symmetry

MDPI

Article

KRRecover: An Auto-Recovery Tool for Hijacked Devices and Encrypted Files by Ransomwares on Android

Senmiao Wang [1], Sujuan Qin [1,*], Nengqiang He [2,*], Tengfei Tu [1,*], Junjie Hou [1], Hua Zhang [1] and Yijie Shi [1]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; wangsenmiao@bupt.edu.cn (S.W.); FoxyWinner@bupt.edu.cn (J.H.); zhanghua_288@bupt.edu.cn (H.Z.); yijieshi2000@bupt.edu.cn (Y.S.)

[2] National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC), Beijing 100029, China

* Correspondence: qsujuan@bupt.edu.cn (S.Q.); hnq@cert.org.cn (N.H.); tutengfei.kevin@bupt.edu.cn (T.T.)

Abstract: Ransomwares on Android have become a challenging threat, performing tasks such as hijacking screen resources, locking devices, and encrypting files. Even worse, with the evolution of ransomwares, many ransomwares can disable USB interfaces of mobile devices. It is difficult for users to recover their devices or decrypt files with the help of other equipment and gives monetary damages to victims. In this paper, we analyse the symmetry between the ransom behaviours and the source code of screen resource hijacked ransomwares, devices locked ransomwares and files encrypted ransomwares. We also propose strategies of recovering hijacked resources, recovering hijacked devices and decrypting encrypted files. To protect mobile devices and private files from ransomwares, we design and implement an automatic recovery application—KRRecover—which is used to recover the hijacked devices and decrypt encrypted files on Android.

Keywords: Android ransomwares; files decryption; hijacked devices recovery

Citation: Wang, S.; Qin, S.; He, N.; Tu, T.; Hou, J.; Zhang, H.; Shi, Y. KRRecover: An Auto-Recovery Tool for Hijacked Devices and Encrypted Files by Ransomwares on Android. *Symmetry* **2021**, *13*, 861. https://doi.org/10.3390/sym13050861

Academic Editor: Giuseppe Bagliesi

Received: 10 April 2021
Accepted: 5 May 2021
Published: 12 May 2021

1. Introduction

Ransomware is a kind of malware which can lock devices, hijack screen resources, and encrypt files. Ransomwares are low-cost in implementation but pose great threat to users. In recent years, with the increasing market share of Android systems, the number of ransomwares is constantly increasing. According to statistics, ransom events in 2020 are 20% higher than that in 2019 [1–3]. To reduce the loss of ransomwares to users, researchers have made positive response from the aspects of system modification, ransomwares detection, and devices recovery.

The Android system has been launched many versions to prevent attackers from implementing ransomwares by making the system window top set. Android L restricts the use of ***getRunningAppProcesses*** and ***getAppTasks*** methods to resist the Activity hijacking ransomwares [4]. Android N forces developers can only dynamically apply for related permissions when they need to reset or delete the password of devices [5]. Android O has disabled five ransomwares commonly used window types, for example, ***TYPE_PHONE*** [6]. According to the official statistics of the distribution of Android operating systems of various versions, 61.3% of devices use Android system below Android N [6]. It indicates that there are still many devices facing the high-risks brought by ransomwares. In addition, different system versions have different ways to make windows overhead the screen. So it is hard to find a unified way to protect devices from being attacked by ransomwares. It is a challenge we are facing for.

Some researchers proposed some methods on how to recover files encrypted by ransomwares on PC. These studies are mainly based on monitoring different objects to obtain related information and recover encrypted files. Lee et al. [7] proposed the method

to obtain keys and decrypt files by monitoring crypto methods. PayBreak [8], a file recovery tool based on Windows 7 system, obtains encryption information by monitoring dynamic link libraries and static link libraries. In addition, it uses escrowed keys and offset iterations to decrypt encrypted files. In terms of automatic recovery of locked devices, to the best of our best knowledge, there is no relevant research that we can reference on PCs.

With the evolution of ransomwares, some ransomwares already have the function of automatically disabling the USB interfaces of devices, i.e., ransomwares can turn off USB debugging switches automatically after they are activated. It means that users cannot recover their devices or decrypt files with the help of other devices.

Contributions To protect mobile devices and private files from ransomwares, we proposed strategies of recovering hijacked screen resource, locked devices and encrypted files. We designed and implemented KRRecover, an auto-recover tool for hijacked devices and encrypted files on Android. To our best knowledge, this is the first time to implement the automatically recovery application to recover hijacked devices and encrypted files. The main contributions are as follows.

Analysis of three typical ransom behaviours on Android. We decompiled 4750 ransomwares and extracted their sensitive API callings and declared permission. For the convenience of research, we used DBSCAN to cluster ransomwares with familiar behaviours together. We found that ransomwares have three typical behaviours, including hijacking screen resource, locking devices and encrypting files. We analysed the three typical behaviours based on source code.

Strategies for recovering hijacked devices and encrypted files. After analyzing the implementation of ransom behaviours, we proposed solutions for recovering hijacked devices and recovering encrypted files.

An on-device tool for recovering hijacked devices and encrypted files. To reduce the loss caused by ransomwares, we implemented KRRecover, an on-device tool of recovering hijacked devices and encrypted files. After KRRecover obtain root authority, it can automatically delete passwords ransomwares set and recover hijacked devices. KRRecover also can recover files encrypted by cryptoAPIs.

2. Related Work

Recently, how to reduce the loss caused by ransomwares has received a lot of attention [8–13]. To our best knowledge, strategies of recovery against ransomwares can be divided into hijacked devices recovery and encrypted files recovery.

2.1. Hijacked Devices Recovery

Hijacked devices recovery aims to release hijacked screen resource and unlock devices. At present, we can achieve the goal with the help of third party tools or Android boot loader mode.

Android Debug Bridge(ADB) [14] is a tool to interact with simulators or real devices through the computer. The tool provides a variety of operation commands and access to UNIX shell. When the devices are locked, if the USB debugging interface is open, users can recover them with the help of ADB by means of connecting the external device via USB interface.

If the USB debugging interface is closed, Android Debug Bridge cannot be used for hijacked devices recovery.

FastBoot [15] is a boot loader mode. With the help of FastBoot, developers can rewrite the system on devices without USB connection. If the device is locked, users can restore the use of the device by rewriting the system with FastBoot [15].

However, rewriting the system may have the risk of losing data on the device. In addition, different device models have different ways to switch into FastBoot [16] by means of pressing different buttons. It is not uniform. Different Android devices have different system designs. The operations are also different after switching into the third-part recovery with the help of FastBoot [15]. If the device system has a mandatory foreground,

users need to recover the device with the help of database operations. It requires that the operators are professionals.

2.2. Encrypted Files Recovery

Encrypted files recovery aims to decrypt files encrypted by ransomwares. At present, many files encrypted by ransomwares are still recovered by means of using reverse engineering. Android security analysis people or users have some background knowledge about Android can find the encryption key in the source code, so as to use the encryption key to decrypt files.

Later, with the development of technology of dynamic test, followed much research that involved obtaining target data when the application is running on Android, like Mobile-sandbox [17]. It can obtain the target data by hooking the underlying function of the application. Research recovered encrypted files on PC by means of using sandbox [17] to hook the underlying function of ransomwares and obtain encryption keys. Then encrypted files can be decrypted with encryption keys. To our best knowledge, there is still no tool for recovering encrypted files on Android.

Chen J et al. [18] use lure technology to protect privacy files on Android. It makes the file index as the lure files to induce ransomwares encrypting the lure file after ransomwares are activated, so as to protect users' private files. It requires lure files should be deployed effectively or privacy files still can be encrypted by ransomwares.

In the light of these, there is an urgency for a light and users-friendly on-device tool of recovering hijacked devices and encrypted files.

3. Analysis of Ransomwares

In this section, we will give the method of ransomwares analysis and analyse the three typical behaviours based on source code.

3.1. Method of Ransomwares Analysis

The method of ransomwares analysis is shown in Algorithm 1. For the convenience of analysis, we use $DBSCAN$, a kind of clustering algorithm, to cluster ransomwares with familiar behaviours together. We use $cluster[]$ to represent the result of clustering, n represent the number of ransomwares we select to analyse in each cluster and sus to save the relation between ransom behaviours and related source code.

To speed up the efficiency of analyzing, we randomly select n ransomwares in each cluster. For each ransomware, we firstly decompile the ransomware to obtain readable AndroidManifest.xml and classes.dex. We extract sensitive permissions in Androidmanifest.xml and save them in permissionList. For each permission, if there exists related sensitive API callings in classes.dex, save $< permission, APIcallings >$ as a key/value pair in sus. We extract sensitive API callings in classes.dex and save them in $APIList$. If some API callings in $APIList$ can be combined to represent ransom behaviour, save $< behaviour, APIcallings >$ as a key/value pair in sus. With the help of sus, $permissionList$ and $APIList$, we can more accurate analysis results.

After analyzing 4700 screen resource and devices hijacked ransomwares and 126 files encrypted ransomwares from open dataset, we found that nearly 98.4% ransomwares have the behaviour of hijacking screen resource. Nearly 97% ransomwares have the behaviour of locking devices and 2.6% ransomwares have the behaviour of encrypting files, i.e., ransomwares usually extort users by these means.

Algorithm 1 Ransomware Analysis

Input: ransomwares

```
function ANALYSIS
    Cluster[] ← DBSCAN(ransomwares)
    for i ← 0 to num do
        Random (n ransomwares in cluster[i])
        for j ← 0 to n do
            Decompile (cluster[i][j])
            permissionList.append (sensitive permissions)
            if (permissionList) then
                if (related sensitive API callings) then
                    sus.append(< permission, APIcallings >)
                end if
            end if
            APIList.append(sensitive API callings)
            if (APIList) then
                if (some API callings can be combined to represent ransom behaviors) then
                    sus.append(< behavior, APIcallings >)
                end if
            end if
        end for
    end for
    Manual Analyze(sus, permissionList, APIList)
    return (< behavior, implementation >))
end function
```

3.2. Hijack Screen Resource

Screen resource hijacking is a kind of ransom behaviour. It prevents users from using their devices normally. As shown in Figure 1, it constantly set interfaces of ransomwares as the top interfaces even users click Home buttons or Back buttons. The typical screen resource hijacking can be represented as R_{HSR}. As shown in Formula (1), R_{HSR} can be represented as a tuple with B_{HSR} and M_{HSR}. As shown in Formula (2), B_{HSR} is the subset of behaviours of screen resource hijacking, including Home button disabled, Back button disabled and so on. As shown in Formula (3), M_{HSR} is the subset of the code implementation of B_{HSR}. ***FULL_SCREEN*** and ***setFlags*** with parameter 1024 can set activities as the first priority and make interfaces as the top activities. ***setCancelable***, ***OnkeyDown*** and other methods can disable buttons like ***Home*** buttons and ***Back*** buttons.

$$R_{HSR} = \{< B_{HSR}, M_{HSR} >\} \tag{1}$$

$$B_{HSR} \subseteq \{btn\_dis_{Home}, btn\_dis_{Back}, USB\_dis, ...\} \tag{2}$$

$$M_{HSR} \subseteq \{\text{ setFags (1024), setCancelable }, \ldots\} \tag{3}$$

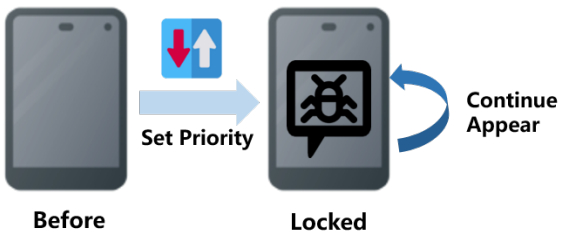

Figure 1. Screen Resource Hijacking.

3.3. Lock Devices

Device locking is the most common ransom behaviour. As shown in Figure 2, it prevents users from using their devices normally by means of resetting users' passwords. Typical device locking can be represented as R_{LD}. As shown in Formula (4), R_{LD} can be represented as a triple with B_{LD},M_{LD} and A_{LD}. A_{LD} contains the absolute paths of databases and files with passwords. As shown in Formula (5), B_{LD} represents typical behaviours, including buttons disabled, USB disabled, devices locked and so on. M_{LD} represents the code implementation of B_{LD}. The detailed implementation of devices locking is described below in Algorithm 2.

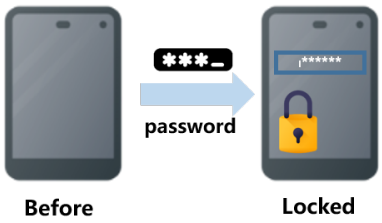

Figure 2. Devices Locking.

Algorithm 2 Implementation of Devices Locking

Input: locksetting.db, save_path, password
1: $Device\_salt \leftarrow$ **get_salt**$(lockingsetting.db)$
2: $Pass\_MD5 \leftarrow$ **MD5**$(password + device\_salt)$
3: $Pass\_SHA1 \leftarrow$ **SHA1**$(password + device\_salt)$
4: $Key \leftarrow$ **hex**$(Pass\_MD5) +$ **hex**$(Pass\_SHA1)$
5: **Delete**$(original\_key)$
6: **if** *save_path* **then**
7: **Write**$(key, save\_path)$
8: **else**
9: **Write**$(key,'/data/system')$
10: **end if**
11: **Lock device**

The *locksetting.db* represents the database with information of salt value. The *save_path* represents the absolute path to save key value with the new password. The password represents the password proclaimed in writing. The implementation of devices locking contains three steps.

(1) **New key initialization.** The attacker obtains the salt vale of the device by means of analyzing related databases or using Android reflection. After getting the salt value, the attacker splices the password and salt value together as a new result and respectively calculate its MD5 and SHA1 value. Then the attacker splices the MD5 and SHA1 as a new key.

(2) **Key substitution.** After the new key is initialized, it is written in password.key. The attacker deletes the file with the original password and uses the new key to replace the original one.

(3) **Device lock.** After the key has been replaced, the attacker makes the ransomware lock the device by means of calling $lockNow()$.

$$R_{LD} = \{< B_{LD}, M_{LD}, A_{LD} > | \; A_{LD} = \{ \text{ path } \}\} \tag{4}$$

$$B_{LD} \subseteq \{ \text{ dis }_{\text{button}} \text{ , dis }_{\text{USB}} \text{ , lock }_{\text{devices}} \text{ ... } \} \tag{5}$$

3.4. Encrypt Files

Files encryption is a kind of typical ransom behaviour. As shown in Figure 3, it prevents users from using or viewing files by means of encrypting files. Sometimes files encryption is accompanied by devices lock or screen resource hijacking. The typical files encryption can be represented as R_{EF}. As shown in Formula (6)–(8), R_{EF} can be represented as a triple with B_{EF},M_{EF} and A_{EF}. A_{EF} represents the absolute path of private files. B_{EF} represents typical behaviours, including finding target files, encrypting files and so on. M_{EF} represents the code implementation of B_{EF}. The typical files encryption can be described below.

(1) **Find target files.** The attacker usually targets the most frequently used files or files suffixed with *.txt*, *.jpg*, *.doc* and so on. The attacker finds eligible files and adds their paths to the category.

(2) **Encrypt files.** To encrypt files, the attacker has to apply permissions related to reading or writing files. The encryption methods like *AES* or *DES* are called to encrypt target files.

(3) **Substitute files.** After encrypting target files, the attacker usually substitutes the original files by means of overlaying original files directly or deleting original files and copying encrypted files to original paths.

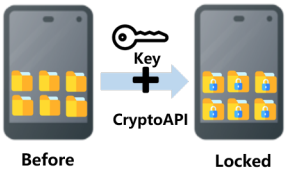

Figure 3. Encrypting files.

$$R_{EF} = \{\langle B_{EF}, M_{EF}, A_{EF} > | A_{EF} = \{ \text{ path } \}\} \tag{6}$$

$$B_{EF} \subseteq \{delete\_files, ecrypt\_files, find\_target...\} \tag{7}$$

$$M_{EF} \subseteq \{AES(), DES(), EndecodeUtils.deCrypto() \ldots\} \tag{8}$$

4. Method of Recovery

In this section, we propose the method to recover devices or files infected by ransomwares with hijacking screen resource, locking devices and encrypting files.

Assumptions. The research of auto-recovery method is based on the following two assumptions in this paper. First, the auto-recover tool can get the root permission by preloading. It will run in the background after users install it and restart their devices, i.e., there is no ransomwares on devices before the auto-recover tool installed and activated. Second, we can recover the encrypted files by using cryptoAPIs and non-dynamic encrypting. As we aim at recovering encrypted files on Android, the way of decompiling ransomwares and obtaining the key is not in our consideration.

Hijacked Screen Resource Recovered. The behaviour of hijacking screen resource is implemented by set different parameters of the system windows, so that ransomwares can make their interfaces the highest priority. Therefore, the key of recovering hijacked screen resource is to judge whether the current top-level Activity or Service belongs to ransomwares.

To recover hijacked screen resource, it is necessary to monitor the processes and top-level activities or services on devices. If an application in the monitor list is running and the current top-level activity or service belong to the application, the device has probably been infected by ransomwares. So we need root permission to kill the running activities and services to recover the screen resource.

Locked Devices Recovery. The system will search for the file that contains encryption information in the */data/system/* directory when the device is woken up. If the file is

found, the password information in the file will be read and set the password as the device unlock password. If the file is not found, the device lock password will not be set. Therefore, the key of recovering locked devices is to judge whether the configuration file in */data/system/* directory is written by ransomwares.

To recover locked devices, it is necessary to monitor the I/O processes and the operation logs on devices. If an application in the monitor list is running and the record of reading or writing .key files in */data/system/* directory are found in the log, the device has probably been infected by ransomwares. So we need root permission and read or write storage permission to delete the *.key* file and recover the locked devices.

Encrypted Files Recovery. Through manual analysis of files encrypted ransomwares that collected from ***AMD*** [19] and ***Derbin*** [20] dataset, we find that most of these ransomwares encrypt files by private key encryption. In addition, it is difficult for developers to generate different private keys for different infected devices [11], i.e., a files encrypted ransomware is based on one encryption key rather than the generation of different keys for different devices. Therefore, the key of recovering encrypted files is to obtain the key of encrypting files.

We obtain the encryption key when ransomwares encrypt files with the help of Xposed [21] on Android. To avoid generating too many redundant logs and occupying too much storage space of the devices, we rewrite the hook file of Xposed [21] so that we can only get the information of the encryption APIs and save the related information as a log file in */sdcard/* directory. At present, most of the ransomwares use ***AES*** or ***DES*** to encrypt files. Therefore, in this paper, we only choose to recover encrypted files by ***AES*** or ***DES*** encryption algorithms. We obtain the encryption key by retrieving the corresponding dynamic logs of ransomwares. After that, we can use key to decrypt encrypted files and recover them.

5. A Device and Files Recovery Application

In this section, KRRecover is proposed. KRRecover is designed to recover hijacked devices and encrypted files by ransomwares. In addition, the workflow of KRRecover is shown as Figure 4.

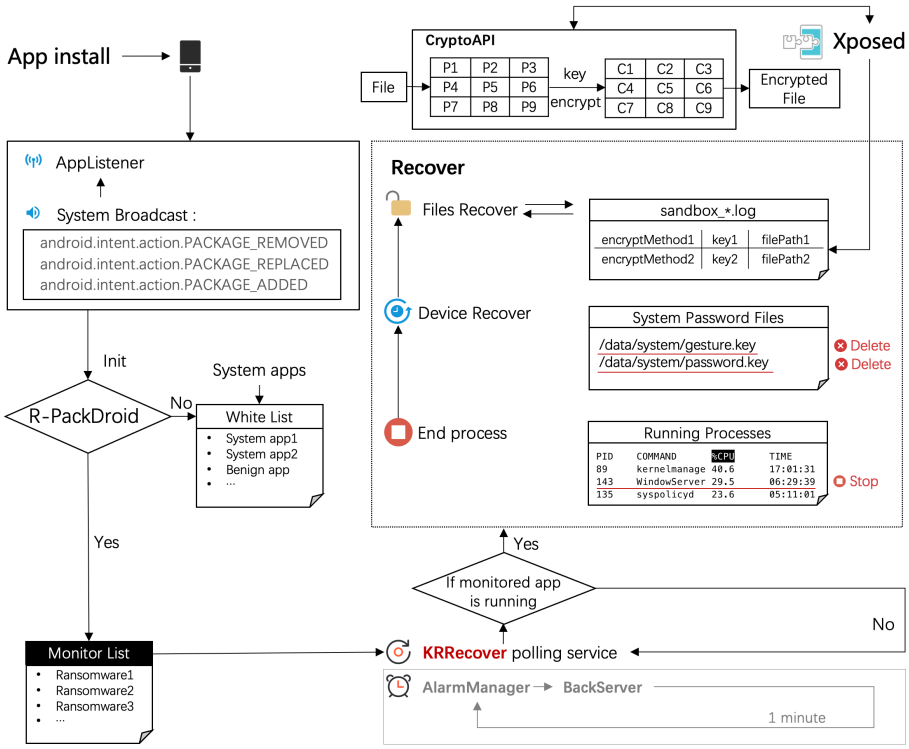

Figure 4. Workflow of KRRecover.

KRRecover includes the initialization part, monitor-trigger part, and recovery part. The initialization part includes white list initialization and monitor initialization. KRRecover initializes the white list by means of collecting the package name of system applications and pre-installed applications. KRRecover initializes the monitor list with the help of R-PackDroid [22–24]. R-PackDroid is an on-device malwares detector for Android applications. If an application is judged as a malware, its packagename will be added in the monitor list. The monitor-trigger part updates the monitor list and locates the encrypted files by monitoring the status of applications and the information of encryption methods. The recovery part recovers the hijacked devices and encrypted files by controlling the processes and monitoring encryption methods.

5.1. Monitor—Trigger Algorithm

The monitor-trigger part decides whether activates the recovery part by means of monitoring the real-time status changes of applications, processes changes and changes of target methods. Algorithm 3 shows how the monitor-trigger part works of KRRecover. ***pname*** in the algorithm represents the packagename of the application, process represents the process of the application, ***sus_run*** represents the tag of whether the application is active, ***m_list*** represents the monitor list, ***w_list*** represents the white list and ***hookinfo*** represents logs of monitored methods.

Algorithm 3 Monitor of KRRecover

Input: pname, method

Function Monitor (*pname*, *method*)
sus_run ← 0
process ← *process of pname*
log ← ””
if *uninstall app* **then**
    if *pname in w_list or pname in m_list* **then**
        remove pname in w_list or m_list
    end if
end if
if *new install app or update app* **then**
    add pname in m_list
    activate app
    if *app is ransom* **then**
        recover device
    else
        remove pname in m_list
        add pname in w_list
    end if
end if
if (*pname in m_list and provess is active*) **then**
    sus_run ← 1
    hook method
    log ← *hook info*
end if
return *sus_run*, *log*

KRRecover initializes the running tag of applications, the processes ID of monitored applications and the operation log of monitored methods. If KRRecover monitors the uninstall information of the application and the package name of the application is in the white list or monitor list, remove the package name. If KRRecover monitors the install or update information of the application, it will add the package name of the application to the monitor list and activate the application. If the application is a ransomware, KRRecover will recover the device or encrypted files. If the application is not a ransomwares, KRRecover

will move its package name from monitor list to the white list. KRRecover will traverse the monitor list to find whether the target application is in the monitor list and is running or not. If so, KRRecover will change the running status tag to **1** and hook the encryption method and assign the operation log to the log variable.

5.2. Devices Recovery Algorithm

The devices recovery is an important part of the recovery part. Algorithm 4 shows the devices recovery algorithm. m_list represents the monitor list, sus_run represents the tag of whether the application is active, ***panme*** in the algorithm represents the packagename of the application and ***Monitor*** represents aforementioned method which can monitor processes and other methods. For convenience of explanation, the storage path of password will be represented by "*/data/system/ .key*".

Algorithm 4 Devices Recovery of KRRecover

Function *Devices Recover*
for $i \leftarrow 0$ *to* $len(m\_list)$ **do**
  $sus\_run, log \leftarrow$ **Monitor**$(m\_list[i], crypto\ APIs)$
  if $sus\_run = 1$ **then**
    $pname \leftarrow m\_list[i]$
    kill the process of pname
  end if
  switch *password type*
  case *input* :
  remove /data/system/password.key
  case *graph* :
  remove /data/system/gesture.key
end for
End function

The devices recovery will be activated after it received the result of ***Monitor*** methods. KRRecover will be activated automatically after the device be operated. The monitor-trigger part updates the monitor list at anytime. If the current monitor list is empty, KRRecover will continue monitor the applications status on the device. If the monitor list is not empty, KRRecover will monitor every application in the list with the help of ***Monitor*** method. In addition, return the processes status information to the variable ***sus_run***. If the monitored application is running, KRRecover will kill the process and recover the device according to different hijacked strategies. In addition, KRRecover will delete the password file and recover the hijacked device.

5.3. Encrypted Files Recovery Algorithm

The encrypted files recovery part is another critical part of the recovery part. Algorithm 5 shows the encrypted files recovery algorithm. ***m_list*** represents the monitor list, ***sus_run*** represents the tag of whether the application is active, ***panme*** in the algorithm represents the packagename of the application and ***Monitor*** represents aforementioned method which can monitor processes and other methods.

Algorithm 5 Encrypted Files Recover of KRRecover

```
Function Files Recover
for i ← 0 to len(m_list) do
    sus_run, log ← Monitor(m_list[i], crypto APIs)
    if sus_run = 1 and crypto API info in log then
        key ← crypt key in log
        locate encrypted files
        decrypt files
    end if
end for
End function
```

KRRecover will traverse the monitor list and monitor whether the encryption methods are used in monitored applications. The variable ***sus_run*** means whether the processes of the application are active. **1** means the application is running and **0** means the application is not running. At the meantime, KRRecover will assign the operation logs to variable ***log***. After that, KRRecover will judge whether the monitored application is running and whether it has encrypted files by coming the ***sus_run*** and ***log***. If so, KRRecover will obtain the encryption key and locate the encrypted files by analyzing the operation logs. Then it will decrypt the encrypted files with the key.

6. Evaluation

This section evaluates the capabilities and effectiveness of KRRecover. We will first introduce the composition of the dataset. Then we will design four experiments to test KRRecover. For each experiment, we will first ask a question. Then we will describe the experiment procedure and result.

6.1. Dataset

The ransomwares we use in the experiment come from the open-source dataset VirusTotal [25] and AMD [19]. The composition of the dataset and the detailed behaviourss of ransomwares are shown in Table 1. For the convenience of evaluation, we divided these ransomwares into four groups.

Table 1. The Composition of Test Data.

	Number	Lock Devices	Hijack Screen Resource	Encrypt Files
Group1	475	✓		
Group2	25		✓	
Group3	108			✓
Group4	18	✓		✓

Group 1: It contains 475 ransomwares with locking devices behaviour. If these ransomwares are activated, ransomwares will automatically reset passwords and lock devices. Taking ransomwares in Group 1 as test data can evaluate whether KRRecover can recover hijacked devices.

Group 2: It contains 25 ransomwares with hijacking screen resource. If these ransomwares are activated, ransomwares will automatically hijacking screen resource, for example, their interfaces continued until users pay ransom. Taking ransomwares in Group 2 as test data can evaluate whether KRRecover can recover hijacked screen resource.

Group 3: It contains 108 ransomwares with encrypting files. If these ransomwares are activated, ransomwares will automatically encrypt files. Users cannot open their files normally until they pay ransom. Taking ransomwares in Group 3 as test data can evaluate whether KRRecover can recover encrypted files.

Group 4: It contains 18 ransomwares with both locking devices and encrypting files. If these ransomwares are activated, ransomwares will automatically reset passwords, lock

devices and encrypt files. Taking ransomwares in Group 4 as test data can evaluate whether KRRecover can recover devices and files when locking devices appeared simultaneously with encrypting files.

6.2. Experiments

All the experiments run on a 12G memory, MSM8974 CPU model, 3931Amh battery capacity, Android 4.4 test device. The version of basic tools is SuperSU v2.79 and XposedInstaller for Android 4.0-4.4.

Before KRRecover is installed, the environment configuration should be set with three steps. Firstly, we got system root privileges with the help of SuperSU. Then we installed XposedInstaller and assign root permissions. Lastly, we installed the sandbox, put the configuration file in the specified location and activated the sandbox module.

To give a better evaluation of KRRecover, we will answer the following four questions. For each question, we will first design an experiment and give the results of the experiment. Then we will give a brief summarization of the experiment.

Q1: Can KRRecover automatically recover devices hijacked by ransomwares?

Q2: Can KRRecover automatically decrypt files encrypted by ransomwares?

Q3: Can KRRecover recover hijacked devices and encrypted files against hybrid ransomwares?

Q4: What is the efficiency of KRRecover?

6.2.1. RQ1: Can KRRecover Automatically Recover Devices Hijacked by Ransomwares?

In this experiment, we took ransomwares in Group 1 and Group 2 as the test dataset. We installed and activated 475 devices locking ransomwares and 25 screen resource hijacking ransomwares on an Android device. To avoid the interplay between different ransomwares, we uninstalled the former ransomware before we installed a new ransomware.

We found that KRRecover can automatically recover the device against 475 devices locking ransomwares and KRRecover can recover screen resource against 25 screen resource hijacking ransomwares. The recovery time is within 1s after each ransom behaviour has been activated. Take the devices locking ransomware 5669f7847448aa9214b465f94eb3e456 (MD5) as an example, Figure 5 shows what the device like after the ransomware was activated. We can see the device is disabled for users unless they input the correct password. Figure 6 shows the device after KRRecover has recovered the device. We can see the password is cancelled and users can use the device normally.

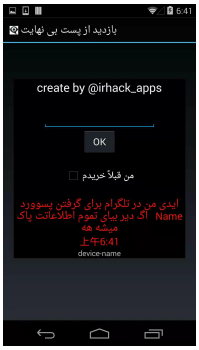

Figure 5. Locked Device.

Figure 6. Recovered Device.

Insight. 475 devices locking ransomwares in the dataset contain PIN locking ransomwares, password locking ransomwares and gesture locking ransomwares. 25 screen resource hijacking ransomwares contain setting interfaces as top level and keeping interfaces popping up. To some extent, KRRecover can recover hijacked screen resource and devices against ransomwares even they implement the function of locking devices and hijacking screen resource in many ways.

6.2.2. RQ2: Can KRRecover Automatically Decrypt Files Encrypted by Ransomwares?

In this experiment, we took ransomwares in Group 3 as the test dataset. We installed and activated 108 files encrypting ransomwares on an Android device. To avoid the interplay between different ransomwares, we uninstalled the former ransomware before we installed a new ransomware. For the convenience of the test, we wrote a *test.txt* file as the target file in the test device in advance.

We found that KRRecover can automatically recover files against 108 files encrypting ransomwares. Take the files encrypted ransomware f3a79953cc2e3babbc87ff44c2bc7031 (MD5) as an example. For the convenience of test, we used *test.txt* as a target file. Figure 7 shows the original file which could be opened with the test sentence The *txt* file is used to test ransomwares. After the ransomware was activated, as shown in Figure 8, the file became as *test.txt.enc*. We couldn't open the file normally and the content had already become disorder code. After KRRecover decrypted the file, as shown in Figure 9, the encrypted file *test.txt.enc* has been restored to *test.txt*. The file could be opened normally at this time and the content of the file was consistent compared with the original file.

Figure 7. Original Files.

Figure 8. Files Encryption.

Figure 9. Files Recovery.

Insight. We decompiled them as a manual verification. We found that these ransomwares implement the encryption function with the help of cryptoAPIs. KRRecover can decrypt files that are encrypted by cryptoAPIs.

6.2.3. RQ3:Can KRRecover Recover Hijacked Devices and Encrypted Files Against Hybrid Ransomwares?

In this experiment, we took ransomwares in Group 4 as the test dataset. We installed and activated 18 hybrid ransomwares on an Android device. These hybrid ransomwares can lock devices and encrypt files at the same time. To avoid the interplay between different ransomwares, we uninstalled the former ransomware before we installed a new ransomware.

We found that KRRecover can both recover locked devices and encrypted files against 18 hybrid ransomwares. Take the ransomware 154eab95dbb2c8134461996b14158bca (MD5) as an example, Figure 10 is the device interface after the ransomware was activated. The ransomware has hijacked the screen resource to distribute the user using the device and demand a ransom. Figure 11 is the device interface after KRRecover has released the hijacked resource. Figure 12 shows that file *test.txt* was encrypted as file *test.txt.enc* and couldn't be opened after the ransomware was activated. Figure 13 is the status of files after KRRecover has decrypted the encrypted files. We can see that the encrypted files had

been restored to *test.txt*. The file could be opened normally and the content of the file was consistent compared with the original file.

Figure 10. Encrypted Devices.

Figure 11. Recovered Device.

Figure 12. Files Encryption.

Figure 13. Files Recovery.

Insight. These hybrid ransomwares all can lock devices and encrypt files at the same time after they are activated. KRRecover can recover hijacked devices and encrypted files against hybrid ransomwares.

6.2.4. RQ4: What Is the Efficiency of KRRecover?

In this experiment, we evaluated the efficiency of KRRecover, including the time consumption of encrypted recovery and energy consumption. The size of KRRecover is 4.9 MB. It runs on a 12G memory, MSM8974 CPU model, 3931 Amh battery capacity, Android 4.4 test device.

To test the efficiency of files decryption, we randomly generated 200 files with different sizes in the test device and activated the ransomware.

Then we used KRRecover to decrypt files and recorded the decryption time of different files. Figure 14 shows the effect of different sizes of files decryption time. We can see from Figure 15 that the larger the file, the longer it takes to decrypt. For each 1 MB size increase, the decryption time needs to be increased by 20 s or so. For the size of a private file in device is usually 5 MB on average, the longest time of recovering a file is no more than 2 min.

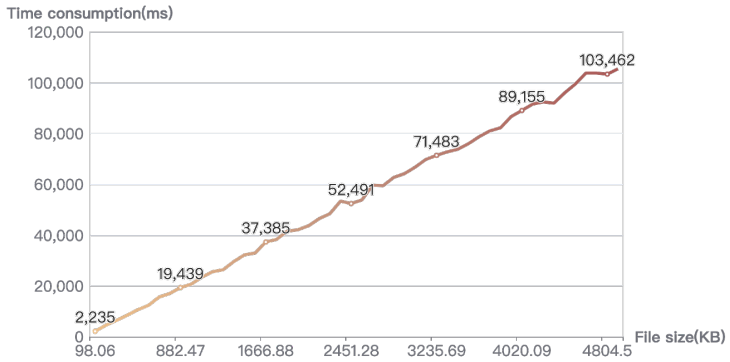

Figure 14. Decryption Time of Files with Different Sizes.

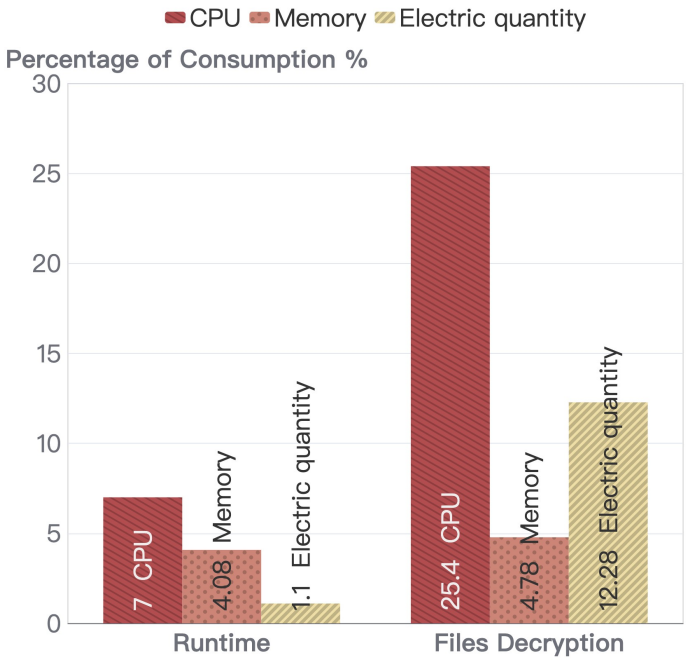

Figure 15. Energy Consumption of KRRecover.

To test the energy consumption of KRRecover, we recorded the CPU consumption, memory consumption and electric quantity of KRRecover at runtime. The details of energy consumption are shown in Figure 14. If KRRecover is in monitoring state, its CPU consumption is 7%, its memory consumption is 4.08% and its electric quantity is 1.1%. If KRRecover is decrypting files, its CPU consumption is 25.4%, its memory consumption is 4.78% and its electric quantity is 12.28%.

Insight. To our best knowledge, there is rare experiment about devices recovery tool and files recovery tools on Android. It is difficult for us to find a benchmark to do some comparison. However, compared with the economic losses caused by ransomwares, we believe that users could think the decryption time and energy consumption of KRRecover is acceptable. KRRecover can recover hijacked devices and encrypted files with little energy consumption in a short time.

7. Conclusions and Limitations

In this paper, we propose the scheme to recover hijacked screen resource, locked devices and encrypted files by ransomwares and implement an auto-recover tool on Android devices. KRRecover can recover hijacked screen resources, locked devices and encrypted files automatically even the USB debugging interfaces of their devices are closed. To our best knowledge, we are the first one to implement the application which can automatically recover locked devices and encrypted files. You can get KRRecover at https://github.com/FoxyWinner/KRRecover_Release (accessed on 20 September 2020).

In this work, we looked into how to automatically recover encrypted files by CryptoAPIs on Android. At present, we still can't recover files encrypted by custom encrypting methods or next generation encryption technologies [26]. In the future, we will devote to the investigation of recovering files encrypted by other encrypting methods on mobile devices. In the meantime, to our best knowledge, there is rare experiment about devices recovery tool and files recovery tools on Android. It is difficult for us to find a benchmark about the efficiency to do some comparison. We will consistently pay attention to different recovery tools on Android and give more efforts on it in the future.

Author Contributions: Conceptualization, S.W.; Data curation, Y.S.; Formal analysis, T.T.; Investigation, H.Z.; Methodology, S.W., S.Q. and H.Z.; Software, S.W. and J.H.; Supervision, N.H.; Writing—original draft, S.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key R&D Program of China under Grant No. 2018YFB0804703..

Institutional Review Board Statement: Not applicable..

Informed Consent Statement: Not applicable.

Data Availability Statement: You can get KRRecover at https://github.com/FoxyWinner/KRRecover_Release accessed on 5 May 2021.

Acknowledgments: This work was supported in part by the National Key R&D Program of China under Grant No. 2018YFB0804703.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Datto's Global State of the Channel Ransomware Report. [Online]. Available online: https://www.datto.com/resources/dattos-global-state-of-the-channel-ransomware-report (accessed on 20 December 2020)
2. Available online: https://www.coveware.com/blog/q2-2020-ransomware-marketplace-report (accessed on 30 December 2020).
3. McAfee Labs Threats Report. 2019. [Online]. Available online: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf (accessed on 15 March 2019).
4. Shao, Y.; Luo, X.; Qian, C. Rootguard: Protecting rooted android phones. *IEEE Comput.* **2014**, *47*, 32–40. [CrossRef]
5. Costamagna, V.; Zheng, C. Artdroid: A Virtual-Method Hooking Framework on Android Art Runtime. Available online: http://ceur-ws.org/Vol-1575/paper_10.pdf (accessed on 6 April 2016).
6. Available online: https://developer.android.com/about/dashboards/index.html (accessed on 13 Deccember 2019).
7. Lee, K.; Oh, I.; Yim, K. Ransomware-prevention technique using key backup. In *Big Data Technologies and Applications*; Jung, J.J., Kim, P., Eds.; Springer International Publishing: Cham, Switzerland, 2017; Volume 194, pp. 105–114.
8. Kolodenker, E.; Koch, W.; Stringhini, G.; Egele, M. PayBreak: Defense against cryptographic ransomware. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS 2017, Abu Dhabi, United Arab Emirates, 2–6 April 2017; ACM: New York, NY, USA, 2017; pp. 599–611.
9. Kharraz, A.; Robertson, W.; Balzarotti, D.; Bilge, L.; Kirda, E. Cutting the gordian knot: A look under the hood of ransomware attacks. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Milan, Italy, 9–10 July 2015; pp. 3–24..
10. Continella, A. Shieldfs: A self-healing, ransomware-aware filesystem. In Proceedings of the 32nd Annual Conference on Computer Security Applications, Los Angeles, CA, USA, 5–9 December, 2016; pp. 336–347.
11. Kharaz, A.; Arshad, S.; Mulliner, C.; Robertson, W.; Kirda, E. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. In Proceedings of the USENIX Security Symposium 2016, Austin, TX, USA, 10–12 August 2016.
12. Andronio, N.; Zanero, S.; Maggi, F. HelDroid: Dissecting and detecting mobile ransomware. In Proceedings of the International Symposium on Recent Advances in Intrusion Detection, Kyoto, Japan, 2–4 November 2015; pp. 382–404.
13. Scaife, N.; Carter, H.; Traynor, P.; Butler, K.R.B. Cryptolock (and drop it): Stopping ransomware attacks on user data. In Proceedings of the ICDCS, Nara, Japan, 27–30 June 2016; pp. 303–312.
14. Available online: https://developer.android.google.cn/studio/command-line/adb (accessed on 13 December 2019).
15. Available online: https://source.android.com/source/running#unlocking-the-bootloader (accessed on 4 January 2020).
16. Spreitzenbarth, M.; Freiling, F.; Echtler, F.; Schreck, T.; Hoffmann, J. Mobile-sandbox: Having a deeper look into android applications. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, Coimbra, Portugal, 18–22 March 2013.
17. Jiang, X.; Liu, M.; Yang, K.; Liu, Y.; Wang, R. A Security Sandbox Approach of Android Based on Hook Mechanism. *Secur. Commun. Netw.* **2018**, *2018*, 9856537. [CrossRef]
18. Chen, J.; Wang, C.; Zhao, Z.; Chen, K.; Du, R.; Ahn, G.J. Uncovering the Face of Android Ransomware: Characterization and Real time Detection. *IEEE Trans. Actions Inf. Forensics Secur.* **2018**, *13*, 1286–1300. [CrossRef]
19. Available online: http://amd.arguslab.org (accessed on 3 March 2019).
20. Arp, D.; Spreitzenbarth, M.; Hubner, M.; Gascon, H.; Rieck, K.; Siemens, C.E. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In Proceedings of the Network Distributed System Security Symposium 2014, San Diego, CA, USA, 23–26 February 2014.
21. Available online: https://repo.xposed.info/module/de.robv.android.xposed.installer (accessed on 15 March 2020).
22. Maiorca, D.; Mercaldo; Giacinto, F.; Visaggio, C.A.; Martinelli, F. R-PackDroid: API Package-Based Characterization and Detection of Mobile Ransomware. In Proceedings of the ACM Symposium on Applied Computing (SAC 2017–Acceptance Rate 15.7%), Marrakech, Morocco, 3–7 April 2017; pp. 1718–1723.
23. Available online: http://prag.diee.unica.it/it/RPackDroid (accessed on 13 December 2019).

24. Scalas, M.; Maiorca, D.; Mercaldo, F.; Visaggio, C.A.; Martinelli, F.; Giacinto, G. On the Effectiveness of System API-Related Information for Android Ransomware Detection. *Comput. Secur.* **2019**, *86*, 162–182. [CrossRef]
25. Available online: https://www.virustotal.com/ (accessed on 20 August 2018).
26. Mathew, S.; Satpathy, S.; Suresh, V.; Anders, M.; Kaul, H.; Agarwal, A.; Hsu, S.; Chen, G.; Krishnamurthy, R. 340 mV–1.1 V, 289 Gbps/W, 2090-Gate NanoAES Hardware Accelerator With Area-Optimized Encrypt/Decrypt GF(2 4) 2 Polynomials in 22 nm Tri-Gate CMOS. *IEEE J. Solid-State Circ.* **2015**, *50*, 1048–1058. [CrossRef]