RESEARCH ARTICLE OPEN ACCESS

Search and malware detection in Google play

AUTHOR

S.Priyankaa
Final B.Tech IT
Velalar College of Engineering and Technology
Erode

M.Subhashree
Final B.Tech IT
Velalar College of Engineering and Technology
Erode

S.Akshai
Final B.Tech IT
Velalar College of Engineering and Technology
Erode

Ms. T.Premamala
Associate Professor
Department of IT
Velalar College of Engineering and Technology
Erode

Abstract

The Google Play store had 82 billion app downloads in 2016 and has reached over 2.7 million apps published in 2017. It has been the theme of multiple issues relating to security, in which malicious software has been agreed and uploaded to the store and downloaded by users, with unreliable degrees of severity. Google Play was launched on March 6, 2012, bringing together the Android Market, Google Music, and the Google eBook store under one kind, marking a shift in Google's digital distribution strategy. The services working under the Google Play standard are: Google Play Books, Google Play Games, Google Play Movies & TV, Google Play Music, and Google Play Newsstand. Fair Play employs a relational, linguistic and behavioural approach based on longitudinal app data. Fair Play identifies and exploits a new relationship between malware and search rank fraud. Fair Play also uses general features such as the app's average rating, total number of reviews, ratings and installs, for a total of 28 features.

Keywords
Rating, Fraud and malware, Reviews

INTRODUCTION

DIGITAL MEDIA

Google Play (formerly Android Market) is a digital sharing service operated and developed by Google. Google Play serves as a digital media store, presenting music, magazines, books, movies, and television programs. They can be downloaded using Android device through the Play Store mobile app or by deploying the application to a gadget from the Google Play website. Applications exploiting hardware capabilities of a device can be targeted to users of devices with particular hardware components, such as a motion sensor (for motion-dependent games) or a front-facing camera (for online video calling). Google play has been the theme of multiple issues relating to security, in which malicious software has been agreed and uploaded to the store and downloaded by users, with unreliable degrees of severity.

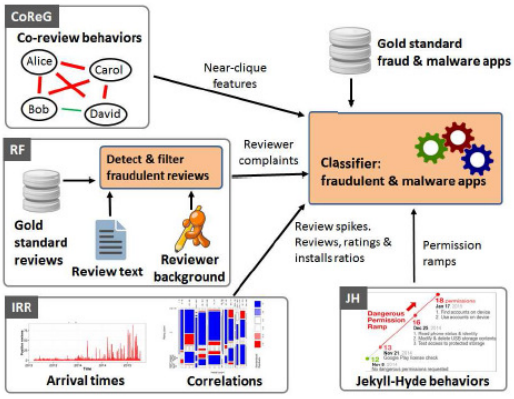
EXISTING SYSTEM MODEL

Focus on the Android app market ecosystem of Google Play. The participants, consisting of users and developers, have Google accounts. Developers create and upload apps, which consist of executables (i.e., "apks"), a set of required permissions, and a description. The app market publishes this information, along with the app's received reviews, ratings, aggregate rating (over both reviews and ratings), install count range (predefined buckets, e.g., 50-100, 100-500), size, version number, price, time of last update, and a list of "similar" apps.

ANDROID MALWARE DETECTION

In recent years, smart phones have experienced unstable growth. Gathered reports suggest that worldwide Smartphone sale in 1/3 quarter of 2011 reached 115 million units, an increase of 42 percent from 0.33 quarter of past years. CNN similarly shows that smart phone shipments have tripled in the precedent three years. Not surprisingly, multiple smart phone platforms are varying for authority on these mobile devices.

SYSTEM ARCHITECTURE



MODULES DESCRIPTION

CO-REVIEW GRAPH MODULE

This module exploits the observation that fraudsters who control many accounts will re-use Its goal is then to detect sub-sets of an app's reviewers that have performed significant common review activities in the past.

Let the co-review graph of an app, be a graph where nodes correspond to user accounts who reviewed the app, and undirected edges have a weight that indicates the number of apps reviewed in common by the edge's endpoint users.

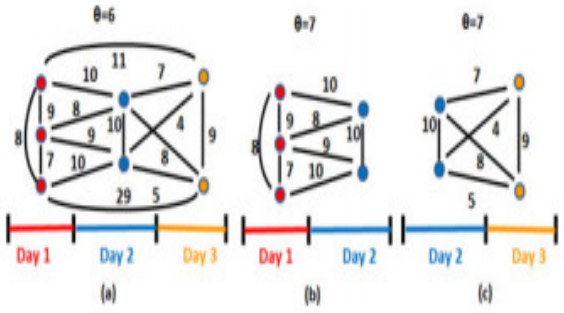


Figure 5.2 Co-Review Graph of an App

In a Weighted Pseudo-Clique Enumeration Problem for a graph $G = V, E$ and a threshold value θ, say that a vertex sub-set U (and its induced sub-graph $G|U|$) is a pseudo-clique of G if its weighted density $\rho = \frac{\sum_{e \epsilon E} w(e)}{\binom{n}{2}}$ [29] exceeds θ; $n = |v|$.[1] U is a maximal pseudo-clique if in addition, no other pseudo-clique of G contains U. The weighted pseudo-clique enumeration problem outputs all the vertex sets of V whose induced subgraphs are weighted pseudo-cliques of G.

PCF SOURCE CODE

```
package searchrankfraud;
import java.awt.Component;
import java.io.File;
import java.io.FileInputStream;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Vector;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.JTextArea;
import  javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableCellRenderer;
public class GooglePlayAdminFrame extends
javax.swing.JFrame {
DBConnection dbn=new DBConnection();
Statement st=dbn.stt;
ArrayList slang1=new ArrayList();
ArrayList slang2=new ArrayList();
ArrayList stop1=new ArrayList();
ArrayList posWd=new ArrayList();
ArrayList negWd=new ArrayList();
ArrayList malwareWords=new ArrayList();
 ArrayList fraudWords=new ArrayList();
 ArrayList benignWords=new ArrayList();
 public GooglePlayAdminFrame() {
 initComponents();
viewDetails();    }
public final void viewDetails()
 {          try
```

```
{               ResultSet rs=st.executeQuery("select *
                 from register where
               Role='"+"Developer"+"'");
                while(rs.next())
  {             String uname=rs.getString(1);
                String pass=rs.getString(2);
                String name=rs.getString(3);
                String mobno=rs.getString(4);
                String adrs=rs.getString(5);
                DefaultTableModel
dm=(DefaultTableModel)jTable1.getModel();
                Vector v=new Vector();
                v.add(uname.trim());
                v.add("**********");
                v.add(name.trim());
                v.add(mobno.trim());
                v.add(adrs.trim());

                dm.addRow(v);           }       }

                catch(Exception e)
      {         e.printStackTrace();
      }
                try
      {           ResultSet rs=st.executeQuery("select *
                 from register where
               Role='"+"User"+"'");
                while(rs.next())
          {   String uname=rs.getString(1);
                String pass=rs.getString(2);
                String name=rs.getString(3);
                String mobno=rs.getString(4);
                String adrs=rs.getString(5);
                 DefaultTableModel
dm=(DefaultTableModel)jTable2.getModel();
                Vector v=new Vector();
                v.add(uname.trim());
                v.add("**********");
                v.add(name.trim());
                v.add(mobno.trim());
                v.add(adrs.trim());
                dm.addRow(v);
          }       }
                catch(Exception e)
          {   e.printStackTrace();
           }   try
          {       ResultSet rs=st.executeQuery("select *
from uploadapp");
                while(rs.next())
            {   String dname=rs.getString(1);
                String maname=rs.getString(2);
                String macategory=rs.getString(3);
                String iname=rs.getString(4);
                String apkfile=rs.getString(5);
                String description=rs.getString(6);
                ImageIcon          image          =          new
ImageIcon(iname.trim());
                DefaultTableModel
dm=(DefaultTableModel)jTable3.getModel();
                 try
               {ResultSet   rs=st.executeQuery("select   *
from UserReviewRating");
                 while(rs.next())
               {String username=rs.getString(1);
                String appname=rs.getString(2);
                String downloadstau=rs.getString(3);
                String review=rs.getString(4);
                String rating=rs.getString(5);
                String time=rs.getString(6);
                DefaultTableModel
dm=(DefaultTableModel)jTable4.getModel();
                Vector v=new Vector();
                v.add(appname.trim());
                v.add(username.trim());
                v.add(downloadstau.trim());
                v.add(review.trim());
                v.add(rating.trim());
                v.add(time.trim());
                v.add("-");
              }
              }
                package searchrankfraud;
                import
de.javasoft.plaf.synthetica.SyntheticaBlueLightLoo
kAndFeel;
                import javax.swing.UIManager;
                public class Main {
                public static void main(String[] args)
              {
                   try       {

                    UIManager.setLookAndFeel(new
SyntheticaBlueLightLookAndFeel());
                    HomePage hp=new HomePage();
                    hp.setTitle("");
```

```
    hp.setVisible(true);
    hp.setResizable(false);          }
    catch (Exception ex)
    {
     //System.out.println(ex);
      }
      }
}
```

REVIEWER FEEDBACK MODULE

we introduced (MAlicious Review Campaign Observer), a novel system that leverages the wealth of spatial, temporal and social information provided by Yelp, to detect venues that are targets of deceptive behaviors. Marco (see figure above) exploits fundamental fraudster limitations to identify suspicious venues. First, Marco identifies venues whose positive review timeline exhibits abnormal review spikes, see the adjacent figure. We prove that if a venue has more than 49 genuine reviews, a successful review campaign for that venue will exceed, during the attack interval, the maximum number of reviews of a uniform review distribution. Second, Marco exploits the observation that a venue that is the target of a review campaign is likely to receive reviews that do not agree with its genuine reviews. In addition, following a successful review campaign, the venue is likely to receive reviews from genuine users that do not agree with the venue's newly engineered rating. Marco defines then the disparity of a review for a venue, to be the divergence of the review's rating from the average rating of the venue at the time of its posting. The aggregate rating disparity score of a venue is then the average rating disparity of all its reviews. This is illustrated in the adjacent figure that plots the evolution in time of the average rating against the ratings of individual reviews received by the ``Azure Nail & Waxing Studio'' (Chicago, IL). The positive reviews (1 day has a spike of 19, 5-star reviews, shown in red in the upper right corner) disagree with the low rated reviews, generating a high average rating disparity.he trained Naive Bayes classifier is used to determine the statements of R that encode positive and negative sentiments. Extract the following features: (i) the percentage of statements in R that encode positive and negative sentiments respectively, and (ii) the rating of R and its percentile among the reviews written by U.

Reviewer Feedback Extraction

Conjecture that (i) since no app is perfect, a “balanced” review that contains both app positive and negative sentiments is more likely to be genuine, and (ii) there should exist a relation between the review’s dominating sentiment and its rating. Thus, after filtering out fraudulent reviews, extract feedback from the remaining reviews. For this, use NLTK to extract 5,106 verbs, 7,260 nouns and 13,128 adjectives from the 97,071 reviews collected from the 613 gold standard apps. Removed non ascii characters and stop words, then applied lemmatization and discarded words that appear at most once. Attempted to use stemming,

extracting the roots of words, however, it performed poorly.

INTER-REVIEW RELATION MODULE

In order to identify the suspicious behaviour this module uses reviews and ratings.

Claim 1. Let R_A denote the average rating. To compensate for the 1 star review, an attacker needs to post at least $\frac{R_A-1}{5-R_A}$ positive reviews.

Proof. Let σ be the sum of all the k reviews.Then, $R_A = \frac{\sigma}{k}$. Let q_r be the number of fraudulent reviews received by A. To compensate for the 1 star review posted at time T, q_r is minimized when all those reviews are 5 stars. Then have that:$R_A = \frac{\sigma}{k} = \frac{\sigma+1+5q_r}{k+1+q_r}$. The numerator denotes the sum of all the ratings received by A and the denominator denotes total number of reviews. Rewriting the last equality, obtain that $q_r = \frac{\sigma-k}{5k-\sigma} = \frac{R_A-1}{5-R_A}$. Divide both the numerator and denominator by k.

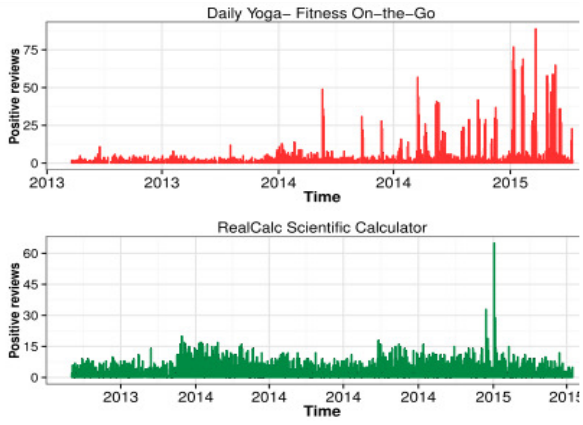


Figure 5.3 Timelines of reviews for two apps.

Use the Pearson's $x2$ test to investigate relationships between the install count and the rating count, as well as between the install count and the average app rating of the 87 K new apps, at the end of the collection interval. Then group the rating count in buckets of the same size as Google Play's install count buckets. The mosaic plot of the relationships between ratings and install counts, $p = 0.0008924$, thus conclude dependence between the rating and install counts. The standardized residuals identifythe cells (rectangles) that contribute the most to the x^2 test. The most significant rating: install ratio is 1:100.

In the mosaic plot of the app install count versus the average app rating, rectangular cells correspond to apps that have a certain install count range ($x\ axis$) and average rating range ($y\ axis$).It shows that few popular apps, i.e., with more than 1,000 installs, have below 3 stars, or above 4.5 stars.

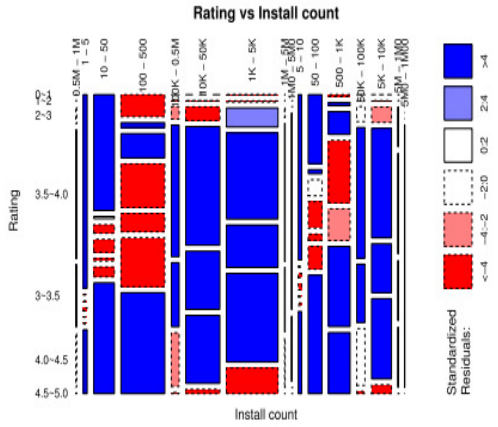


Figure 5.5 Mosaic plots between install count and app rating

Extract temporal features: the number of days with detected spikes and the maximum amplitude of a spike. Also extract the following:

The ratio of installs to ratings as two features, I_1/Rt_1 and I_2/Rt_2

The ratio of installs to reviews, as I_1/Rv_1 and I_2/Rv_2 . (I_1, I_2) and denotes the install count interval of an app, (Rt_1, Rt_2)its rating interval and (Rv_1, Rv_2)its (genuine) review interval.

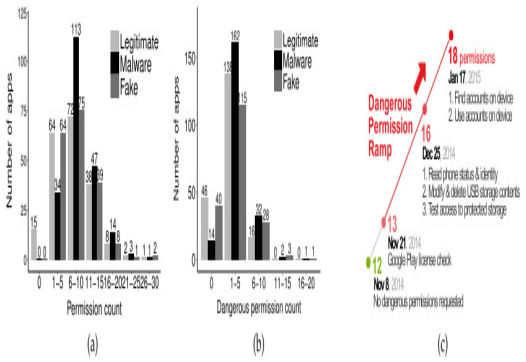


Figure 5.6 (a) Distribution of total permissions (b) Compares the number of dangerous permissions. (c) Dangerous permissions ramp.

Extracts the following information

i. The total number of permissions requested by the app,
ii. Its number of dangerous permissions,
iii. The app's number of dangerous permission ramps, and
iv. Its total number of dangerous permissions added over all the ramps.

CONCLUSION

Fraud is detected based on ranking based evidences Rating based evidences, review based evidences. Here proposed an optimization based aggregation method. The next step is to deploy the android lightweight client on Google's Android market. The system can also act as an early warning system. It is capable of detecting malicious or abnormally behaving applications in the early stages of propagation.

REFERENCES

[1] S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[2] J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[4] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.

[5] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

[6] (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[7] M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[8] *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.

[9] "PDCA12-70 data sheet," Opto Speed SA, Mezzovico, Switzerland.

[10] A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.