "Experimental investigation of classification algorithms for ITS dataset," in *Pacific Rim Knowledge Acquisition Workshop, PKAW 2008*, Hanoi, Vietnam, 15–16 December 2008, 2008, pp. 262–272.

Paper 3

Detecting SMS-based Control Commands in a Botnet from Infected Android Devices by Anh Nguyen and Lei Pan, School of Information Technology, Deakin University, Australia.

Detecting SMS-based Control Commands in a Botnet from Infected Android Devices

Anh Nguyen
School of Information Technology
Deakin University
Burwood, VIC 3125, Australia
Email: anguyen@deakin.edu.au

Lei Pan
School of Information Technology
Deakin University
Burwood, VIC 3125, Australia
Email: l.pan@deakin.edu.au

Abstract

An increasing number of Android devices are being infected and at risk of becoming part of a botnet. Among all types of botnets, control and command based botnets are very popular. In this paper, we introduce an effective and efficient method to detect SMS-based control commands from infected Android devices. Specifically, we rely on the important radio activities recorded in Android log files. These radio activities are currently overlooked by researchers. We show the effectiveness of our method by using the examples from published literature. Our method requires much less user knowledge but is more generic than traditional approaches.

Index Terms

botnet; logging; Android; digital forensics

1. Introduction

With the increasing number of botnets and widespread of malware, a large number of Android devices are being infected and thus become a part of botnet. Android devices are attractive and vulnerable, due to their design features as a free and open-source operating system for mobile devices [1]. Furthermore, Android systems have a weak security model in which special permissions are granted easily to perform malicious tasks.

The interest of building a botnet with mobile devices is shown fairly recent. Many mobile devices become part of command and control botnets [2]. Zeng et al. [3] build a simple botnet with less than 200 control commands on a mobile network. They also show that even if an innocent user deletes a control message sent via SMS, the command has been regardlessly executed upon receipt of the SMS.

Traditional approaches to unveil the existence of a bot take a long time. Our research question is to effectively and efficiently detect the control commands from an Android device transmitted within a botnet. We take advantage of Android log files which are embedded in the kernel level, and in particular, the radio activity logs which record many important system events related to botnet activities. Our contributions in this paper include a detailed introduction of the Android log system and the identification of three important radio activities associated with SMS-based botnet activities. Our new detection method requires much less user knowledge but is more generic than traditional reverse engineering approach to decompile the binary files into source codes.

The rest of this paper is organized as follows: Section 2 lists the related work on Android malware analysis; Section 3 illustrates Android log files; Section 4 describes our detection method, our experiments and the results; and we conclude in Section 5.

2. Literature Review

There are three commonly used malware analysis approaches for Android devices — reviewing source codes, applying a sandbox and analyzing Android kernel information. Reviewing the source codes of a program is a reliable approach for analyzing Android malware. For example, Blasco [4] analyzes an Android malware package (named `RU.apk`) by reverse engineering the binary file into source codes. The *apk* package is firstly uncompressed by using a ZIP program; then Blasco locates the `class.dex` file from the uncompressed files; a Google tool named *Dex2jar* is then applied to convert `class.dex` to `class.jar`; finally, the Java decompiler *jd* is applied to retrieve source codes from `class.jar`. More specifically, the app `RU.apk` secretly forwards

all incoming SMS messages to two Russian premium numbers. Blasco's case study shows the power of reverse engineering an app package *apk*; however, it is not applicable when the *apk* file is unavailable. The second approach to analyzing a botnet is to use a sandbox. A sandbox is injected to an Android device where any use of special permissions such as calling a phone, sending SMS, connecting to the Internet is handled by the sandbox instead of the handset [5], [6]. The third approach is to analyze kernel-level information. As revealed in [7], [8], different apps revoke system calls differently according to the recorded information in kernel-level log files. The collected log files contain the filtered events with all system calls from the target application. By using this method, Isohara et al. build app signatures based on the intensity of I/O and process management operations.

However, none of the above approaches applies to our case. Our reasons are threefold: It is generally infeasible to acquire botnet source codes for many botnets do not use a static binary program or an app package; the sandboxing technique is complicated and often blocks bot's activities; botnets evolve so quickly that fixed signatures rarely work for a long period of time. Inspired by all of the existing work we focus on the recorded behaviors of a handset.

Lessard and Kessler [9] list a few forensic methods to retrieve information from an Android device. One of the practical combinations of techniques is to firstly enable USB debugging, then to gain the root access to the device, and lastly to create *dd* disk images of the mounted file systems on Android. These acquired image files can be analyzed with popular forensic tools such as *Forensic ToolKit (FTK)* and *EnCase*. However, this process is labor-intensive and error-prone. Zhong et al. [10] use a customized Google map on Android device to retrieve an individual user's geo-spatial data which is then sent to the police via 3G network. Though sending data to the police seems to be safe and reliable, it brings excessive concerns of privacy and misuse of personal data. An alternative approach is to use commercially available and automated phone forensic tools such as *Device Seizure* from Paraben and *Oxygen forensic suite*. Both approaches require the USB debugging mode on the handset to be enabled. However, changing the setting of a handset may alter digital evidence stored onboard. And it is challenging to enable any write blocking mechanism on a handset.

The next section will introduce Android logging system and log files.

3. Android Log Files

Google modified the original Linux kernel to suit Android. More than 160 kernel files are created or modified to support new 'Log Device' and 'Android Debug Bridge' (*adb*). Hence, Android systems use *syslog* for the unmodified Linux kernel and its own app log for the new 'Log Device' and 'Android Debug Bridge'. The *syslog* file format is defined as an international standard in RFC5424 [11], and the app log file format is defined by Google.

3.1. Android Syslog

Syslog has a hierarchical structure and a simple syntax as follows:

- A log file consists of multiple *syslog* messages.
- Each *syslog* message has two or three parts: the first part is a header; the second part is structured data; and the third part is an optional message. A white space symbol is inserted between each part to separate them.
- The header consists of a priority value, the syslog version, a timestamp, a hostname, an application name, a process ID and a message ID.
- The structured data is a list of parameter names and their associated values.
- The optional message contains the detailed message which is stored in ASCII or in UTF-8 codes.
- By default, a missing value is indicated by a hyphen symbol in ASCII.

Comparing with a standard Linux system, Android system excludes the program `syslogd`. So, the ordinary Linux log directory `/var/log` does not exist on Android; instead, an equivalent file `proc/kmesg` is created on Android for logging Linux kernel messages. There are two methods to access this file — either by using the Linux command `dmesg` through the 'Android Debug Bridge' (*adb*) or by directly accessing the file via the file system.

Originally designed for hardware debugging purposes, kernel messages mainly contain basic information of the hardware and the operating system. Since these kernel messages keep track of the boot process, the running system processes and their process IDs. These messages have forensic values. Below is an example of kernel message from an Android device. Specifically, the following log entries include some warning, informational and notice messages generated during the boot process: The kernel information with a timestamp is the first message; then the next three messages show the

Android device has a ARMv7 processor and 512MB RAM. The fifth message shows the memory mapping; and the rest indicate the process of mounting three storage devices `mtdblock1`, `mtdblock2` and `mtdblock3` as `yaffs` file system.

```
<5>Linux version 2.6.29-gc497e41        \
(kroot@kennyroot.mtv.corp.google.com) \
(gcc version 4.4.3 (GCC) ) #2 Thu Dec \
8 15:07:43 PST 2011
<4>CPU: ARMv7 Processor [410fc080]      \
revision 0 (ARMv7), cr=10c5387f
<4>CPU: VIPT nonaliasing data cache,    \
VIPT nonaliasing instruction cache
<6>Memory: 512MB = 512MB total
<5>Memory: 515712KB available (2756K    \
code, 683K data, 108K init)
<6>yaffs: dev is 32505856 name is       \
"mtdblock0"
<4>yaffs: Attempting MTD mount on 31.0\
, "mtdblock0"
<6>yaffs: dev is 32505857 name is       \
"mtdblock1"
<4>yaffs: Attempting MTD mount on 31.1\
, "mtdblock1"
<6>yaffs: dev is 32505858 name is       \
"mtdblock2"
<4>yaffs: Attempting MTD mount on 31.2\
, "mtdblock2"
```

The numbers appearing in the beginning of each message are bounded by brackets. According to [11], each of these numbers indicates priority values. A priority value is calculated by multiplying the facility number by 8 and adding the severity level. And kernel messages have zero as their facility number. Since Android only stores kernel messages in *syslog* format, the priority values in the above example are equal to the default *syslog* severity levels:

0 Emergency: system is unusable
1 Alert: action must be taken immediately
2 Critical: critical conditions
3 Error: error conditions
4 Warning: warning conditions
5 Notice: normal but significant condition
6 Informational: informational messages
7 Debug: debug-level messages

3.2. Android App Log

Android app logs are handled by the Android Logging system, which is completely separate from the Linux kernel log system. Unlike hardware information, the Android Logging System emphasizes the status of the running programs on the device. It includes four basic components: An Android kernel driver called `logger`, four logging buffers allocated to store log messages — *events*, *main*, *radio* and *system*, a standalone program `logcat` to view log messages, and a programming interface (via *Eclipse* and *DDMS*) designed to view and filter the logs.

To illustrate the Android Logging System, we use the picture from [12] in Figure 1. There are two devices in the figure — a target and a host. The target is the Android device whose logs are being analyzed, and the host is the computer to which the device is connected. On the target, the log files are stored in the `/dev/log` directory at the lowest level. Corresponding to the four log buffers, the

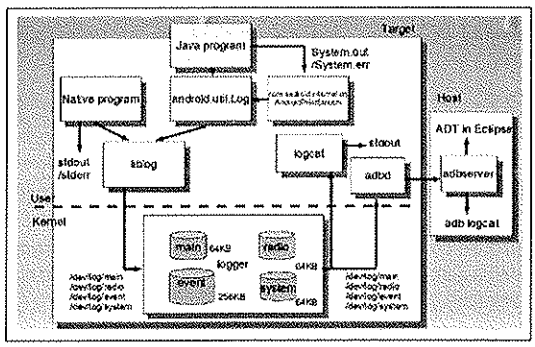

Figure 1: Android Logging System from [12]

log files are `/dev/log/main`, `/dev/log/radio`, `/dev/log/event` and `/dev/log/system`. In `/dev/log/radio`, we often find the log messages with message tags "RIL", "AT", "RILD", "RILC", "RILJ", and "GSM". All of the log files are 64 KB except that `/dev/log/event` is 256 KB. On each Android handset, the log sizes are fixed so that the newer records will replace the earlier ones. The contents of each record are generated with the `liblog` library before they are stored in these log files. As shown in Figure 1, the `liblog` library is in charge of recording all application logs. The native Android apps can directly access `liblog`, and the third-party apps should use the API `android.util.log` to access `liblog`. In addition to outputting the log messages to the log files, the programs may display the messages on the device screen via `stdout` and `stderr` channels.

Similar to the kernel log messages, each Android log message contains multiple parts — a message tag indicating the application that the message originated from, a timestamp, the priority of the event represented by the message and the log message itself. The `logcat` tool chronologically sorts the events and can be invoked on the target or on the host through the Android Debug Bridge Daemon (*adbd*). By default, the Android `logcat` tool only outputs three parts of the log message — the priority tag denoted by one of the 5 letters ('E' for ERROR, 'I' for INFO, 'D' for DEBUG, 'W' for WARN, and 'V' for VERBOSE), the message tag indicating the message origin and the log message itself. Furthermore, the separation symbols vary between different parts: a slash symbol '/' is immediately after a priority tag and before a message tag, and a colon symbol ':' is immediately after the message tag and before the log message.

The following examples are the outputs from `logcat`. The command `logcat -b main` retrieves the main application log `/dev/log/main`, and the following error message indicates the absence of an error trace file:

```
E/Trace   (264): error opening trace \
file: No such file or directory (2)
```

The command `logcat -b events` retrieves the system events log `/dev/log/events`, and the following information message indicates the start of Android bootup process:

```
I/boot_progress_start(36): 22060
```

`logcat -b radio` retrieves the radio and phone-related information log `/dev/log/radio`, and the fol-

lowing debugging message indicates an incoming phone call:

```
D/AT      (  943): AT< RING
```

The command `logcat -b system` retrieves the system debugging log `/dev/log/system`, and the following error message indicates a run-time error caused by missing a method:

```
E/AndroidRuntime( 8681): java.lang.NoSuch\
MethodError: com.herongyang.AboutAndroid.\
getObbDir
```

The next section shows our proposed method of using Android log files to detect botnet control commands via SMS.

4. Using Android Logs to Detect Botnet Control Commands

Our detection method is to search for radio events. Our reasons are 1) botnet uses radio as a channel to send and receive short control commands to meet the need of frequent communications to its zombies; 2) the radio log on actual handsets is large enough to store generally a few days' telecommunication logs; 3) the address of destination nodes and source nodes are stored in clear text in the radio logs; 4) the radio logs are well kept in the kernel so that a complete erase of all entries is difficult.

Specifically, we focus on two categories in `/dev/log/radio`: The first category is to retrieve the device information such as `D/RIL onRequest: GET_IMSI` and `D/RIL onRequest: GET_IMEI` which retrieve the device IMSI and IMEI numbers respectively; the second category is to send SMS messages such as `D/RIL onRequest: SEND_SMS`. Our detection method is very simple but effective. We show two examples below.

To carry out the experiment, we use the information of a prepaid Optus device and the same SMS examples from [3]. The first example short message reads as:

> Free ringtones download at www.myringtone.com, using username VIP, password YTJiNGQxMWw to log on

The only obscured text is "YTJiNGQxMWw" which is a Base-64 encoded string "a2b4d11". Without knowing what this number means, we acquire the radio log of the recipient's device. By filtering out the irrelevant entries and the ones before the recipient of the SMS, we obtain the following 3 entries:

```
D/RIL (  32): onRequest: GET_IMSI
D/RIL (  32): onRequest: GET_IMEI
D/RIL (  32): onRequest: SEND_SMS
```

Now, the behavior of this apparent spam SMS triggers our device to retrieve system information (at least IMSI and IMEI numbers) and then send a new SMS to somewhere else. We then examine the radio log and find that our IMSI and IMEI numbers have been sent to an American number +14083631980. The following debugging message is located in one of the later events:

```
D/AT  (  32): AT> AT+CMGS="+14083631980"\
,145 IMSI=085950200100263914          \
IMEI=353502020487979^Z
```

where the command "AT+CMGS" indicates that the SMS has been sent to +14083631980, "145" is the international phone number format, and the characters before "^Z" are the SMS body. Our findings are consistent to the original paper, which the first SMS is a bot command "SEND_SYSINFO a2b4d11".

The second SMS is also from [3]:

> Your paypal account was hijacked (Err msg: NzkxMjAzNDIxODExMDUyM183Mz). Respond to http://www.bhocxx.paypal.com using code Q3MDk2NDUyXzEyMzQ1Njc4

This time, we need to concatenate the two encoded strings "NzkxMjAzNDIxODExMDUyM183Mz" and "Q3MDk2NDUyXzEyMzQ1Njc4". We use Base-64 to decode the result string as "7912034218110523_7347096452_12345678". Without knowing what these three numbers mean, we acquire the radio log of the recipient's device. By filtering out the irrelevant entries and the ones before the recipient of the SMS, we obtain the following entry:

```
D/RIL (  32): onRequest: SEND_SMS
```

Now, this apparent spam SMS lets the device send out a SMS. We examine the radio log and find that a blank SMS has been sent to an American number +17347096452. The following debugging message is located in one of the later events:

```
D/AT  (  32): AT> AT+CMGS="7347096452",\
129 ^Z
```

where the command "AT+CMGS" indicates that the SMS has been sent to 7347096452, "145" is the phone number format without the country code, and the characters before "^Z" indicate an SMS with an empty body. Our findings are also consistent to the original paper, which the second SMS is a bot command "FIND_NODE 7912034218110523_7347096452_12345678". According to [3], this command attempts to locate a bot whose ID is 7912034218110523 with passcode 12345678 to the bot whose phone number is 7347096452.

In this section, we propose our detection method and apply it to treat the two disguised SMS messages. We successfully detect botnet control commands with the help of the log file `/dev/log/radio`. The success of our method relies on the integrity and trustworthiness of the radio log file which is currently overlooked by many researchers. As a simple solution to the log integrity problem, we have implemented an Android program which captures all the log entries in real-time and stores the log entries in a SQLite database on external storage devices. We believe that this mitigation method would help to prevent the loss of Android logs. We conclude this paper in the next section.

5. Conclusions and Future Work

In this paper, we explain the Android logging system in details. We identify a few indicative yet significant logging symbols to effectively detect the control commands of a botnet. Our method relies on the Android radio log which is overlooked by most researchers. We show the effectiveness of our method by using the examples from published literature. Our method requires much less user knowledge but is more generic than traditional reverse engineering approach.

Our future work is to improve the current method so that it does not require a non-tampered radio log file. Since validating log entries is beyond the scope of this paper,

our next step is to distinguish whether a log message is genuine and to recover the deleted or altered log messages prior to the log analysis.

References

[1] R. Meier, *Professional Android 2 Application Development*. John Wiley & Sons, 2010.

[2] C. Xiang, F. Binxing, Y. Lihua, L. Xiaoyi, and Z. Tianning, "Andbot: towards advanced mobile botnets," in *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, ser. LEET'11, 2011, pp. 11–17.

[3] Y. Zeng, K. G. Shin, and X. Hu, "Design of sms commanded-and-controlled and p2p-structured mobile botnets," in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, ser. WISEC '12, 2012, pp. 137–148.

[4] J. Blasco, "Introduction to android malware analysis," *INSECURE Magazine*, vol. 34, pp. 25–37, 2012.

[5] T. Blasing, L. Batyuk, A.-D. Schmidt, S. Camtepe, and S. Albayrak, "An android application sandbox system for suspicious software detection," in *Proceedings of the 5th International Conference on Malicious and Unwanted Software (MALWARE)*, oct. 2010, pp. 55–62.

[6] B. Davis, B. Sanders, A. Khodaverdian, and H. Chen, "I-arm-droid: A rewriting framework for in-app reference monitors for android applications," in *Proceedings of IEEE Mobile Security Technologies (MoST)*, 2012.

[7] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based behavior analysis for android malware detection," in *Proceedings of the Seventh International Conference on Computational Intelligence and Security (CIS)*, dec. 2011, pp. 1011–1015.

[8] P.-M. Chen, H.-Y. Wu, C.-Y. Hsu, W.-H. Liao, and T.-Y. Li, "Logging and analyzing mobile user behaviors," in *Proceedings of International Symposium on Cyber Behavior*, 2012.

[9] J. Lessard and G. C. Kessler, "Android forensics: Simplifying cell phone examinations," *Small Scale Digital Device Forensics Journal*, vol. 4, no. 1, pp. 1–12, 2010.

[10] B. Zhong, L. Niu, and H. Chen, "Design and implement of 3g mobile police system," in *Proceedings of the 2nd International Conference onConsumer Electronics, Communications and Networks (CECNet)*, april 2012, pp. 1752–1755.

[11] R. Gerhards, "The Syslog Protocol," RFC 5424 (Proposed Standard), Internet Engineering Task Force, March 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5424.txt

[12] T. Kobayashi, "Logging system of android," 2010, presented at CELF's Japan Technical Jamboree 34.

Poster 1

Recovering Private Data: A Comparison of Three Methods,
Kalpana Singh, School of Information Technology, Deakin University, Australia.

Recovering Private Data: A Comparison of Three Methods

Kalpana Singh
School of Information Technology
Deakin University, Melbourne, Australia
e-mail: kalpana@deakin.edu.au

***Abstract*—We examine a recent proposal for data privatization by testing it against three well-known attacks. We show that all three attacks successfully retrieve the original data. We compare the strengths of the three attacks. Finally, we indicate how the data privatization method examined can be modified to assist it to withstand these attacks.**

I. INTRODUCTION

Privacy preservation is becoming an increasingly important issue in many data mining applications dealing with sensitive data such as health-care records. A number of data privacy preserving techniques have been developed [1], [2], [3], [4]. There are two general approaches to privacy preserving data mining: the randomization approach [2] and the Secure Multi-party Computation approach [5]. We focus only on the former.

In the randomization approach, random noise is added to the original data, and only the resulting perturbed data are shared [1], [2], [4]. There are two different major randomization methods: the Random Perturbation scheme [4] and the Randomized Response scheme [2]. In the literature, perturbation is of two main types: additive perturbation [1] and multiplicative perturbation [6], where random data (noise) is respectively either added or multiplied with the data. While multiplicative noise techniques can provide a good level of privacy, as shown by Kargupta et al. [7], additive noise perturbation techniques are more effective in maintaining data quality which is a primary requirement of these methods. Noise addition techniques are widely used in areas of PPDM including classification, clustering and association rule mining.

There exists a growing body of literature on additive perturbation techniques which work by adding random noise to the data in such a way that the individual data values are distorted while, at the same time, preserving the underlying distribution properties. Agrawal and Srikant [1] proposed a scheme for PPDM using random perturbation in which a random number is added to the value of each sensitive attribute. It has been shown [1] that this scheme suffers from information loss, but Agrawal and Aggarwal [8] developed a novel reconstruction algorithm called Expectation Maximization algorithm which minimizes the data loss. Du and Zhan [2] also proposed an approach to conduct privacy-preserving decision tree building based on the randomized response technique.

The usefulness of additive noise perturbation techniques in preserving privacy [1] was firstly questioned by Kargupta et al. [4] who showed that attackers can derive a good estimation of the original dataset values from the perturbed dataset using a Spectral Filter that exploits some theoretical properties of random matrices and, as a result, the data privacy can be seriously compromised. Huang et al. [9] further proposed two reconstruction algorithms which are efficient when the added noise is independent of the original data; one is based on Principal Component Analysis [9], the other one chooses Maximum Likelihood Estimation [9] as an estimator.

The purpose of the current paper is to test a specific additive-perturbation method to see how well it withstands three classical data-reconstruction methods. We choose Spectral Filtering (SF) [4], Bayes-Estimated Data Reconstruction (BE-DR) [9] and Multiple Miner attack with Fusion (MDMF) [10]. The particular data-privatization method tested is based on Chebyshev polynomials of the first kind which are explained in detail in Section II. Based on over 60 vectors of dataset, we examine the ability of our three chosen methods to recover the original data. We show that the multiple miner method with fusion gives the best results followed by Spectral Filtering and finally by BE-DR. In addition, for each attack, we are able to specify how to modify the data privatization algorithm to make the data resistant to the attack. Together with Zhong et al. we developed a combination multiplicative additive Chebyshev polynomial method. We tested it using 4500 vectors and determined that it is resistant to all three additive methods described in the above. We also tested it against a classical multiplicative attack [11] method and it resisted.

II. THE CHEBYSHEV POLYNOMIAL PERTURBATION METHOD

An additive perturbation technique based on Chebyshev polynomials [12] is currently being developed in [13]. The authors of the present paper have tested the algorithm using the three attacks mentioned in the previous section. Before describing our methodology, we first provide a short description of Chebyshev polynomials and how they can be used in data privatization.

Zhong et al. [13] propose an additive perturbation algorithm based on Chebyshev polynomials as follows:

A. Chebyshev Data Perturbation Algorithm (CDP)

1) *Data:* The authors of [13] consider numerical data. While the original data is in matrix form, our computer stores it in vector form and gives the output, i.e. perturbed data, in vector form.

2) *Setting parameters:*

m: the number of entries in the original vector; *n:* the degree of the Chebyshev polynomial T_n of the first kind; *l*: a positive integer divisor of n, $l > 1$. The values of m, n and l are all integer.

3) *Data perturbation:*

The initial values of the above parameters for the purposes of producing perturbed data are: $m = (o_{11}, o_{21}, \ldots o_{m1})$, n, l.

a) *Preparation:* Derive T_n.

b) *Division process:* Divide the original data into intervals with length l: Let $D = \frac{m}{l}$ and $T_0 = 1$. Now there are D intervals $t_1, t_2, \ldots t_D$. The first interval t_1 contains the original data $o_{11}, o_{21}, \ldots o_{l1}$; the second contains the next l elements $o_{(l+1)1}, \ldots o_{(2l,1)}$ and so on.

c) *CDP data processing:* In this step we generate noise to add to the original data. If an element o_{i1} of the original data set is in interval j, then we add it to $pt(t_i)_j$, as defined by the equation (2), to obtain the elements of the perturbed data set matrix.

Thus, each element o_{i1} in the j^{th} interval, $(j-1)l+1 \le i \le jl$ will be added to $pt(t_i)_j = T_n\left(-1 + \frac{1}{n} + \frac{2\left(1-\frac{1}{n}\right)j}{|t_i|+1}\right)$ (2)

For each i and each interval j *with* $(j-1)l+1 \le i \le jl$, the perturbed data is therefore,

$$\tilde{p}_{i1} = o_{i1} + pt(t_i)_j, j = 1,2, \ldots, D$$

4) *Restoration: The data can easily be restored by using:* $o_{i1} = \tilde{p}_{i1} - pt(t_i)_j$.

The authors of [13] use multiple records and single attributes during transmission of data for privacy reasons, and they claim that an attacker cannot estimate the original dataset values from the perturbed dataset. In this paper, our aim is to reconstruct the original dataset values from the perturbed dataset and break the privacy method of the paper [13]. For the reconstruction we choose three different attack methods against additive perturbation: these attacks are Spectral Filtering (SF) [7], Bayes-Estimated Data Reconstruction (BE-DR) [10] and the last is Multiple Data Mining and Fusion (MDMF) [11].

III. Summary and Discussion

We have analysed and tested a data privatization method based on Chebyshev polynomials of the first kind which was proposed in [13]. Our reconstruction attacks were Spectral Filtering, Bayes Estimated Data Reconstruction and Multiple Data Mining and Fusion methods. While all three attack methods were successful, a comparison between them found that the MDMF attack method was more successful than the other methods.

We tested our three attack methods by considering three sizes lxD of matrices $l>D$, $l=D$ and $l<D$ for the BE-DR and MDMF attack methods. For the SF attack method, in order to generate the necessary eigenvalues, we only chose $l>D$ and $l=D$ matrix sizes. We found that the square matrices resisted all three attacks better than those with $l>D$ and $l<D$.

In considering the implications of this analysis for the Chebyshev polynomial data privatization method, we can say that if the original dataset does not have a normal distribution, does not have independent original data elements or identically distributed noise data $pt(t_i)_j$, then the SF method will likely fail to recover the private data. This statement can also be made for the BE-DR method. Since it is possible, given a data set, to determine its statistical distribution in advance of privatization, the original data set should first be tested before Chebyshev polynomial privatization is used.

However, the MDMF reconstruction method will recover the data no matter what its distribution. Therefore, the Chebyshev polynomial method needs to be adjusted in order to prevent this attack. One suggestion, proposed in [14] is to compose the additive method with a multiplicative one. The authors demonstrate that this adds security to their data privatization method. We tested this new suggested method on 4500 vectors of dataset and we find that our new method resists all three additive attacks above as well as the multiplicative KS attack method [11].

References

[1] R. Agrawal, and R. Srikant, "Privacy-preserving data mining," In *Proceeding of the ACM SIGMO Conference on Management of Data, Dallas, Texas, ACM Press,* pp. 439–450, May 2000.

[2] W. Du, and Z. Zhan, "Using randomized response techniques for privacy-preserving data mining", *In Proceedings of The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, pp 505 – 510, 2003.

[3] K. Singh, J. Zhong, L. Batten and P. Bertok, "An efficient solution for privacy preserving, secure remote access to sensitive data", *International conference of Advanced Computer Science and Information Technology*, pp 173-191, 2012.

[4] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the privacy preserving properties of random data perturbation techniques,"*In Proc. of the 3rd Int'l Conf. on Data Mining*, pp 99-106, 2003.

[5] W. Du, and M. Atallah, "Secure multi-party computation problems and their applications: a review and open problems,"*In: New security paradigms workshop*, pp 11–20, 2001.

[6] J. J. Kim, and W. E. Winkler,"Multiplicitive noise for masking continuous data," *Technical report, Statistical Research Division, U.S. Bureau of the Census,* April 17 2003.

[7] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar,"Random-data perturbation techniques and privacy-preserving data mining," *Knowledge and Information Systems*, pp 387-414, 2005.

[8] D. Agrawal, and C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," In *Proccedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principlesof Database Systems*, California, USA, pp 247-255, 2001.

[9] Z. Huang, W. Du, and B. Chen, "Deriving private information from randomized data," In *Proceedings of the 2005 ACM SIGMOD Conference*, Baltimroe, MD, pp 37–48, June 2005.

[10] M. Sramka, R. Safavi-Naini, and J. Denzinger, "An Attack on the Privacy of Sanitized Data That Fuses the Outputs of Multiple Data Miners," In PADM, pp 130-137, 2009.

[11] K. Liu, "Multiplicative Data Perturbation for Privacy Preserving Data Mining", 2007, thesis report.

[12] J.C. Mason, D.C. Handscomb, "Chebyshev Polynomials," Prentice Hall, SIAM, 2002.

[13] J. Zhong, V. Mirchandani, and P. Bertok, "Chebyshev Data Perturbation", Working paper.

[14] L. Liu, K. Yang, L. Hu, and L. Li, "Using Noise Addition Method Based on Pre-mining to Protect Healthcare Privacy", *Control Engineering and Applied Informatics (CEAI), Vol.14, No.2,* pp. 58-64,2012.

Poster 2

Privacy and Efficiency in Cloud Database Models,

W.P.E. Priyadarshani,
Wayamba University of Sri Lanka, Sri Lanka.

G.N. Wikramanayake
University of Colombo, Sri Lanka.

Privacy and Efficiency in Cloud Database Models

W.P.E. Priyadarshani and G.N. Wikramanayake
Wayamba University of Sri Lanka, Sri Lanka.
University of Colombo School of Computing, Sri Lanka.

Cloud computing is an increasingly important area in Information Technology. However, despite its long successful adoption, Database as a Service has a number of problems regarding security, privacy and efficiency. This abstract explains efficiency-related factors of current cloud database models and then moves into a study of the impact of data privacy maintenance on efficiency.

There is no reliable evidence that cloud services can provide a high level of data privacy in a fully efficient environment. Not only that, but also there has been little discussion about privacy and efficiency of cloud database models in the research community. Some exceptions to this statement include the authors of (Ali et al., 2010) who consider privacy and efficiency from the view point of organizational decision-makers in the cloud migration process.

This research analyses levels of efficiency in cloud Database models based on a set of factors and metrics described by the authors of (Bojanova et al, 2011).

Analysis on Efficiency of Cloud Database Models

We chose the following cloud service models to which to apply metrics in measuring efficiency:

- Kinetic Cloud
- Oracle Secure Backup (OSB)
- TPA (Third party Auditor)
- Vamsidhar & Venkata Proposed model
- Privacy Manager
- Zhou M, et al proposed model
- NetDB2 Multi-Shares
- DEPSKY Data model (Bessani et al, 2011)

The comparative analysis was done using a Likert Scale and the values chosen were determined by a consideration of the client side view of efficiency. The architecture of each cloud database model was assigned to one of the Likert scale values shown in Table 1. The final level of efficiency is calculated by taking the mean of the four factors mentioned in the infrastructure metrics. In our analysis the levels of efficiency are categorized as low, medium and high and we assume that the three levels are equally weighted.

Hardware cost, Software cost and System Administration cost are caculated by considering three levels of adoption such as: Only storage is outsourced, Data is partially outsourced or DB is fully outsourced. The values are assigned to Real Time Provisioning cost by considering three levels such as: migrating to a completely new platform, upgrading into a new system or database is fully deploying in existing cloud service provider. Figure 2 explaines the final results of the analysis.

Cloud Computing Infrastructure Metrics (High cost =1, Medium cost =2, Low cost = 3)			
Hardware Cost	**Software Cost**	**Real Time Provisioning Cost**	**System Administration Cost**
Cost is HIGH if more traditional infrastructure such as independent servers are used rather than Virtual Servers.	Cost is HIGH if the model uses service management software rather than virtualization software.	Cost is HIGH if elapsed time is high for moving to the new system.	Cost is HIGH if more traditional infrastructure such as independent servers are used rather than Virtual Servers.

Table 1: Cloud Computing Infrastructure Metrics for level of efficiency

Types of cloud Database Models	Level of Efficiency
Kinetic Cloud	Medium
Oracle Secure Backup (OSB)	High
TPA (Third party Auditor)	Medium
Vamsidhar & Venkata Proposed	Low
Privacy Manager	Medium
Zhou M, et al proposed	High
NetDB2 Multi-Shares (Alzain and Pardede, 2011)	Medium
DEPSKY Data model (Bessani et al, 2011)	Medium

Table 2: Level of efficiency of existing database models

Efficiency Varience Against Privacy of Cloud Database Models

The paper on level of user control in cloud databases (Priyadarshani et al., 2012) indicates that there is a direct link between the level of user control in cloud databases and the level of privacy provided, by analyzing eight different models in detail on important factors regarding privacy. In Figure 1, for each of the same models we derive the corresponding level of privacy provided (shown on the x-axis in green) and level of efficiency (shown on the x-axis in blue). From the figure we can see that, in most models, the levels of privacy and efficiency are close.

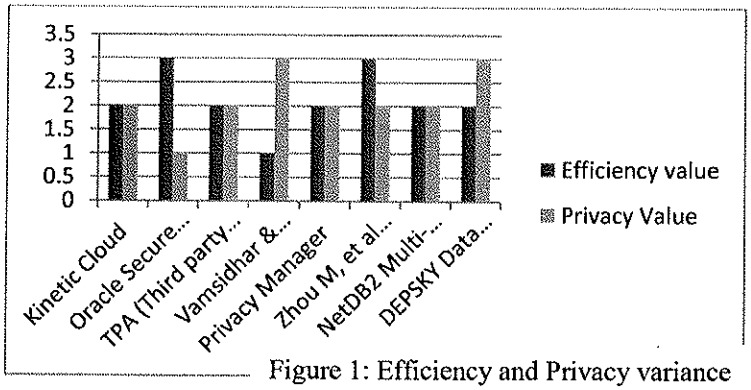

Figure 1: Efficiency and Privacy variance

Conclusion

This study has gone some way towards enhancing our understanding of connections between privacy and efficiency in existing cloud database models. The present study, however, makes several noteworthy contributions to privacy and efficiency anticipated organizational customers for selecting a better cloud DaaS providers in the world today.

References

Khajeh-Hosseini A, Greenwood D. and Sommerville I. (2010) 'Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS', IEEE 3rd International Conference on Cloud Computing.

Bojanova I. and Samba A. (2011), 'Analysis of Cloud Computing Delivery Architecture Models', Workshops of International Conference on Advance Information Networking and Applications, 978-0-7695-4338-3/11, DOI 10.1109/WAINA.2011.74

Priyadarshani E., Wikaramanayake G. N., Batten L. M., Computer Society of Sri Lanka, Enhancement of User Level Controls in Cloud Databases, in 30th National IT Conference, Colombo, Sri Lanka: Jul, 2012.