2023 International Seminar on Computer Science and Engineering Technology (SCSET) | 979-8-3503-0147-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/SCSET58950.2023.00062

An APP Privacy Information Leakage Detection Method Based on Deep Learning

Nishui Cai
Security Technology Research Institute,
China Telecom Research Institute,
Shanghai, China
cainishui@chinatelecom.cn

Tianting Chen
Security Technology Research Institute,
China Telecom Research Institute,
Shanghai, China
chentt8@chinatelecom.cn

Lei Shen
Security Technology Research Institute,
China Telecom Research Institute,
Shanghai, China
shenlei8@chinatelecom.cn

***Abstract*-With the increasing number and types of APP vulnerabilities, the detection technology and methods need to be enriched and personalized according to different types of security vulnerabilities. Therefore, a single detection technology can no longer meet the needs of business security diversity. First of all, the new detection method needs to clarify the relevant features of APP business security; Secondly, the new detection method needs to re-adapt the features related to APP business security; Thirdly, the new detection method needs to be trained and applied according to different AI algorithms. In view of this, we designed an APP privacy information leakage detection scheme based on deep learning. This scheme specifically selects business security-related features for the type of privacy information leakage vulnerability of APP, and then performs feature processing and adaptation to become the input parameters of CNN network algorithm. Finally, we train and call the CNN network algorithm. We selected the APP of the Telecom Tianyi Space App Store for experiment to evaluate the effectiveness of our APP privacy information leakage detection system based on CNN network. The experimental results show that the detection accuracy of our proposed detection system has achieved the desired effect.**

Keywords-APP detection, privacy information leakage, deep learning, CNN network, ACTIVITY vulnerability, AI algorithm, feature processing

I. INTRODUCTION

APP privacy protection is getting more and more attention [1][2][3]. APP vulnerability detection technology continues to develop in the direction of APP business domain segmentation, technology personalization and AI technology combination[4][5][6], and special security subject detection, such as anti-pattern detection, illegal replication and cloning detection[8][9][10]15], aggressive push detection[7], user analysis and comment detection based on topic modeling or clustering technology, ICC (inter-component communication) vulnerability detection, application collusion detection based on machine learning Detection of malware in app stores based on life measurement.

Based on AI detection technology, we need to solve the problem of feature processing adaptation, how to effectively select and extract relevant features, and then convert them into input parameters for AI algorithm adaptation, and use AI technology to detect and find security vulnerabilities[4][5][6].

The new detection system faces three problems. First, the new detection system needs to establish a security feature library in the business domain; Secondly, the security features of the new detection system need to be re-adapted; Thirdly, the new detection system needs to be trained and applied according to different AI algorithms[11][12][13][14].

We have designed a scheme of APP privacy information leakage vulnerability detection system based on CNN network. This scheme is specifically aimed at the privacy information leakage characteristics of APP ACTIVITY vulnerability, selects feature parameters and carries out feature processing to adapt the input parameters of CNN network algorithm, and finally trains and calls the CNN network algorithm. We selected the APP of Telecom Tianyi Space App Store to conduct experiments to evaluate the effectiveness of our APP ACTIVITY vulnerability detection system based on CNN network. The experimental results show that the detection accuracy of this method has the expected effect.

II. BASIC METHODS OF APP VULNERABILITY DETECTION AND PRIVACY INFORMATION LEAKAGE

A. Basic Methods of APP Vulnerability Detection

Generally, development-related features are extracted from three different levels, namely (1) binary-level features are extracted from the .dex file, (2) resource-level features are extracted from the decompiled resource file and the overall package structure, and (3) configuration-level features are extracted from the Android Manifest file.(See Figure 1)

1. APP code-level features

Although after compiling from the source code, we focus on the characteristics related to privacy information disclosure and extract three code-level features from the .dex bytecode, including array features, opcode features and construction features.

2. APP configuration level features

Each application declares a manifest file, which describes the basic information about the application, including its package name, application ID, application components, requested permissions, device compatibility configuration, etc. It is embedded in the .apk file in an encoded format. After decompiling the application, we can obtain the decoded AndroidManifest.xml file and extract the function from it.

3. APP resource-level features

We further seek to extract features from decompiled resource files. After decompiling the application, we will get three additional directories, namely "assets", "lib" and "res". All necessary resource files (such as icons, pictures, sounds, .xml files, compressed files, native library files, language files, etc.) are placed in these three corresponding directories.

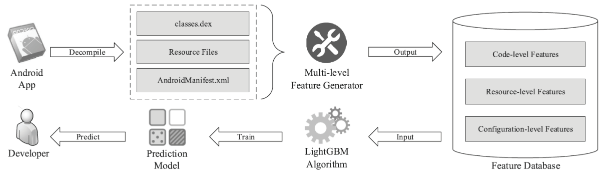

Figure 1. Basic flow diagram of APP detection.

B. Privacy Information Leakage of APP

This article is a special study on the activity security risk most related to privacy information leakage in Android applications.

Activity is one of the four components defined by Android and the most important component used most. Activity is used to provide an interface for users to interact with applications. Its life cycle is shown in the following figure: (See Figure 2)

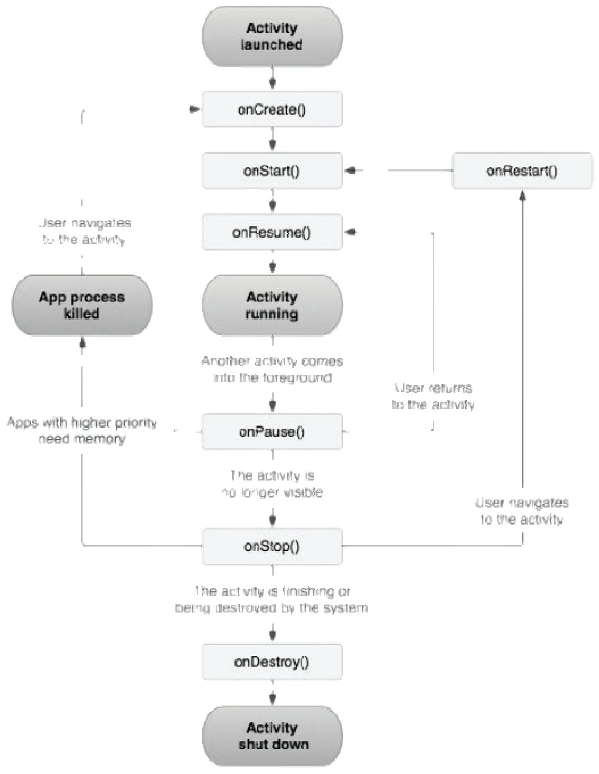
Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 2. Life cycle of activity components.

OnCreate(): called when creating an activity

OnStart(): called when the activity interface becomes visible to the user

OnResume(): called when the activity interface gets the focus (the interface button can be clicked and the text box can be entered)

OnPause(): Call when the activity loses focus (the button cannot be clicked and the text box cannot be entered)

OnStop(): called when the activity becomes invisible

OnDestory(): called when the activity is destroyed

OnRestart (): Called when the activity starts again

If the activity component sets the export permission, the component can be directly called by other external components, which may lead to the risk of disclosing private data or application crash. Activity is called by a malicious application, which may have the following threat description: modify the status or data of the program; The called activity may return private information to malicious applications, resulting in data disclosure; It may crash the application and cause denial of service and other vulnerabilities.

The current static detection method is mainly based on extracting the AndroidManifest.xml file and scanning all activity components. Check the EXPORTED and other properties of components to detect whether there are components that can be exported. If there are activities that can be exported, there are risks. This method has a high false alarm rate. If there are activities that need to be exported because of business needs, this detection method has great limitations. When a component needs to share data or interact with other apps and needs to have an exported activity, it needs to further control its permissions or verify the parameters passed by the int, or when the exported activity comes from a third-party SDK, it is restricted by the function calls of the third-party SDK, and it should not be roughly considered that the APP has activity risks. Therefore, based on the original static detection method, this paper mainly considers to improve the accuracy of detection by adding permissions and the extraction and analysis of the content API and parameter features.

III. APP PRIVACY SENSITIVE INFORMATION LEAKAGE DETECTION METHOD BASED ON CNN NETWORK

A. APP ACTIVITY feature composition

1. Code characteristics

1) The code features of this vulnerability

a)The code exists in android: exported="true"

b) The action tag of <intent filter> is set.

Examples are as follows:

```
● <activity
      android: exported="true"
      android: name=".other.ComponentActivity">
   </activity>
● <activity
     android: name=".other.ComponentActivity">
     <intent-filter>
        <action android:
name="android.intent.action.VIEW"/>
     </intent-filter>
       </activity>
```

For an activity, if there is no intent filter, exported is false by default; When there is an intent filter, exported is true by default. Therefore, the activity components configured with the action value of intent-filter or with android: exported="true" can be exported.

2) Specific steps

Decompile APK. You can use Apktool, jadx and other decompilation tools to obtain the AndroidManifest.xml list file of APK, traverse the AndroidManifest.xml file, find the action value configured with the intent-filter, or display the one set with android: exported="true". Get the activities that can be exported by the APK to be detected.

3) Feature processing

The attribute value of android: name of the activity that can be exported by the APK to be detected is processed by word segmentation. First, use "." for word segmentation, and divide a string into several segments. For example, in the above example, divide ". other. ComponentActivity" into "", "other", "ComponentActivity", remove the empty after segmentation, and then detect whether there are capital characters in the separated string, and divide "ComponentActivity" into "Component", "Activity" according to the characteristics of camel naming method. Use word embedding to convert the segmented string words into feature vectors. A word corresponds to a 1 * N-dimensional feature vector.

2. API features

Intent is the "medium" for communication between different components, which provides information about the mutual invocation of components. Intent plays the role of component value transfer between different activities. Generally, it can be used to start activities, start services and send broadcasts.

The APIs related to Activity and Intent are as follows

Intent: Intent

Intent (Context packageContext, Class<?>cls): used to create display intention objects

Intent (String action): used to create an implicit intention object

PutExtra (String name, Xxx value): save additional data

Xxx getXxxExtra (String name): Get additional data

SetData (Uri data): uri data with a specific format

Activity: Activity

StartActivity (Intent intent): generally start an activity

StartActivityForResult (int reqCode, Intent int): Start activity with callback

OnActivityResult (int reqCode, int resultCode, Intent data): callback method

SetResult (int resultCode, Intent data): Set the result to be returned

Finish (): End the current activity

Getlntent(): get the intention to start the activity

According to the android: name attribute value of the activity that can be exported by the APK to be detected, traverse the decompiled program code, obtain the API related to the exported activity, and obtain the function parameters. As shown in the following figure,

Example: The code is the relevant description of the LoginHandleActivity, and the int passes the information of the account "account" and password "password" from the outside.

```
public class LoginHandleActivity extends
AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_login_handle);
//Prompt of successful login
TextView loginsuccess=findViewById(R.id.loginsuccess);
//Get the value passed
Intent intent=getIntent();
String account=intent.getStringExtra("account");
String password=intent.getStringExtra("password");
//Set prompt content
Loginsuccess.setText ("Congratulations to
user:"+account+"Password:"+password+"Login
succeeded!");

//Print prompt message

Toast.makeText(this,""+loginsuccess.getText(),
Toast.LENGTH_ SHORT).show();
```

This step mainly extracts the function parameters of the API related to the export activity. Here, we first determine the parameter type, int and float types are directly converted to binary. The string type first determines whether it is a URL. If so, we use regular matching to obtain the domain name information. Then, the domain name and non-URL strings are segmented according to "." or "_", and word embedding is used to convert the segmented string words into feature vectors.

3. Permission information

1) Feature acquisition

Traverse the androidmanifest.xml file to obtain the permission information of the APK application. The example is as follows:

```
<uses-permission
android:name="android.permission.READ_CONTACTS"/>
<uses-permission
android:name="android.permission.READ_EXTERNAL_S
TORAGE"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_S
TORAGE"/>
```

The above example applies for android.permission.READ_ CONTACTS 、 android.permission.READ_ EXTERNAL_ STORAGE 、 android.permission.WRITE_ EXTERNAL_ STORAGE。

2) Feature processing

The following is a description of all application permissions for Android, 142 items in total.

Onehot code is used here. 142 digits correspond to 142 permissions.

For example, if one hot code is used in the above example, 000000000 1... 11

B. System Architecture

Data set: This paper uses the drozer tool, which is a security testing framework under the Android system. It can be used to analyze common security problems of applications, and also to scan security vulnerabilities of some systems. Download the APP in the application mall, and verify the risk APP with and without activity through the drozer tool as the training set and test set samples of this paper. The method proposed in this paper adopts supervised learning.

System architecture diagram: (See Figure 3)

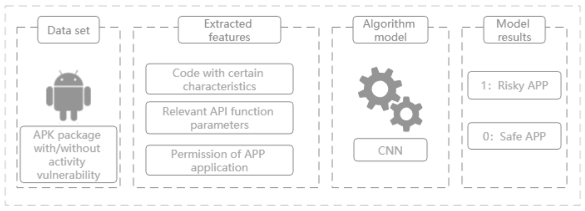
Data set
APK package with/without activity vulnerability
Extracted features
Code with certain characteristics
Relevant API function parameters
Permission of APP application
Algorithm model
CNN
Model results
1: Risky APP
0: Safe APP

Figure 3. System architecture diagram.

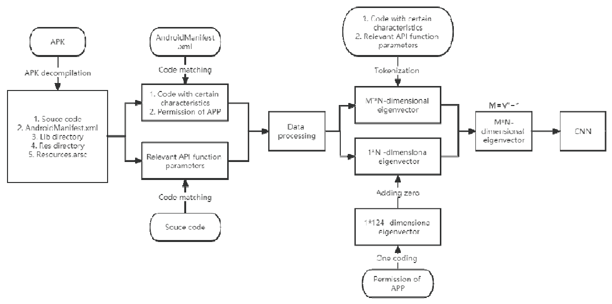

Figure 4. Method flow chart.

The method in this paper, as shown in Figure 4, is mainly aimed at Android APP. The decompiled code, resource file, configuration file, etc. are obtained by decompiling the APK file. And select the attribute value of android: name of the exportable activity and the permission information of APK application from the configuration file androidmanifest.xml using the code feature matching method. At the same time, the code feature matching method is used to extract the function parameter values of the API related to the export activity from the decompiled code.

The attribute value of android: name of the exportable activity and the function parameter value of the exported activity-related API need to be pre-processed to convert the string into a feature vector. Here, word segmentation is performed first, and word embedding is used to convert the segmented string words into feature vectors. Suppose that a word corresponds to an N-dimensional vector, and the attribute value of the android: name of the exported activity and the function parameter value of the exported activity-related API are M 'after word segmentation, that is, a feature vector 1 of M' * N dimension is finally generated.

Among them, the extracted permission information is encoded with one hot. There are 124 Android permission sets collected in this paper. After the extracted permission information is encoded with one hot, a 1 * 124 feature vector 2 will be generated. Perform zero filling operation on eigenvector 2 to generate a 1 * N eigenvector 2 '.

Finally, the feature vector 1 of M '* N dimension and the feature vector 2' of 1 * N dimension are combined to generate a feature vector of M * N dimension. Finally, the M * N dimension feature vector is input into the CNN network model.

IV. EXPERIMENTS

In this paper, CNN is used to build a two-class model, and the output result is 1 with risk or 0 without risk. (See Figure 5)

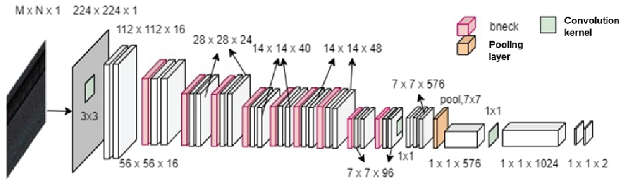

Figure 5. CNN architecture.

The following figure shows the specific layers of CNN network. (See Table 1)

Table 1. The specific layers of CNN network.

Input	Operator	exp size	#out	SE	NL	s
2242×1	con2d，3×3	-	16	-	HS	2
1122×16	bneck，3×3	16	16	√	RE	2
562×16	bneck，3×3	72	24	-	RE	2
282×24	bneck，3×3	88	24	-	RE	1
282×24	bneck，5×5	96	40	√	HS	2
142×40	bneck，5×5	240	40	√	HS	1
142×40	bneck，5×5	120	48	√	HS	1
142×48	bneck，5×5	144	48	√	HS	1
142×48	bneck，5×5	288	96	√	HS	2
72×96	bneck，5×5	576	96	√	HS	1
72×96	con2d，1×1	-	576	√	HS	1
72×576	Pool，7×7	-	-	-	-	1
12×576	con2d，1×1，NBN	-	1024	-	HS	1
12×1024	con2d，1×1，NBN	-	k	-	-	1

Loss function construction

Focal loss, as seen in Equation (1), is used in this paper, and its expression is as follows.

$$Loss_{fl} = \begin{cases} -\left(1-y'\right)^{\gamma}\log y' & y=1 \\ -y'^{\gamma}\log\left(1-y'\right) & y=0 \end{cases} \quad (1)$$

For the two-classification model, the cross-entropy loss function is mostly selected in the selection of the loss function. In this patent, the loss function uses the focal loss and sets y to be greater than 0. Its advantage is that it can reduce the loss of easily classified samples. It makes us pay more attention to the difficult and misclassified samples, as shown in Figure 6.

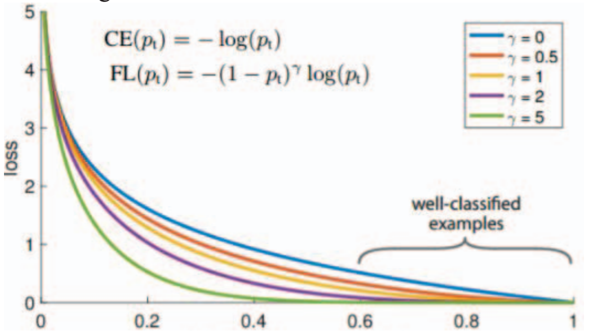

Figure 6. Probability of ground truth class.

In the APP training model of this method, the effect is the best when y=2.

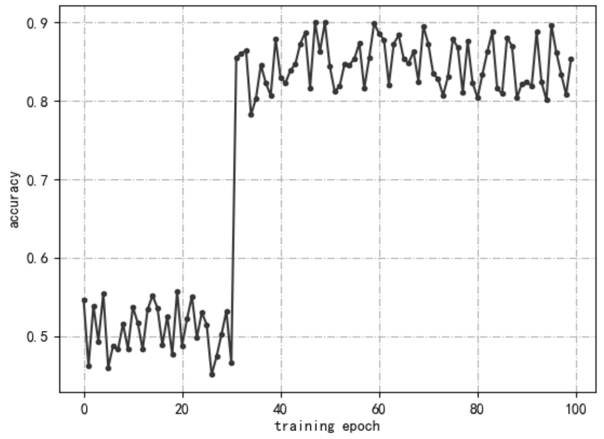

Figure 7. In the introduction γ and α Previous training accuracy.

In the introduction γ and α After that, the training speed became faster and the accuracy rate of 100 rounds of training increased from 86.95% , as shown in Figure 7, to 95.98%, as shown in Figure 8.

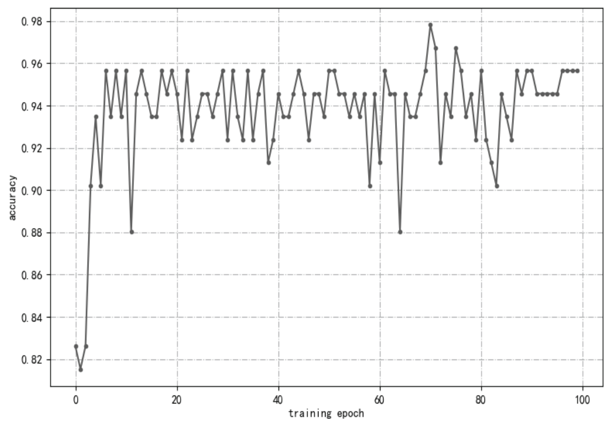

Figure 8. In the introduction γ and α Training accuracy after.

V. CONCLUSION

This paper proposes a scheme of APP ACTIVITY privacy information disclosure vulnerability detection system based on CNN network, which specifically aims at the privacy information disclosure characteristics of APP ACTIVITY vulnerability, selects feature parameters and performs feature processing to adapt the input parameters of CNN network algorithm, and finally trains and calls the CNN network algorithm. We selected the APP of Telecom Tianyi Space App Store to conduct experiments to evaluate the effectiveness of our APP ACTIVITY vulnerability detection system based on CNN network. The experimental results show that the detection accuracy of our proposed system has achieved the desired effect.

In the future, we plan to focus on expanding our research work from two aspects: on the one hand, we will continue to carry out the special test of privacy information leakage detection of the display network APP, and continue to accumulate relevant data; On the other hand, it will continue to optimize and improve the in-depth learning algorithm of CNN network to meet the requirements of better detection accuracy.

ACKNOWLEDGMENTS

This work is financially supported by the national key research and development plan project "Research on the protection of personal rights and interests of private data" (project number: 2021YFB3101300), subject 5 "Research on key technologies for the protection of personal rights and interests" (project number: 2021YFB3101305), At the same time, it has also been supported by the scientific research project "Research and development of key technologies for data security and intelligence" (project number: T-2022-33) of China Telecom Corporation, "Research and prototype development of data flow risk identification technology based on high-level AI" (project number: 22HQJ6YF0080-003).

REFERENCES

[1] Imad Ali Hassoon; Nicolae Tapus; Anwar Chitheer Jasim. Enhance privacy in big data and cloud via diff-anonym algorithm. 2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet). 2017.

[2] Chehara Pathmabandu; John Grundy; Mohan Baruwal Chhetri; Zubair Baig. An Informed Consent Model for Managing the Privacy Paradox in Smart Buildings. 2020 35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). 2020.

[3] Seung-Hyun Kim; Han-Gyu Ko; In-Young Ko; Daeseon Choi. Effects of Contextual Properties on Users' Privacy Preferences in Mobile Computing Environments. 2015 IEEE Trustcom/ BigDataSE/ ISPA. 2015.

[4] S. Priyadharshini; S. Shanthi; A Survey On Detecting Android Malware Using Machine Learning Technique. 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS). 2021.

[5] Zhi Lu; Vrizlynn L. L. Thing. "How Does It Detect A Malicious App?" Explaining the Predictions of AI-based Malware Detector. 2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). 2022.

[6] Shaurya Rawat; Rushang Phira; Prachi Natu. Use of Machine Learning Algorithms for Android App Malware Detection. 2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT). 2021,

[7] Tianming Liu; Haoyu Wang; Li Li; Guangdong Bai; Yao Guo; Guoai Xu. DaPanda: Detecting Aggressive Push Notifications in Android Apps. 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2019.

[8] Byoungchul Kim; Jaemin Jung; Sangchul Han; Soyeon Jeon; Seong-je Cho; Jongmoo Choi. A New Technique for Detecting Android App Clones Using Implicit Intent and Method Information. 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN). 2019.

[9] Guoai Xu; Chengpeng Zhang; Bowen Sun; Xinyu Yang; Yanhui Guo; Chengze Li; Haoyu Wang.

AppAuth: Authorship Attribution for Android App Clones. IEEE Access. 2019.

[10] Eunhoe Kim; Sungmin Kim; Jaeyoung Choi. Detecting Illegally-Copied Apps on Android Devices. 2013 International Conference on IT Convergence and Security (ICITCS). 2013.

[11] Carlos Cilleruelo; Enrique-Larriba; Luis De-Marcos; Jose-Javier Martinez-Herráiz. Malware Detection Inside App Stores Based on Lifespan Measurements. IEEE Access. 2021.

[12] Muhammad Sajidur Rahman; Blas Kojusner; Ryon Kennedy; Prerit Pathak; Lin Qi; Byron Williams. So {U} R CERER: Developer-Driven Security Testing Framework for Android Apps. 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). 2021.

[13] Li Yuan. Detecting similar components between android applications with obfuscation. 2016 5th International Conference on Computer Science and Network Technology (ICCSNT). 2016.

[14] Yi He; Qi Li. Detecting and defending against inter-app permission leaks in android apps. 2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC). 2016.

[15] Byoungchul Kim; Kyeonghwan Lim; Seong-Je Cho; Minkyu Park. RomaDroid: A Robust and Efficient Technique for Detecting Android App Clones Using a Tree Structure and Components of Each App's Manifest File. IEEE Access. 2019.