HUNT RANK SCHEME AND MALWARE DISCOVERYIN ANDROID APP STORE

[1] Dr.P.Chitti Babu, [2] Jaya sree
[1]Professor &Principal , [2]MCA Student
[1,2]MCA Department,
[1,2]Annamacharya PG college of Computer Studies , Rajampet, Y.S.R kadapa, Andhra Pradesh, India

Abstract : Duplicitous behaviors in Google Play, the most popular Android app market, fuel hunt rank misuse and malware production. To identify malware in previous work has focused on app executable and permission analysis. In this work, I introduce FairPlay, a unique system that discovers and controls traces left behind by fraudsters, to detect both malware and apps subjected to hunt rank scheme. FairPlay correlates review activities and uniquely combines detected review relations with linguistic and behavioral signals collected from Google Play app data, in order to identify suspicious apps. FairPlay achieves over 95% accuracy in classifying gold standard datasets of malware, duplicitous and genuine apps. I show that 75% of the identified malware apps engage in hunt rank scheme. FairPlay discovers hundreds of duplicitous apps that currently avoid Google Bouncer's discovery technology. FairPlay also helped the discovery of more than 1,000reviews, reported for 193 apps, that reveal a new type of "coercive" review campaign: users are harassed into writing positive reviews, and install and review other apps.

Index Terms: Introduction, cloud computing, existing work, proposed work

I. INTRODUCTION

The commercial success of Android app markets such as Google Play and the incentive model they offer to popular apps, make them appealing targets for fraudulent and malicious behaviours. Some fraudulent developers deceptively boost the hunt rank and popularity of their apps (e.g., through fake reviews and bogus installation counts), while malicious developers use app markets as a launch pad for their malware. The motivation for such behaviours is impact: app popularity surges translate into financial benefits and expedited malware proliferation. Fraudulent developers frequently exploit crowd sourcing sites (e.g., Freelancer, Fiverr, Best App Promotion) to hire teams of willing workers to commit fraud collectively, emulating realistic, spontaneous activities from unrelated people (i.e., "crowd sourcing"). I call this behaviour "hunt rank scheme". In addition, the efforts of Android markets to identify and remove malware are not always successful. For instance, Google Play uses the bouncer system to remove malware. However, out of the 7, 756 Google Play apps I analysed using virus total, 12% (948) were flagged byat least one anti-virus tool and 2% (150) were identified as malware by at least 10 tools. Previous mobile malware detection work has focused on dynamic analysis of app executables as well as static analysis of code and permissions. However, recent Android malware analysis revealed that malware evolves quickly to bypass anti-virus tools. In this work,I seek to identify both malware and hunt rank scheme subjects in Google Play. An "install job" posting from Freelancer, asking for 2000 installs within 3 days, in an organized way that includes expertise verifications and provides secrecy assurances. Text enlarged for easier reading, I posit that malicious developers resort to hunt rank scheme to boost the impact of their malware. Unlike existing solutions, I build this work on the observation that fraudulent and malicious behaviours leave behind tell-tale signs on app markets. Resource constraints can compel fraudsters to post reviews within short time intervals. Legitimate users affected by malware may report unpleasant experiences in their reviews. Increases in the number of requested permissions from one version to the next, which I will call "permission ramps", may indicate benign to malware transitions. The purpose of this application is to introduce, an efficient technique which is being used for to detect malware and fraudulent apps in Google store. By using this detect the fake reviews and bogus installation counts.

This application has been used to perform following list of operations.

- Detect Malware Apps in Google store
- Detect Apps which are assigned with fake Ratings.

II RELATED WORK

Literature Survey

Google Bouncers Technology

Bouncer quietly and automatically scans apps (both new and previously uploaded ones) and developer accounts in Google Play with its reputation engine and cloud infrastructure. According to Google, bouncer was responsible for a 40% drop in the number of malicious apps in its app store.Bouncer scanning software, developed by Google, is designed to search the Android market for software that could be malicious, the company announced Thursday on its blog. With the success of Android this year, the company says it wants to protect its many users and their devices from harm. Bouncer will scan current and new applications, plus developer accounts. The blog post explained how the service will function.

Working of Bouncer Technology

Once an application is uploaded, the service immediately starts analysing it for known malware, spyware and trojans. It also looks for behaviours that indicate an application might be misbehaving, and compares it against previously analysed apps to detect possible red flags. I actually run every application on Google's cloudinfrastructure and simulate how it will run on an Android device to look for hidden, malicious behaviour. I also analyse new developer accounts to help prevent malicious and repeat-offending developers from coming back.Bouncerwas tested in 2011 and comparing the first half of the year to the second, Google mobile reported a 40% decrease in malicious downloads. Google says from the beginning, Android was designed with security in mind. And, although a company can't prevent malware, it can control the amount of damage those threats can cause with a dynamic security plan.

Malware Removal

Android is designed to prevent malware from modifying the platform or hiding from you, so it can be easily removed if your device is affected. Android market also has the capability of remotely removing malware from your phone or tablet, if required. Google's long been fine-tuning its security features for its various products. Although in the past Google's products have clashed with that of other mobile service providers due to security concerns.

FairPlay

The present system introduces FairPlay, a system to automatically detect malicious and fraudulent apps.

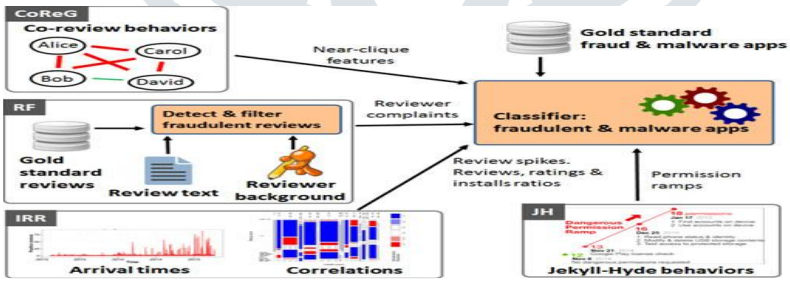


Fig 1:FairPlay System Architecture

The Coreg module identifies suspicious, time related co-review behaviours. The RF module uses linguistic tools to detect suspicious behavioursreported by genuine reviews. The IRR module uses behavioural information to detect suspicious apps. The JH module identifies permission ramps to pinpoint possible Jekyll-Hyde app transitions.

FairPlay Overview

FairPlay organizes the analysis of longitudinal app data intothe following 4modules; illustrated in the Co-ReviewGraph (CoReG) module identifies apps reviewed in a contiguoustime window by groups of users with significantlyoverlapping review histories. The Review Feedback (RF)2module exploits feedback left by genuine reviewers, whilethe Inter Review Relation (IRR) module leverages relationsbetween reviews, ratings and install counts. The Jekyll-Hyde (JH) module monitors app permissions, with a focus on dangerousones, to identify apps that convert from benign tomalware. Each module produces several features that areused to train an app classifier. FairPlay also uses general featuressuch as the app's average rating, total number ofreviews, ratings and installs, for a total of 28 features.

III PROPOSED WORK

This application having following list of modules. They are

- The Co-Review Graph
- Reviewer Feedback
- Inter-Review Relation
- Jekyll-Hyde App Detection

The Co-Review Graph

This module exploits the observation that fraudsters who control many accounts will re-use them across multiple jobs. Its goal is then to detect sub-sets of an app's reviewers that have performed significant common review activities in the past. In the following, I describe the co-review graph concept, formally present the weighted maximal clique enumeration

problem, then introduce an efficient heuristic that leverages natural limitations in the behaviours of fraudsters.

Reviewer Feedback

In this module Reviews written by genuine users of malware and fraudulent apps may describe negative experiences. The RF module exploits this observation through a two-step approach.

- Detect and filter out fraudulent reviews.
- Identify malware and fraud indicative feedback from the remaining reviews.

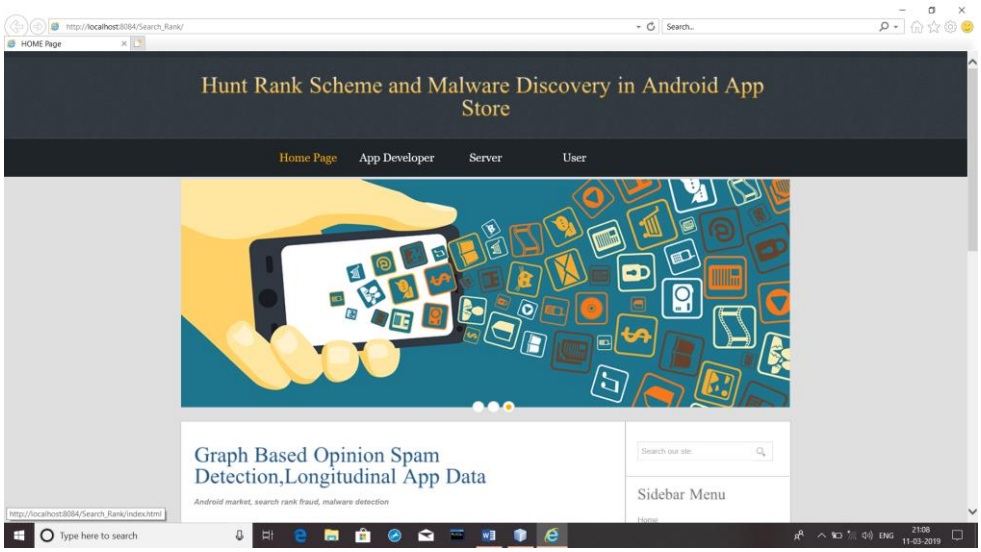
Inter-Review Relation

This module leverages temporal relations between reviews, as well as relations between the review, rating and install counts of apps, to identify suspicious behaviours.

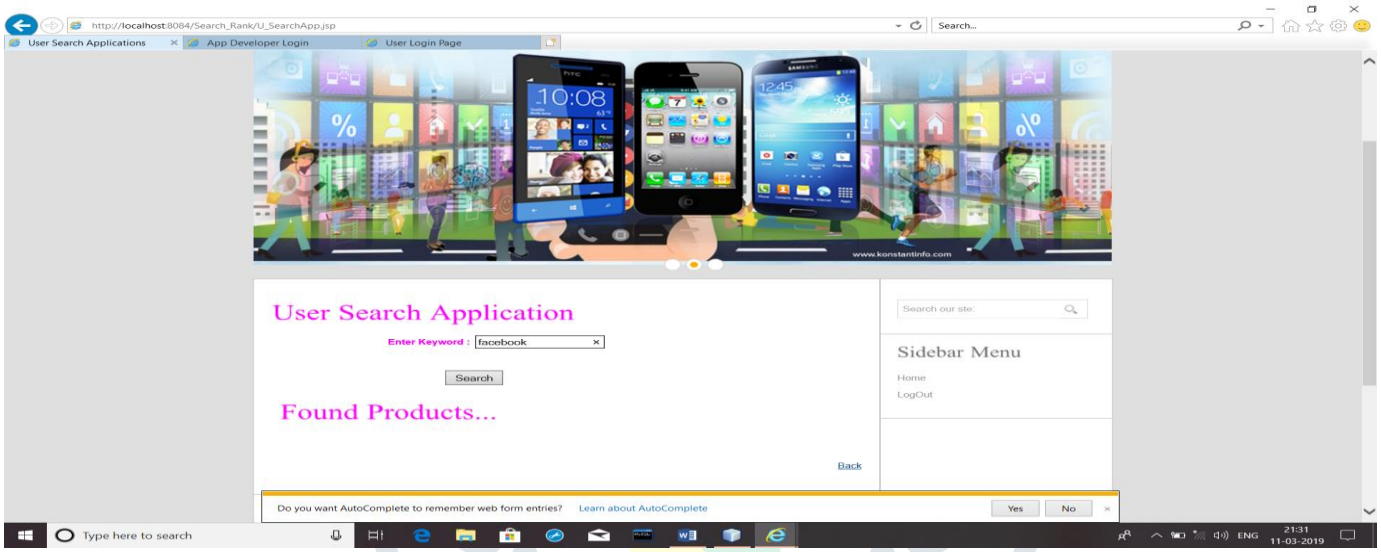
Jekyll-Hyde App Detection

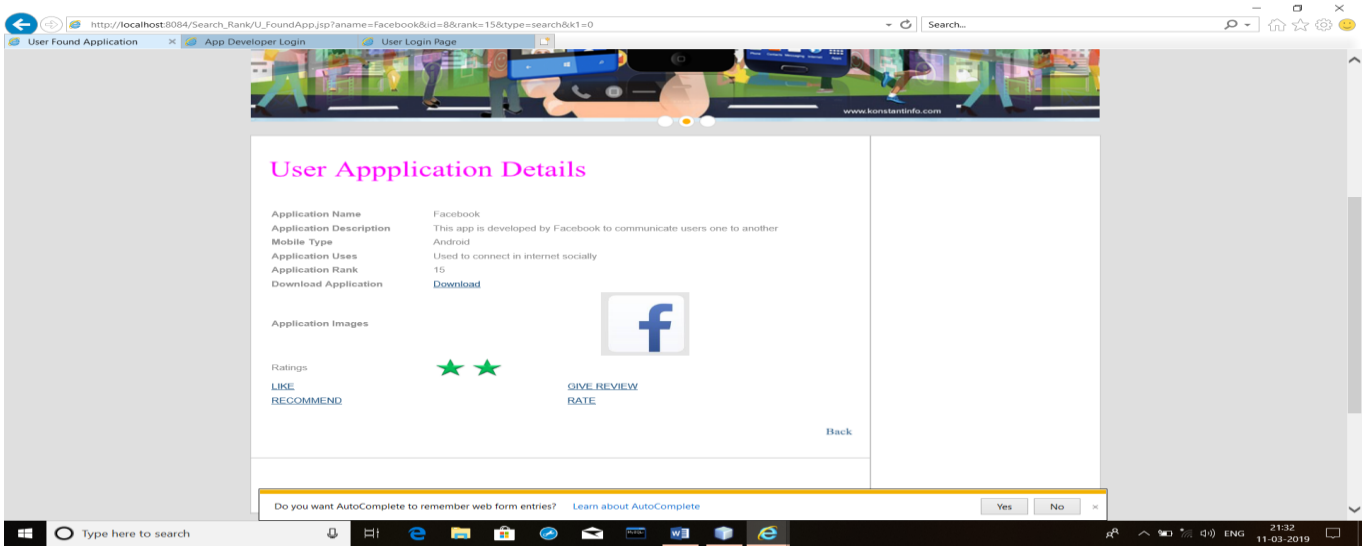
In this module I can find distribution of the total number of permissions requested by malware, fraudulent and legitimate apps. Surprisingly, not only malware and fraudulent apps but also legitimate apps request large numbers of permissions. In addition, Android's API level 22 labels 47 permissions as "dangerous". The most popular dangerous permissions among these apps are "modify or delete the contents of the USB storage", "read phone status and identity", "find accounts on the device", and "access precise location".

IV RESULTS

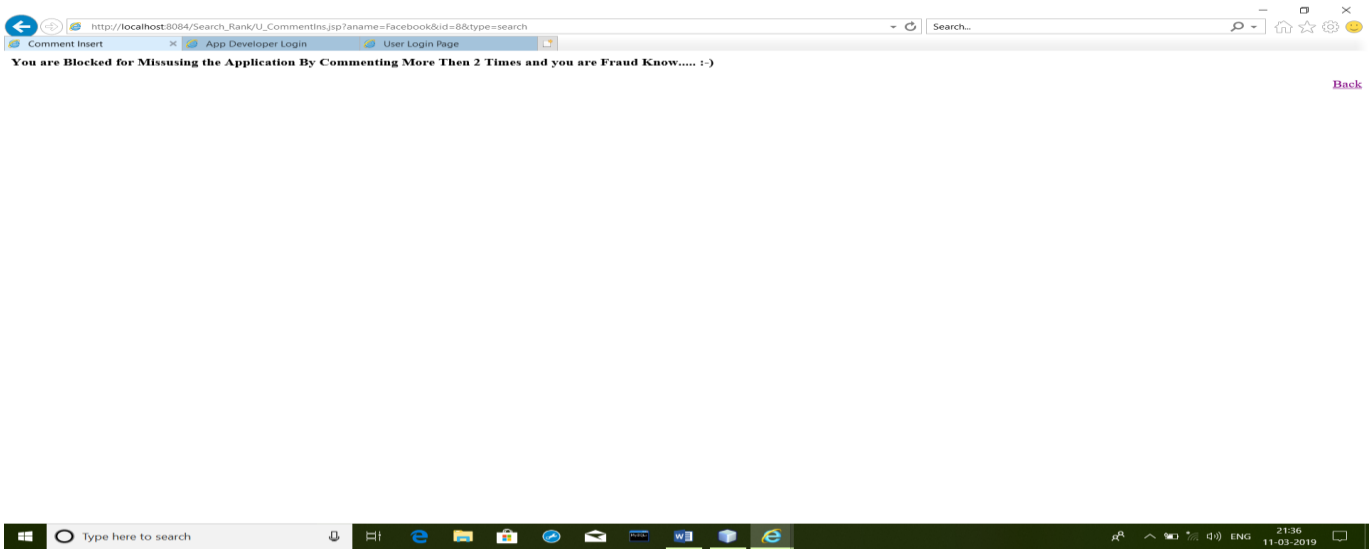


Screen 1 : home page and data owner, user registration details

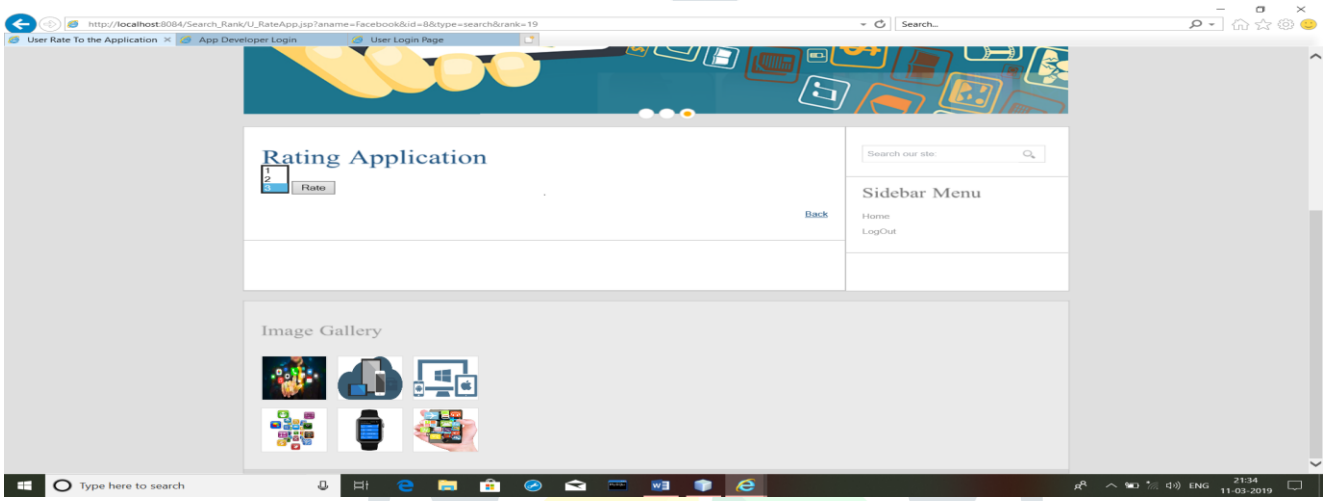




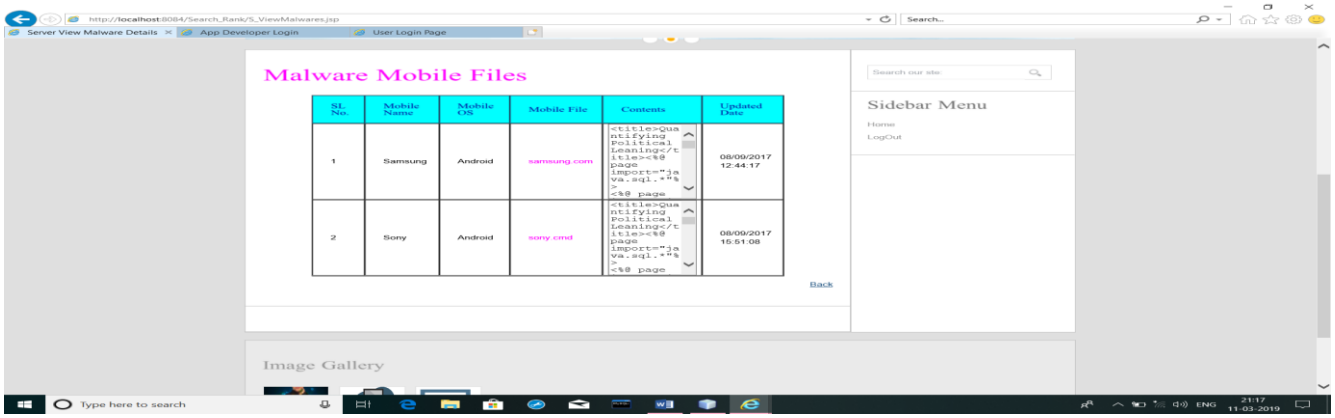
Screen 2 : User Application Details



Screen 3 Fraud user



Screen 4 Rate the Application



Screen 5: malware mobile users

V CONCLUSION AND FUTURE WORK

Finally, I conclude that, it has been a great pleasure for me to work on this "Hunt Rank Scheme and Malware Discovery in Android App Store" project successfullycompleted.Through FairPlay todetect both fraudulent and malware Google Play apps. Our experiments on a newly contributed longitudinal app dataset have shown that a high percentage of malware is involved in search rank fraud on both are accurately identified by FairPlay. By using this FairPlay to successfully detect the fake reviews and bogus installation counts. The FairPlay satisfy the all test cases successfully. This can be use temporal dimensions of review post times to identify suspicious review spikes received by apps.

REFERENCES

[1] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-Scalable Byzantine Fault-Tolerant Services," in ACM Symposium on Operating Systems Principles (SOSP), 2005, pp. 59–74.

[2] M. K. Aguilera, R. Janaki Raman, and L. Xu, "Using Erasure Codes Efficiently for Storage in a Distributed System," in International Conference on Dependable Systems and Networks (DSN), 2005, pp. 336–345.

[3] W. Aiello, M. Bellare, G. D. Crescenzo, and R. Venkatesan, "Security amplification by composition: The case of doublyiterated, ideal ciphers," in Advances in Cryptology (CRYPTO), 1998, pp. 390–407.

[4] C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic, "Robust Data Sharing with Key-value Stores," in ACM SIGACTSIGOPS Symposium on Principles of Distributed Computing (PODC), 2011, pp. 221–222.

[5] A. Beimel, "Secret-sharing schemes: A survey," in International Workshop on Coding and Cryptology (IWCC), 2011, pp. 11–46.

[6] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and Secure Storage in a Cloud-ofclouds," in Sixth Conference on Computer Systems (EuroSys), 2011, pp. 31–46.

[7] G. R. Blakley and C. Meadows, "Security of ramp schemes," in Advances in Cryptology (CRYPTO), 1984, pp. 242–268.

[8] V. Boyko, "On the Security Properties of OAEP as an Allor-nothing Transform," in Advances in Cryptology (CRYPTO), 1999, pp. 503–518.

[9] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable Encryption," in Proceedings of CRYPTO, 1997.

[10] Cavalry, "Encryption Engine Dongle," http://www. cavalrystorage.com/en2010.aspx/.

[11] C. Charnes, J. Pieprzyk, and R. Safavi-Naini, "Conditionally secure secret sharing schemes with disenrollment capability," in ACM Conference on Computer and Communications Security (CCS), 1994, pp. 89–95.

[12] A. Desai, "The security of all-or-nothing encryption: Protecting against exhaustive key search," in Advances in Cryptology (CRYPTO), 2000, pp. 359–375.

[13] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki, "HYDRAstor: a Scalable Secondary Storage," in USENIX Conference on File and Storage Technologies (FAST), 2009, pp. 197–210.

[14] M. Dürmuth and D. M. Freeman, "Deniable encryption with negligible detection probability: An interactive construction," in EUROCRYPT, 2011, pp. 610–626.

[15] EMC, "Transform to a Hybrid Cloud," http://www.emc. com/campaign/global/hybridcloud/index.htm.

[16] IBM, "IBM Hybrid Cloud Solution," http://www-01.ibm. com/software/tivoli/products/hybrid-cloud/.

[17] J. Kilian and P. Rogaway, "How to protect DES against exhaustive key search," in Advances in Cryptology (CRYPTO), 1996, pp. 252–267.