Poster: Highlighting Disclosure of Sensitive Data on Android Application with Static Analysis

Takuya Sakashita, Shinpei Ogata, Haruhiko Kaiya and Kenji Kaijiri
Graduate School of Science and Technology, Shinshu University
Wakasato 4-17-1, Nagano, Nagano 380-8553, Japan
{13tm518g, ogata, kaiya, kaijiri}@shinshu-u.ac.jp

1. INTRODUCTION

In this paper, a method for highlighting disclosure of sensitive data on an android application is proposed. The sensitive data for example are e-mail addresses, phone number, etc. In our method, the source code of the application is analyzed statically for the highlighting. We aim to improve insufficient information of the existing permission system so that android users can determine whether the application is malware more correctly. Three types of information are highlighted. Firstly, it's where sensitive data are sent. Secondly, it's the type of the sensitive data such as phone number. Finally, it's whether android permissions for the sending of the sensitive data are related on the actual program. CASE tool for supporting the static analysis and highlighting is also created.

2. PROBLEM

2.1 Malware

Recently, various android malware has appeared in Google play. The malware causes serious problems to users. A kind of the malware discloses sensitive data for abusing the data. Other kind of the malware sends premium SMS for stealing user's money without user's permission. In contrast, some applications send sensitive data to the outside for online back-up of addresses. This sending is often reasonable. Precise and automatic determination of whether an application is malware is impossible. Such behavior of the application however does not be informed to the users.

2.2 Lack of Information to Suspect Malware

A permission system is provided for protecting privacy and security of android users in the android OS. For example, the permissions are "read phone state and identity," "full internet access," etc. A user allows an application to use such permissions when he/she installs the application. There are however three problems on how to show permission information to the user. Firstly, what an application does with permissions is not shown. The user cannot therefore have the opportunity to know why the application needs the permissions without users' informal articles of review sites. Secondly, the user cannot confirm what sensitive data the application uses in detail. The permission information is abstracted excessively as "read phone state and identity." Thirdly, it is unclear whether sensitive data are sent to the outside. The user should suspect disclosure of sensitive data if phone number read with "read phone state and identity" is sent to a URL with "full internet access" as Fig.3. Whether such permissions are related on the actual program however is not shown.

2.3 Lack of Android User's Knowledge

Various users from novices to experts of the android exist. The novices might overlook malware even if installation information mentioned above is detailed. A method to share application information among various users therefore is needed. The fact that over 70 percentages of 308 users read the reviews of the application which he/she installs [1] bears out our opinion.

3. METHOD

We try to settle the problems in section 2 with the following solutions. Firstly, the types of sensitive data which are read by using the permission mentioned-above are analyzed. The relation between sensitive data and the outside such as URL where sensitive data are sent also is analyzed. We also propose a method for highlighting analyzed information as Web pages so that various users can communicate each other for determining malware more correctly before its installation.

3.1 Overview

Fig. 1 shows the overview of our highlighting method. Actions and data are explained in the following sections.

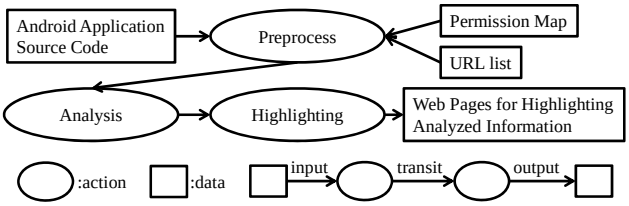

Figure 1. The overview of our highlighting method.

We propose one artifact and one process newly. The artifact is "Web pages for highlighting analyzed information." The process is how to highlight disclosure of sensitive data with static analysis.

3.2 Analyzing Disclosure of Sensitive Data

Table 1. The elements to be mapped by static analysis

ID	Source code elements and permission information to be mapped	
1	Sensitive data type (SDT)	Permission for getting sensitive data
2	SDT	API call to get sensitive data
3	API call for sending data to the outside	Permission needed for sending data to the outside
4	Variable (field, argument and local variable)	Value read with API call
5	Variable	Variable
6	Variable	Method invoked
7	Value read with API call	SDT identified by API call

The inputs to our method are source code, the permission map [2] and URL lists. The source code is the target of analysis. It is not

very hard to get source code of android applications from Google play because of recent decompiling tools such as apk-tools, dex2jar and JD for Java. We therefore focus on the source code analysis for highlighting the analysis result. The permission map for grasping the mapping between permissions and API calls is used in our analysis. We also use the Abstract Syntax Tree (AST) parser in Eclipse for realizing static analysis. Table 1 shows the elements to be mapped in the analysis for identifying the relations among permissions, sensitive data and the outside.

The relations among permissions, SDTs and API calls can be analyzed in the "Preprocess" action because these relations are already clear. The SDTs for example are "Phone number of my own," etc. The SDT is taken into account if an API call is mapped to the SDT directly. For example, "android.telephony.Telephony Manager.getLine1Number()" is mapped to "Phone number of my own." We name the SDT by referring to android API reference. These relations are corresponding to 1, 2 and 3 in Table 1. Then, the relations among variables, SDTs and values can be determined in the "Analysis" action because the relations can be identified on the basis of the flow and calling relationship of methods. These relations are corresponding to 4, 5, 6 and 7 in Table 1. Some API calls receive/send data from/to the outside with permissions. The Web page in Fig. 3 is generated when such API call was analyzed.

Fig. 2 shows an image of the analysis for mapping between the permissions of "full internet access" and "read phone state and identity." This source code is a simplified example of malware which sends phone number to the outside. The boxed elements in Fig. 2 are mapped by the analysis. The types and/or values of each boxed element are shown as the strings with under line in Fig. 2. The bottom of Fig. 2 shows the result of the mapping. The encircled elements depict the type or value of the elements in Table 1. The lines depict the mapping between the encircled elements. The relations of the permissions is identified by tracing from "execute(…)" to "getLine1Number()." The URL of "http://..." is also identified by tracing from "httpPost" to the literal of "http://..." via "uri", but Fig. 2 does not depict it.

```
public class MainActivity extends Activity {
    private List<NameValuePair> post_params = new ArrayList<NameValuePair>();
    public void onCreate(Bundle savedInstanceState) {
        String tel =((TelephonyManager)getSystemService("phone")).getLine1Number();
               Variable   API call, Permission:READ_PHONE_STATE, SDT:Phone number of my own
        exec_post(tel);
    }
    private void exec_post(String tel) {
        this.post_params.add(new BasicNameValuePair("tel", tel));
                 Variable
        URI uri = new URI("http://example.com/script/sample.php");
        HttpPost httpPost = new HttpPost(uri);
        httpPost.setEntity(new UrlEncodedFormEntity(this.post_params, "UTF-8"));
          Variable
        HttpClient httpClient = new DefaultHttpClient();
        httpClient.execute(httpPost, this.response_handler);
    }                  API call, Permission:INTERNET
}
```

Figure 2. The analysis of the relation of the permissions.

3.3 Highlighting Analyzed Information

We finally highlight the analyzed information as Web pages so that users can understand disclosure of sensitive data before the application installation. Fig. 3 shows the analysis result in Fig. 2. At the top of Fig. 3, The "Possibility of Disclosure" shows the SDT. Next, the relation of the "Permissions" is shown. The "Purpose" is identified the purpose of sending sensitive data on the basis of URL lists. The purpose for example is unknown or advertisement. The "URL" means the outside where the sensitive data are sent. The "Request Parameters" is also shown. Most of this information can be hidden by selecting SDT which is a toggle button. A user can easily grasp the sensitive data an application handles even if a lot of information is highlighted.

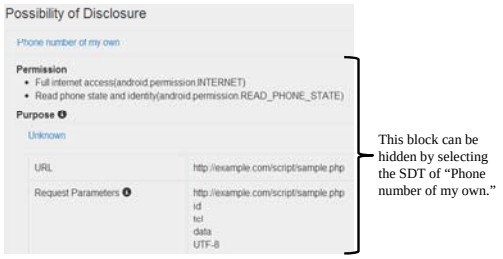

Figure 3. A part of the Web page as the result of highlighting.

4. DISCUSSION

The highlights of permission relations, disclosure possibility and URL may make the user determine is malware correctly. On the other hand, our analysis may make the users misunderstand malware if the accuracy of analysis is low. We therefore plan to evaluate the accuracy of our analysis in the future.

5. RELATED WORK

Enck et al.[3] have realized TaintDroid for monitoring disclosure of sensitive data. It however is difficult to bridge the gap of the knowledge between novices and experts because the disclosure is notified to individual users. Our research adopts complementary approach of such method by highlighting the disclosure as Web pages. Secroid [4] is a service to estimate the risk of individual applications before the installation. For example, the risk becomes high if an application might send phone number to the outside. The user therefore can understand the risk intuitively. In contrast, the user might recognize valid applications as malware. We adopt neutral highlighting without determining the risk.

6. CONCLUSION

In this paper, a method for highlighting disclosure of sensitive data by analyzing an android application is proposed. As future work, the effectiveness of our method compared with the existing permission system is evaluated through providing the highlighted information to end users. The accuracy of our analysis also is evaluated. How to introduce dynamic analysis for enhancing the sufficiency and accuracy of the analysis is considered.

7. REFERENCES

[1] Felt, A.P., et al., 2012 Android Permissions: User Attention, Comprehension, and Behavior. In Proc. of SOUPS 2012 (Washington, DC, USA, July 11-13, 2012).

[2] Felt, A.P., et al., 2011 Android Permissions Demystified. In Proc. of CCS'11(Chicago, Illinois, USA, October 17–21, 2011).

[3] Enck, W., et al., TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In Proc. of the 9th USENIX Symposium on OSDI'10 (Vancouver, BC, Canada, Oct. 4-6, 2010).

[4] Secroid, http://secroid.com/, accessed at 28 May 2013.