

ICEIC 2019

International Conference on Electronics,
Information, and Communication (ICEIC) 2019

January 22 (Tue) ► 25 (Fri), 2019
Pullman Auckland Hotel, New Zealand

◀◀ Information

◀◀ Time Table

◀◀ Contents

◀◀ Papers

◀◀ Author Index

◀◀ Exit

Time Table

January 22nd (TUE), 2019

Room	Princes A+B	Princes C	Gallery 2	Gallery 3	Regatta B	Regatta C
15:00~17:00	Registration (Lobby 1F)					
17:30	Welcome Reception (Top of the Town 14F)					

January 23rd (WED), 2019

Room	Princes A+B	Princes C	Gallery 2	Gallery 3	Regatta B	Regatta C
08:30~16:00	Registration (Lobby 1F)					
09:00~10:30		P01	SA01	SA02	SA03	SA04
10:45~11:00	Opening Ceremony (Princes A+B)					
11:00~12:30	Plenary Talk 1, 2 (Princes A+B)					
12:30~14:00	Lunch					
14:00~15:30		P02	SB01	SB02	SB03	SB04
15:30~16:00	Coffee Break					
16:00~17:30		P03	SC01	SC02	SC03	SC04

January 24th (THU), 2019

Room	Princes A+B	Princes C	Gallery 2	Gallery 3	Regatta B	Regatta C
09:00~16:00	Registration (Lobby 1F)					
10:00~12:00		P04	SD01	SD02	SD03	SD04
12:00~13:30	Lunch					
13:30~15:00		P05	SE01	SE02	SE03	SE04
15:00~15:30	Coffee Break					
15:30~17:00		P06	SF01	SF02	SF03	SF04
18:00	Banquet (Princes A+B)					

January 25th (FRI), 2019

Room	Princes A+B	Princes C	Gallery 2	Gallery 3	Regatta B	Regatta C
09:00~11:00	ICEIC 2019 Group Discussion and Wrap-up					



A 4.7 μA Quiescent Current Synthesizable Digital Low Dropout Regulator in 28-nm CMOS	890
Jonghyun Oh, Jun-Eun Park, and Deog-Kyoon Jeong <i>Seoul National University, Korea</i>	
Extended ICP based SLAM using straight line and point cloud	892
Ung-Hee Lee, Sung-Hyeon Joo, Kyung-Tae Nam, Jong-Wan Seo, and Tae-Yong Kuc <i>Sungkyunkwan University, Korea</i>	
Study on the performance improvement of face recognition-based security system using high-level image processing technology	897
Hyun Ahn ¹ , Young-Hwan Yoon ¹ , Sangjoon Lee ¹ , Yong-Min Lee ¹ , and Kye-Shin Lee ² ¹ <i>Sun Moon University, Korea</i> , ² <i>The University of Akron, USA</i>	
Self-Adaptive System Verification based on SysML	900
Seung-Min Lee ¹ , Soojin Park ² , and Young B. Park ¹ ¹ <i>Dankook University, Korea</i> , ² <i>Sogang University, Korea</i>	
Analysis of Correlation Characteristics of 10230 Period PRN Code using Concatenated Gold Code	903
Seung Tae Kim and Jae Min Ahn <i>Chungnam National University, Korea</i>	
A Sidelobe Mitigation Method for an FM-radio-based PCL System	905
So-Young Son ¹ , Geun-Ho Park ¹ , Hyoung-Nam Kim ¹ , Kyu-Ha Song ² , and Jun-Il Ahn ² ¹ <i>Pusan National University, Korea</i> , ² <i>Agency for Defense Development, Korea</i>	
Various Device Structures for Steep Switching Silicon-On-Insulator Feedback Field Effect Transistor	909
Changhoon Lee and Changhwan Shin <i>Sungkyunkwan University, Korea</i>	
Real Time Android Ransomware Detection by Analyzed Android Applications	912
Ju-Seong Ko, Jeong-Seok Jo, Deuk-Hun Kim, Seul-Ki Choi, and Jin Kwak <i>Ajou University, Korea</i>	
Net Load Variability in Future Jeju Power System with Very High Penetration of Renewable Generation	917
Jinyeong Lee, Rakkyung Ko, Sangmin Ryu, and Sung-Kwan Joo <i>Korea University, Korea</i>	
Image Restoration for CsI(Tl)-Scintillator Mammography Detectors	919
Dong Sik Kim and Eunae Lee <i>Hankuk University of Foreign Studies, Korea</i>	

Real Time Android Ransomware Detection by Analyzed Android Applications

Ju-Seong Ko*, Jeong-Seok Jo, Deuk-Hun Kim, Seul-Ki Choi, Jin Kwak

Ajou University

Suwon-si, Gyeonggi-do, Republic of Korea

*jsko.isaa@gmail.com, jsjo.isaa@gmail.com, dhkim.isaa@gmail.com, skchoi.isaa@gmail.com, security@ajou.ac.kr

Abstract—Recently, damage caused by ransomware has been increasing in PC and Android environments. There are many studies into real-time ransomware detection because the most important time to prevent encryption is before ransomware is able to execute its malicious process. Traditional analyses determine an application is ransomware or not by static/dynamic methods. Those analyses can serve as components of a method to detect ransomware in real time. However, problems can occur such as the inability to detect new/variant/unknown ransomware. These types require signed patches from a trusted party that can only be created after attacks occur. In a previous study into real-time new/variant/unknown ransomware detection in a PC environment, important files are monitored and only programs that have been previously analyzed and evaluated as non-malicious are allowed. As such, programs that have not been analyzed are restricted from accessing important files. In an Android environment, this method can be applied using Android applications to prevent emerging threats and verify consistency with user intent. Thus, this paper proposes a method of detecting new/variant/unknown ransomware in real time in an Android environment.

Keywords—Android; New/Variant/Unknown Ransomware; Real-Time Detection; Encryption

I. INTRODUCTION

Ransomware is becoming increasingly problematic, causing significant damage worldwide. Ransomware—a compound word of ransom and software—is used to encrypt and disable a user’s valuable data, requiring payment to recover it. Unlike other malware, the encrypted data cannot be recovered because only the attacker has the decryption key. The spread of ransomware is primarily facilitated by phishing mails, untrusted sites, and file sharing sites with social engineering techniques [1].

Recently, Android has become a target for malware because it is an open source platform with globalization and fragmentation. Additionally, Android allows users to install applications from not only untrusted sites but also the Google Play Store. This enables hackers to fraud users [2].

The most important is to prevent data encryption by ransomware. As such, various methods to determine in real time whether an application is ransomware or a normal encryption have been studied. Traditional methods to detect

ransomware applications include static analysis [3] and dynamic analysis [4]. However, they are considered vulnerable and easily circumvented by attackers.

In [5], a 3-step method is proposed with traditional approaches for detecting ransomware applications in real time. The first step of their method is monitoring, the second is dynamic analysis, and third is static analysis. However, some problems can occur such as false detection, as well as the possibility of circumvention and vulnerability for new/variant/unknown ransomware.

Reference [6] pointed out that methods to patch information of ransomware—such as detection based on signatures provided by a trusted party—cannot prevent new/variant/unknown ransomware. This is because the trusted party can only get a ransomware sample after users have been attacked. They proposed a method to detect ransomware in a Windows environment in real time without patches from a trusted party. However, problems can occur if this method is implemented in an Android environment because of its Windows components and results in performance overhead by inspecting all access operations on monitoring process. As such, we need to implement a similar method in an Android environment and decrease overhead by reducing the number of monitored targets.

Section 2 of this paper discusses works related to ransomware on Android, ransomware detection methods, and Shannon entropy. In section 3, we analyze how specific problems in the previous studies may occur. In section 4, we propose our method for detecting ransomware applications in real time with improved monitoring performance in an Android environment. We then outline a requirement for implementation of our method in section 5. Finally, we present our conclusions and discuss future work to implement the method in section 6.

II. RELATED WORKS

A. Ransomware in an Android Environment

Ransomware is any type of malware that demands a sum of money from the infected user. On Android, there are two general categories of ransomware: lock-screen and crypto. In lock-screen types, the hijacked resource is blocked by an image that fully covers the screen. Crypto ransomware encrypts the

user's valuable data. Android ransomware typically fits the general definition of a trojan horse. In some cases, the malicious APKs copy only the name and icon of the normal application or disguise it as a legitimate file in an SMS or e-mail. In this way, attackers use social engineering to manipulate victims into installing malicious APKs and executing functions [7]. An example of how attackers spread ransomware as legitimate applications is shown as Fig. 1 [8] and Fig. 2 [9].

Subject:
Urgent W2 Request

Message body:
Hi [TARGET],
How are you today? I need you to send me the W2 of all the Company's Employees, I need it for a Quick Review
thanks
[IMPERSONATED EXECUTIVE]

Fig. 1. Spreading by e-mail [8]

BEST - BUY CUSTOMER! You've been chosen for a 1K REWARD to use at any location. To claim go to <http://bit.ly/3w4K0ff> & enter 7622 txt stop to opt out

Fig. 2. Spreading by SMS [9]

Such attacks can be successful because Android devices are allowed to install applications from third-party app stores. Additionally, once Google releases a new version of the OS, manufacturers may not upgrade their OSs immediately, or at all. A device running on an older version of the OS can result in certain vulnerabilities remaining unpatched for an attacker to exploit [10].

B. Static Analysis

Static analysis involves an automated tool that takes the source code of a program as input and examines it without executing. It then verifies the code structure and statement sequences, checks how variables are processed throughout the different function calls, analyzes strings in information files—such as xml—and yields a result. A static analysis starts by representing the analyzed app's information to some abstract models based on the purpose of the analysis. Those models provide a simplified interface for supporting client analyses [3]. Fig. 3 is an example of static analysis of xml files by simplocker [5].

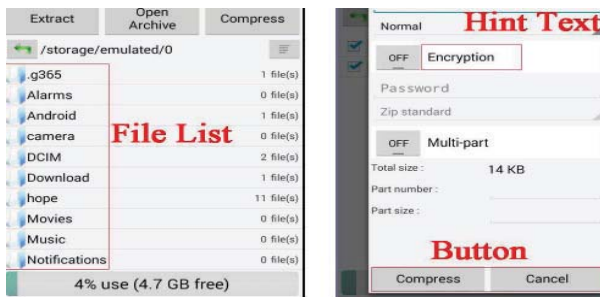


Fig. 3. An example of static analysis [5]

C. Dynamic Analysis

While static analysis can rely on Java bytecode extracted by disassembling APKs and does not require code execution, dynamic analysis can examine all code executed by an application [11]. This is necessary because malware can hide

malicious events behind activities that require an interaction with an Android user. Therefore, to test the apps dynamically, a tool is required that can simulate these apps to start their malicious activity. In dynamic analysis, the application is executed in a controlled environment—such as sandbox—and virtual machine to trace its behaviors [12]. Fig. 4 is an example of dynamic analysis by executing code.

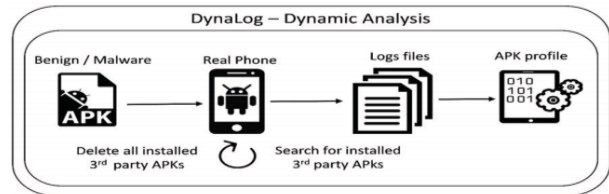


Fig. 4. An example of dynamic analysis [11]

D. Android Ransomware Detection in Real Time

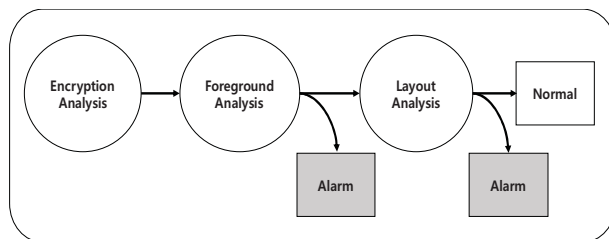


Fig. 5. Overall Process in [5]

In [5], a 3-step analysis method is proposed to determine whether an application is ransomware or not. Whether the user intended for encryption on their device is an important factor in this method. An overall process of their method is shown in Fig. 5.

First, for encryption analysis, important files predefined by the user are monitored and inspected those files were encrypted using Shannon entropy value in real time. If an important file is determined to be encrypted, the process moves to the next step; otherwise, it continues to monitor.

Second, foreground analysis checks for the encryption application's activity at the top of the activity stack—called Activity Manager. The goal of this step is to make sure that the encryption process is triggered by the foreground. That is, it checks that the encryption process is shown on the device's screen. If the application's activity is at the top of the stack, the process moves to next step; otherwise, it stops the application and sends an alarm to the user.

Finally, in layout analysis information from manifest, layout, and xml files is extracted. It is assumed that normal encryption applications show a list of files, texts, and buttons with hint strings on the screen seen by the user—this can be accomplished by static analysis. If the application has these components, the step ends and returns to the first step; otherwise, it stops the application and sends an alarm to the user.

E. New/Variant/Unknown Ransomware Detection

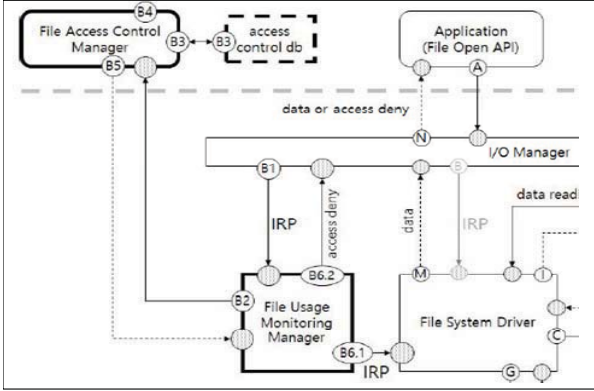


Fig. 6. A part of the structure in [6]

In [6], a method to detect new/variant/unknown ransomware without patch information from trusted parties is proposed. For this, a file usage monitoring manager, file access control manager, and access control db were created. A part of the structure from this method is shown in Fig. 6.

When an application uses APIs related to file access to read or write to a file, such an operation should be informed to the I/O manager of the OS. The I/O manager generates an I/O request packet and sends the packet to the file system driver.

The file usage monitoring manager is executed in the kernel mode of the OS and communicates to the I/O manager and file access control manager. When an application opens a file, it intercepts the file system driver-initiated packet.

The file access control manager is operated in the user mode of the OS and creates/manages a whitelist. If the file usage monitoring manager sends the file access request of an application, it determines whether the application is allowed to access the file according to a whitelist recorded in the access control db.

The access control db is stored in the user mode of the OS and it saves/manages a whitelist which consists of two elements: one is the identity of a file type and the other is a list of accessible applications for each file type.

If a request related to file access occurs, this method check the application is in whitelist—only applications in the list are allowed to access the file. If a new application wants to access a file it should be analyzed, and, if it is evaluated as being non-malicious, included in the list.

F. Shannon Entropy

Shannon entropy indicates the uncertainty of data. Cipher data has a higher entropy value than plain data because of encryption algorithm confusion properties. When ransomware encrypts a file, the file's entropy increases [6]. In [5], Shannon entropy was used to determine whether a monitored file was encrypted or not in the encryption analysis step. The Shannon entropy value, e , can be computed as in (1).

$$e = \sum_{i=0}^{255} P_{B_i} (-\log_2 P_{B_i}), P_{B_i} = \frac{F_i}{totalbytes} \quad (1)$$

F_i means the number of byte value, i , in the file. The maximum entropy value is 8 if all F_i are equal in i from 0 to 255.

III. PROBLEM ANALYSIS

There are two methods to detect Android ransomware and new/variant/unknown ransomware in real time. However, problems with these methods—including reduced accuracy and increased performance overhead—can occur. To address this, we first analyze the problems inherent to each method in this chapter.

A. Problems in Real-time Android Ransomware Detection

We analyzed two of the three steps: encryption analysis and layout analysis. The former relates to monitoring for encryption function, while the latter performs static analysis for information extraction.

• Encryption Analysis

In this step, predefined files are monitored and encrypted files are inspected. For this, Shannon entropy is used, but the results can be inaccurate. Theoretically, the entropy of cipher data is greater than that of plain data. However, in the case of plain data with high entropy, the entropy can be lower after encryption. This can produce inaccuracies such as a plain data that is considered to be encrypted, or cipher data that is considered unencrypted.

• Layout Analysis

In this step, information is extracted via static analysis from a non-executed file. For this, layout files are analyzed to verify that all components are on screen and xml files are checked for strings that exist in malicious applications. However, strings are only inspected based on a whitelist of words expected to be found in normal applications. As such, inaccuracies can occur, such as strings with the same meaning circumventing the whitelist.

TABLE I. SUMMARY OF PROBLEMS IN JING CHEN'S METHOD

Step	Problems
Encryption Analysis	Inaccuracy of entropy for plain and cipher data
Layout Analysis	Inaccuracy from circumventing the method

B. Problems in New/Variant/Unknown Ransomware Detection

We analyzed the processes by which the file usage monitoring manager intercepts a request packet and the file

access control manager checks an application against the whitelist in the db.

To verify whether the application is non-malicious, whenever a request related to file access occurs, the method performs all analysis processes. The processes include the following sequences.

- The file usage monitoring manager sends a request packet to the file access monitoring manager
- The file access monitoring manager reads the whitelist to verify whether the requesting application is allowed.
- The file access monitoring manager sends the result to the file usage monitoring manager

The above process has file I/O operations, as well as communication between managers. As such, if the process is performed several times or many I/O requests are made, increased performance overhead can occur.

TABLE II. PROBLEMS IN TWO STUDIES

Num	Problems
Problem 1	Less accurate ransomware detection
Problem 2	High performance overhead to verify applications

IV. PROPOSED METHOD

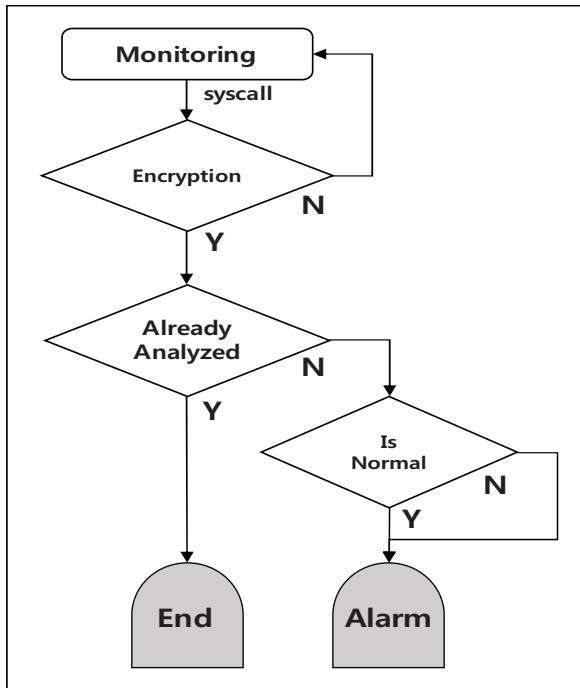


Fig. 7. Overall process of our method

The overall process of our method for real-time ransomware detection in the Android environment is shown in

Fig. 7. This method is composed of 4 analysis steps: Monitoring, Encryption, Already Analyzed and Is Normal.

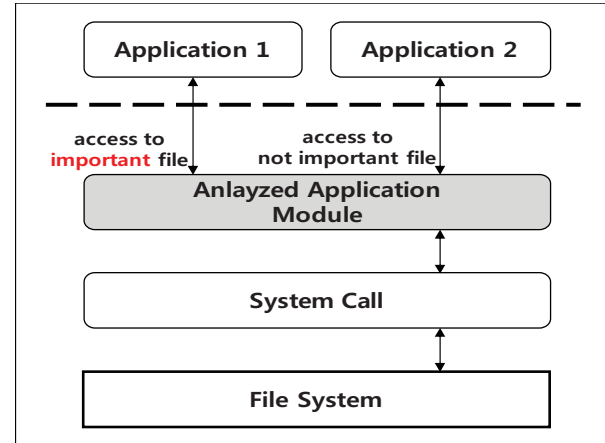


Fig. 8. Analyzed application module for the proposed method

A. Monitoring

In this step, important files predefined by the user are monitored. The analyzed application module intercepts requests from APIs to read or write to a file. If the target of the operation is not in the predefined files, the module sends the requests to origin system call or it pauses the requests. Subsequently, it moves to the next step with logs including the application package name and the name of the target file. In this step, analysis is performed on the operations of read and write. The analyzed application module is shown in Fig. 8.

B. Encryption

In this step, whether the targeted file was encrypted or not is determined according to the result of Shannon entropy. If the target file was not encrypted, the method returns to the monitoring step, or proceeds to the next step. However, we analyzed that Shannon entropy can produce inaccurate results. To address this, we used the average entropy of the divided parts of the file—such as the average of each parts' entropies divided by 4,096 bytes.

C. Already Analyzed

In this step, the package name is used to check whether the application was previously analyzed. For this, package names of pre-analyzed applications should be stored statically. If the application is previously analyzed and determined to be normal, access is granted and the process ends. Otherwise, the behavior is denied and the process moves to the next step.

D. Is Normal

In this step, information is extracted, analyzed and provided to the user. Besides static analysis and dynamic analysis, many studies about how to extract/analyze information from Android applications can be included. To improve the accuracy of problem 1, we modify the basis of the layout analysis inspection from a whitelist to a blacklist with words such as “pay” and “coin.”

The improvements to this method based on problems in the previous studies are shown as Table 3.

TABLE III. IMPROVEMENTS BY PROPOSED METHOD

Num	Improvements
Problem 1	Enhanced accuracy from average of entropy and proper analysis method
Problem 2	Decreased overhead by limiting the target of monitoring operations to encryption exclusively

V. REQUIREMENT FOR IMPLEMENTATION

In our method, we should know which application requests system call and which file is targeted by the operation. However, it is difficult to take the information from Android devices without some specific ways. Therefore, we need a system call hooking to acquire the application package name and target file name.

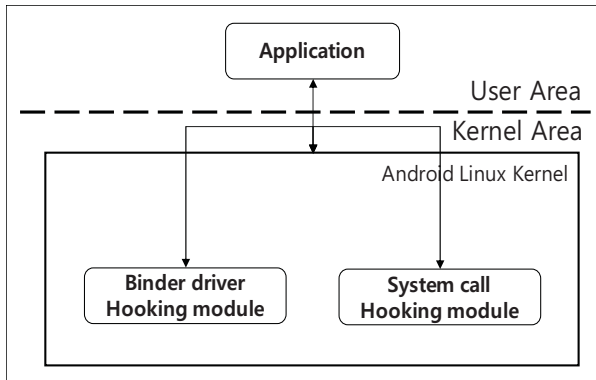


Fig. 9. An example of system call hooking

Fig. 9 is an example of a system call hooking in an Android environment [12]. In the monitoring step, we should store logs with the requesting application package name and target filename. By intercepting system calls, we can get this additional information necessary for the Monitoring step.

VI. CONCLUSIONS

In this paper, we analyzed the problems of two previous studies about how to detect new/variant/unknown ransomware in real time on Android. Subsequently, we proposed a real-time ransomware detection method as well as requirements to implement the method. This method can prevent new/variant/unknown ransomware with less performance overhead in an Android environment.

We focused on the main function of processes from the Monitoring and Already Analyzed steps. However, how we provide information to the user in the Is Normal step is also important. Thus, additional studies about how to analyze the applications will be required. In future work, implementation of the proposed method should be performed with detailed analysis methods and evaluated for detection accuracy.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2017R1E1A1A01075110).

REFERENCES

- [1] Korea Internet & Security Agency, "16 Ransomware Trends and '17 Outlook", 2017
- [2] Meet Kanwal, Sanjeev Thakur, and Rishabh Lashkari, "An app based on static analysis for Android ransomware", 2017 8th International Conference on Computing, Communication and Automation, 2017.
- [3] Li Li, Tegawende F. Bissyande, Mike Papadakis, Siegfried Rasthofer, Alexandre Bartel, Damien Ocateau, Jacques Klein, and Yves Le Traon, "Static Analysis of Android Apps: A Systematic Literature Review", ELSEVIER Information and Software Technology Vol. 88, 2017.08.
- [4] Mohammed K. Alzaylaee, Suleiman Y. Yerima, and Sakir Sezer, "Improving Dynamic Analysis of Android Apps Using Hybrid Test Input Generation", IEEE International Conference On Cyber Security And Protection Of Digital Services (Cyber Security 2017), 2017.07.
- [5] Jing Chen, Chiheng Wang, Ziming Zhao, Kai Chen, Ruiying Du, and Gail-Joon Ahn, "Uncovering the Face of Android Ransomware Characterization and Real-Time Detection", IEEE Transaction on Information Forensics and Security, 2017.
- [6] Dae-Youb Kim, Geun-Yeong Choi, and Ji-Hoon Lee, "White List-based Ransomware Real-time Detection and Prevention for User Device Protection", 2018 IEEE International Conference on Consumer Electronics, 2018.
- [7] Robert Lipovsky and Lukas Stefanko, "Android Ransomware: From Android Defender to DoubleLocker", ESET, 2018.01.
- [8] McAfee. Part of Intel Security, "What's on the Horizon for 2016", Mobile Threat Report of Intel Security, 2016.
- [9] Ben Nahorney, "ISTR Email Threats 2017", An ISTR Special Report, 2017.10
- [10] Ankita Kapratwar, Fabio Di Troia, and Mark Stamp, "Static and Dynamic Analysis of Android Malware", Hindawi Security and Communication Networks, 2018.06.
- [11] Keyur Kulkarni, "Open Source Android Vulnerability Detection Tools: A Survey", arXiv Cryptography and Security, 2017.07.
- [12] Youn-Sik Jeong, Hwan-Taek Lee, Seong-Je Cho, Sangchul Han, and Minkyu Park, "A kernel-based monitoring approach for analyzing malicious behavior on Android", the 29th Annual ACM Symposium on Applied Computing, 2014.