Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 2946735, 9 pages
http://dx.doi.org/10.1155/2016/2946735

Research Article

The Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform

Sanggeun Song, Bongjoon Kim, and Sangjun Lee

School of Computing, Soongsil University, Sangdo-ro, Dongjak-gu, Seoul 06978, Republic of Korea

Correspondence should be addressed to Sangjun Lee; sangjun@ssu.ac.kr

Received 31 December 2015; Accepted 10 March 2016

Academic Editor: Seung Yang

Due to recent indiscriminate attacks of ransomware, damage cases including encryption of users' important files are constantly increasing. The existing vaccine systems are vulnerable to attacks of new pattern ransomware because they can only detect the ransomware of existing patterns. More effective technique is required to prevent modified ransomware. In this paper, an effective method is proposed to prevent the attacks of modified ransomware on Android platform. The proposed technique specifies and intensively monitors processes and specific file directories using statistical methods based on Processor usage, Memory usage, and I/O rates so that the process with abnormal behaviors can be detected. If the process running a suspicious ransomware is detected, the proposed system will stop the process and take steps to confirm the deletion of programs associated with the process from users. The information of suspected and exceptional processes confirmed by users is stored in a database. The proposed technique can detect ransomware even if you do not save its patterns. Its speed of detection is very fast because it can be implemented in Android source code instead of mobile application. In addition, it can effectively determine modified patterns of ransomware and provide protection with minimum damage.

1. Introduction

Ransomware [1] is a type of malware that uses malicious codes to intrude the system before users notice it, to encrypt important files, to require money using encrypted files as a hostage, and to give monetary damages to users. The rapid growth of the mobile market has been the main target of hackers to obtain illegal gains by using ransomware. The market share of Korea's Android OS is approximately 80% of the total share of smartphone market as shown in Table 1. Compared to other OS such as iOS, Windows Phone, or Blackberry, Android holds a high market share close to monopoly, while the others combined have less than 15% share in the mobile device market [2]. The share of the Android platform is so high that the platform is the main target of ransomware attacks. Damage cases of Android-based smartphones are continuously growing recently.

Traditional vaccine system can detect a system if it is infected with ransomware and cure it. However, it cannot prevent attacks by ransomware without obtaining information on the ransomware. In addition, files cannot be recovered without the encryption key because files are already encrypted even if the traditional vaccine system can remove the ransomware [3]. Users can avoid infections by updating the vaccine system from time to time. However, this method has limited efficacy. Existing vaccine system can detect ransomware using intrusion detection method based on files [4]. However, this approach cannot detect modified ransomware with new patterns because it can only prevent ransomware based on analysis information of the ransomware. Therefore, an active instead of a passive prevention method is urgently required.

In this paper, a ransomware prevention technique on Android platform is proposed. The proposed method can monitor file events that occurred when the ransomware accesses and copies files. This technique can detect and remove the ransomware using the CPU and I/O usage as well as the information stored in the DB. This proposed method can detect modified patterns of ransomware without obtaining information about the ransomware. In addition, it can be implemented on the kernel and framework source of Android so that it can detect ransomware relatively

Table 1: Smart device operating system market share [2].

Period	Android	iOS	Windows Phone	Blackberry OS	Others
2015	82.8%	13.9%	2.6%	0.3%	0.4%
2014	84.8%	11.6%	2.5%	0.5%	0.7%
2013	79.8%	12.9%	3.4%	2.8%	1.2%
2012	69.3%	16.6%	3.1%	4.9%	6.1%

faster than other programs that run at the application level. Furthermore, it can continuously monitor the ransomware without separately downloading or updating.

The remainder of this paper is organized as follows. Related work is briefly discussed in Section 2. Our proposed approach is described in Section 3. Evaluation of the proposed approach is given in Section 4. Finally, several concluding remarks are given in Section 5.

2. Related Research

2.1. Ransomware. Ransomware spreading methods are similar to those of malicious code Trojan Horse [5] that contains malicious routine and pretends as a normal program. Ransomware intrudes into users' devices after pretending as a normal application such as Trojan Horse. Ransomware restricts the use of the system in various ways after intruding the system. It is mainly classified into the following three types: Scareware, Lock-Screen, and Encrypting [6].

(i) Scareware. It informs users that the device has been infected with malicious codes. It suggests the purchase of fake antivirus programs to treat them. It finally extorts money from the user.

(ii) Lock-Screen. It disables users' PC in any way. It locks the system so that the users are not able to run the operating system when executing the system. When a user runs his system, it disables the operating system and sends the message that your PC has illegal contents that you will be fined by impersonating FBI or government agencies.

(iii) Encrypting. This is the most serious type of ransomware. It prevents the use of important files in your device by encrypting them. It extorts money by encrypting users' files in PCs and letting users deposit the ransom for files to a virtual account to decrypt.

Ransomware accesses users and gives damage to them in various ways. For example, CryptoLocker [7] can encrypt files in PC. Reveton [8, 9] will impersonate law enforcement agencies such as FBI. SimpleLocker [10] targets smartphone users of the Android environment. This ransomware can be serious security threat to cloud computing [11] as it becomes the basic infrastructure of information system.

2.2. Existing Techniques

2.2.1. Process Using Hash Information. The processing method of CryptoLocker is to compare Hash information. CryptoLocker generates files encrypted with ".encrypted" [12]. The encrypted files are then added to the Hash Information. Signature, Public Key values, and their sizes will increase. Recovery tools are generally used to process CryptoLocker. They include different decryption key index information by infected users. Recovery tools compare Hash information and encrypted files in the data files, confirm the validation of key from key index information stored therein, and then proceed to decoding [13].

By looking at encrypted files' recovery methods used in existing vaccines, these methods obtain a sample by decompiling the ransomware and perform decryption using the decryption key found by the code analysis of the sample [14]. There is a risk that when a new ransomware appears, users have to wait until a security company finds the decryption key value through sample analysis. Intelligent sensing techniques are required to detect new patterns of ransomware because ransomware constantly threatens the safety of mobile device.

System-based behavior detection technique [15] is based on the detection of occurrences of several behaviors in a computer system. It performs "integrity checking" and "behavior blocking" [16, 17]. Integrity checking technique conducts frequent inspections in order to confirm the integrity of the computing system. This approach calculates and writes the Hash values for execution files and directories on a clean computer system that is not infected by malware. Behavior blocking technique monitors all actions within the computer system. When a suspicious action occurs in similar way of malicious infections, this approach tracks the cause of executable file and blocks the execution of a suspicious action so that it has no progress.

2.2.2. Process Using CPU and I/O Usage. Statistical technique is one malware analysis technique that detects abnormal behaviors by analyzing the resources of the system. NIDES (Next-generation Intrusion Detection Expert System) [18] of SRI (Stanford Research Institute) International is a typical system based on statistical techniques. NIDES sets a goal of detecting abnormal behaviors that occurred in the system with a profiling technique after collecting Processor usage, I/O rate, Memory usage, and so forth, over a long time. Korea Electronics and Telecommunications Research Institute uses the technique using the mean difference of CPU or Memory usage in order to provide a reliable service on the host [19, 20]. However, this technique only operates against the attacks of DDoS. In this paper, a technique is proposed to prevent the intrusion of ransomware on Android platform based on statistical methods using Process, Memory, and Storage I/O usage.

2.3. Android Application Permissions Analysis. Android market applications demand Android system permissions in order to perform the correct operation. Applications registered in an official store show users permission requirements when they are downloaded. However, ordinary users may

Table 2: Difference in permission between Ransomware App and Normal App [21, 22].

Type	Permission	Behavior	Ransomware App	Normal App
System	GET_TASK	Allows an application to get information about currently or recently running tasks	O	O
	WRITE_SETTINGS	Allows an application to read or write system settings	O	O
	SYSTEM_ALERT_WINDOW	Allows an application to alert system	O	O
	RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcasted after the system finishes booting	O	X
	READ_PHONE_STATE	Allows read only access to phone state	O	X
	READ_EXTERNAL_STORAGE	Allows an application to read from external storage	O	X
	WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage	O	X
	WAKE_LOCK	Allows using PowerManager WakeLocks to keep Processor from sleeping or screen from dimming	O	X
	GET_ACCOUNTS	Allows access to the list of accounts in Accounts Service	O	X
	BIND_DEVICE_ADMIN	Must be required by device administration receiver to ensure that only the system can interact with it	O	X
	DISABLE_KEYGUARD	Allows applications to disable the keyguard if it is not secure	O	X
SMS	RECEIVE_SMS	Allows an application to receive SMS messages	O	O
	SEND_SMS	Allows an application to send SMS messages	O	O
	READ_SMS	Allows an application to read SMS messages	O	X
Contact	READ_CONTACTS	Allows an application to read user's contacts data	O	O
	READ_CALL_LOG	Allows an application to read the user's call log	O	O
	CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call	O	O
Network	INTERNET	Allows applications to open network sockets	O	X
	ACCESS_NETWORK_STATE	Allows applications to access information about networks	O	X
	READ_HISTORY_BOOKMARKS	Allows an application to read the user's browsing history and bookmarks	O	X

unintentionally download or run applications without carefully looking at them. Ransomware distributors will distribute the ransomware and pretend as a normal application on an official store using this security weakness.

To design the proposed method, the different kinds and functions of permissions on the Android system and permissions needed by ransomware are analyzed. Permissions to adversely affect the Android system are largely classified as System, SMS, Contact, and Location [21]. Difference in permissions between Ransomware App and Normal App is shown in Table 2. A total of 14 kinds of ransomware that appeared between 2014.01 and 2015.09 based on the report of virustotal [22] are included in the comparison (Table 2).

The functions of the corresponding permissions are not necessarily safe. These permissions access a lot of information, including the configuration information of the device, the list of applications, resource statistics, and personal information such as location information and SMS information. Normal applications use these permissions. Therefore, users generally agree to install applications without doubt, even when it is the ransomware that requires permissions for the System, SMS, Contact, and Location.

3. The Proposed Technique

To have efficient implementation, the proposed technique is designed with three modules: Configuration, Monitoring, and Processing (Figure 1). Configuration module generates a monitoring list table for a smooth operation of the proposed method. It is the module for the initial setting. Monitoring module is responsible for monitoring Processor, Memory, and Storage I/O usages of every process in real time based on statistical techniques. Finally, processing module determines the handling of the process suspected as ransomware by the Monitoring module and makes an exception or isolation of the process.

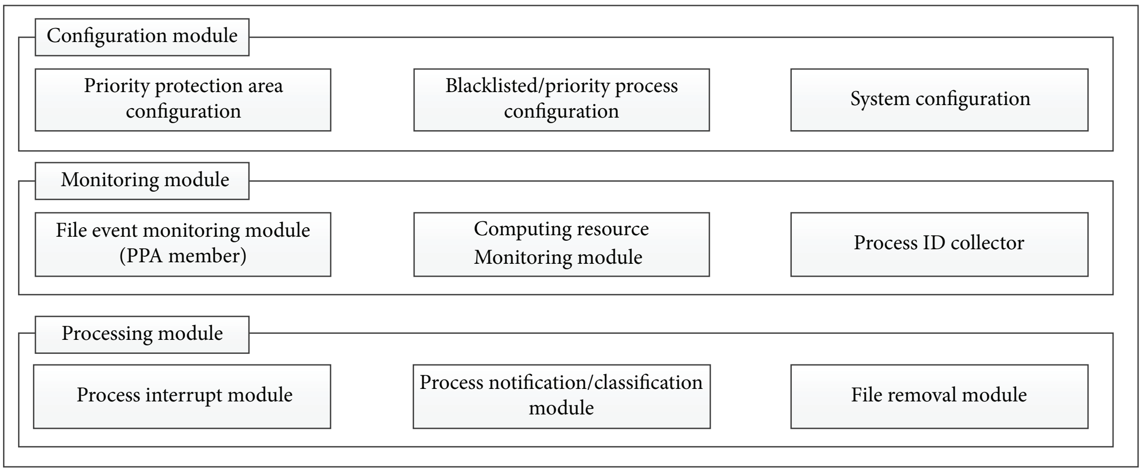

Configuration module
Priority protection area configuration
Blacklisted/priority process configuration
System configuration
Monitoring module
File event monitoring module (PPA member)
Computing resource Monitoring module
Process ID collector
Processing module
Process interrupt module
Process notification/classification module
File removal module

Figure 1: Overview of the proposed system.

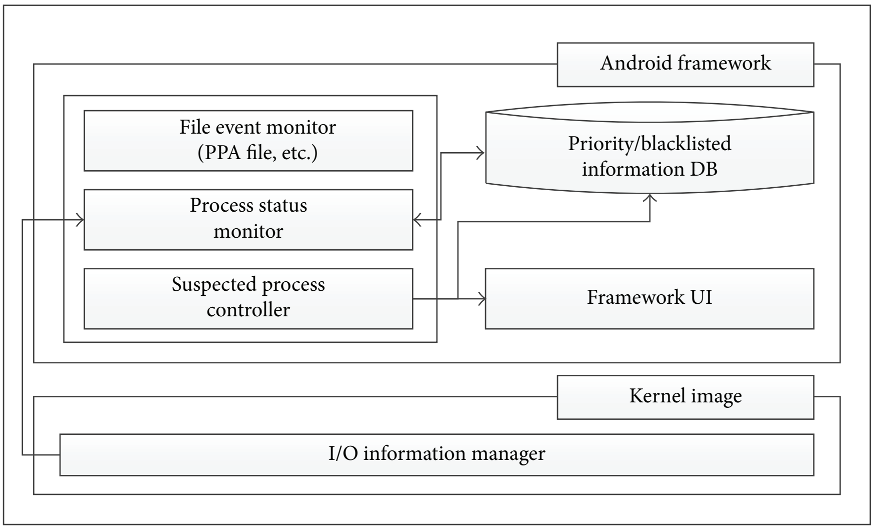

Android framework
File event monitor (PPA file, etc.)
Process status monitor
Suspected process controller
Priority/blacklisted information DB
Framework UI
Kernel image
I/O information manager

Figure 2: Proposed system on Android platform.

The proposed technique implements the configuration module, the monitoring module, and the processing module using the framework and kernel of the Android platform as shown in Figure 2. In addition, user's UI part added to the Android Settings and the database used in the configuration module are implemented within the framework. In the kernel, a part for generating I/O information to monitor the process is implemented, through which a kernel image is produced.

Algorithm 1 shows the operation flow of the basic technique proposed in this paper. Details are described later in different topics.

```
begin
  Input: process id P
  ProcessInfo ← ProcessDatabase(P);
  if ProcessInfo is blacklist then
    KillProcess(P);
    ProcessRemovalProcedure(P);
    return;
  else if ProcessInfo is not priority protection member then
    enqueueMonitoringProcessID(P);
  end
end
```

Algorithm 1: The main procedure of proposed technique.

3.1. Configuration Module. The configuration module is the basic setup to be applied when the proposed technique detects a ransomware. In this paper, default setting values that are information about the process or application installed by default on the Android platform are saved in a database. The foremost role of the configuration module is to specify the location of the files needed to be protected from the attacks of the ransomware. An area of these important files is called priority protection area (hereinafter PPA). If the proposed technique is run correctly, it will collect the information of PPA, register them to the watch list table for the monitoring module, and protect the corresponding files in real time. The second role is to register user's handling for the suspected process detected by the monitoring module into the database and maintain the handling. If the user finally determines the process as a ransomware, it stores the information of the corresponding process. It will automatically detect and delete the process depending on the user's feedback. If the user determines the process as normal, it records the information of the process and forces the system to maintain the process without terminating the process even if the process is redetected.

3.2. Monitoring Module.

The monitoring module is responsible for detecting the ransomware by monitoring the PPA area and the process. The monitoring module is largely composed of two modules (file monitoring and process monitoring) based on the roles.

(i) File Monitoring Module. It continuously monitors the status of the input/output events such as reading, writing, copying, and deleting of a file belonging to a PPA set in the configuration module and detects the attacks of the ransomware. Algorithm 2 shows the operation flow of the file monitoring module proposed in this paper.

(ii) Process Monitoring Module. It continuously monitors Processor share by Process, Memory usage, I/O count, Storage I/O count, and so forth and detects the ransomware. Algorithm 3 shows the operation flow of the process monitoring module proposed in this paper.

Upon detecting the suspected process, it also handles malicious or exceptional processes in the database applied in the configuration module. For the process registered as a malicious process, the monitoring module will stop the process at the moment of detection and automatically delete the process. For the process registered as an exceptional process, it will allow the normal execution because it is specified as safe by the user.

3.2.1. File Event Monitoring.

The monitoring module monitors the modification and deletion events of files and directories existing in a PPA. Monitoring path is generally through external storage of the device. Basic monitoring path is shown in Table 3.

Observer is arranged to monitor file events in each directory. File event monitoring using Observer is based on the patterns of ransomware to generate encrypted files after reading and writing target files and deleting original files. Observer can detect events of ransomware deleting and modifying files without obtaining data on the ransomware. Observer is responsible for monitoring modification and deletion events that occurred in each path while the device is on. If the event for the file occurs, Observer will pass the file event information to the monitoring module and find which process is the one that produced the event. The process found by the Observer is primarily checked through an exceptional handling process. If the process is not in a list, it is determined as a process suspicious of ransomware. The process will be stopped first. The technique inquires the user about subsequent handling of the process. Depending on the user's determination, the handling of the process in the database will be updated and managed.

```
Begin
  Input: process ids $P_n$
  While
    for all process id $P_i$ do
      Flag ← isOccuriedEventInPPA($P_i$);
      if Flag is enabled then
        KillProcess($P$);
        Result ← ProcessNotification($P_i$);
        if Result is block then
          ProcessRemovalProcedure($P_i$);
        end
        addProcessDatabase($P_i$);
      End
    end
  End
End
```

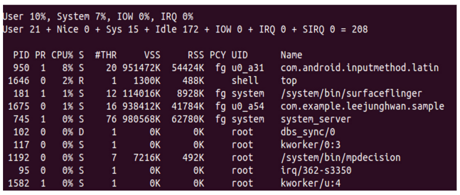
Algorithm 2: The file monitoring module of proposed technique.

```
begin
  Input: process ids $P_n$, Threshold $T$
  while
    for all process id $P_i$ do
      ProcessInfo ← getProcessInformation($P_i$);
      if ProcessInfo has occupied resources then $T$ then
        KillProcess($P$);
        Result ← ProcessNotification($P_i$);
        if Result is block then
          ProcessRemovalProcedure($P_i$);
        end
        addProcessDatabase($P_i$);
      end
    end
  end
end
```

Algorithm 3: The process monitoring module of proposed technique.

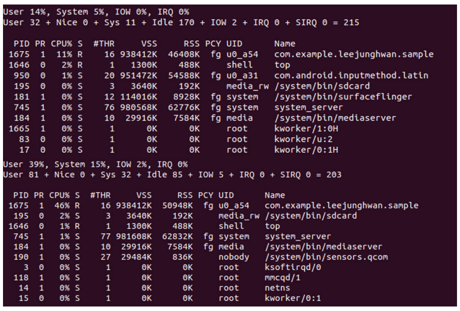
Table 3: Basic monitoring path summary.

Location	Path
Android	/sdcard/Android/com
Picture	/sdcard/Pictures
DCIM	/sdcard/DCIM
Downloads	/sdcard/Downloads
Music	/sdcard/Music
Movies	/sdcard/Movies



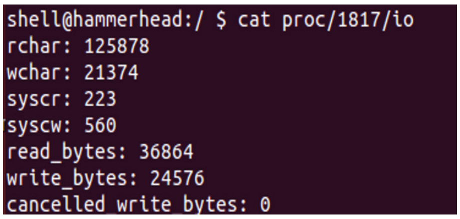
```
User 10%, System 7%, IOW 0%, IRQ 0%
User 21 + Nice 0 + Sys 15 + Idle 172 + IOW 0 + IRQ 0 + SIRQ 0 = 208

  PID PR CPU% S  #THR     VSS     RSS PCY UID      Name
  950  1   8% S    20 951472K  54424K  fg u0_a31   com.android.inputmethod.latin
 1646  0   2% R     1   1300K    488K     shell    top
  181  1   1% S    12 114016K   8928K  fg system   /system/bin/surfaceflinger
 1675  0   1% S    16 938412K  41784K  fg u0_a54   com.example.leejunghwan.sample
  745  1   0% S    76 980568K  62780K  fg system   system_server
  102  0   0% D     1      0K      0K     root     dbs_sync/0
  117  0   0% S     1      0K      0K     root     kworker/0:3
 1192  0   0% S     7   7216K    492K     root     /system/bin/mpdecision
   95  0   0% S     1      0K      0K     root     irq/362-s3350
 1582  1   0% S     1      0K      0K     root     kworker/u:4
```

Figure 3: Process status of the process when a latent ransomware is carried out.



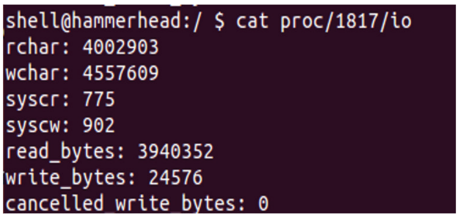
```
User 14%, System 5%, IOW 0%, IRQ 0%
User 32 + Nice 0 + Sys 11 + Idle 170 + IOW 2 + IRQ 0 + SIRQ 0 = 215

  PID PR CPU% S  #THR     VSS     RSS PCY UID      Name
 1675  1  11% R    16 938412K  46408K  fg u0_a54   com.example.leejunghwan.sample
 1646  0   2% R     1   1300K    488K     shell    top
  950  0   1% S    20 951472K  54588K  fg u0_a31   com.android.inputmethod.latin
  195  0   0% S     3   3640K    192K     media_rw /system/bin/sdcard
  181  1   0% S    12 114016K   8928K  fg system   /system/bin/surfaceflinger
  745  1   0% S    76 980568K  62776K  fg system   system_server
  184  1   0% S    10  29916K   7584K  fg media    /system/bin/mediaserver
 1665  1   0% S     1      0K      0K     root     kworker/1:0H
   83  0   0% S     1      0K      0K     root     kworker/u:2
   17  0   0% S     1      0K      0K     root     kworker/0:1H
User 39%, System 15%, IOW 2%, IRQ 0%
User 81 + Nice 0 + Sys 32 + Idle 85 + IOW 5 + IRQ 0 + SIRQ 0 = 203

  PID PR CPU% S  #THR     VSS     RSS PCY UID      Name
 1675  1  46% R    16 938412K  50948K  fg u0_a54   com.example.leejunghwan.sample
  195  0   2% S     3   3640K    192K     media_rw /system/bin/sdcard
 1646  0   1% R     1   1300K    488K     shell    top
  745  1   1% S    77 981608K  62832K  fg system   system_server
  184  1   0% S    10  29916K   7584K  fg media    /system/bin/mediaserver
  190  1   0% S    27  29484K    836K     nobody   /system/bin/sensors.qcom
    3  0   0% S     1      0K      0K     root     ksoftirqd/0
  118  1   0% S     1      0K      0K     root     mmcqd/1
   14  1   0% S     1      0K      0K     root     netns
   15  0   0% S     1      0K      0K     root     kworker/0:1
```

Figure 4: Process status when a latent ransomware performs active encrypting.



```
shell@hammerhead:/ $ cat proc/1817/io
rchar: 125878
wchar: 21374
syscr: 223
syscw: 560
read_bytes: 36864
write_bytes: 24576
cancelled_write_bytes: 0
```

Figure 5: I/O status when the ransomware is latent.



```
shell@hammerhead:/ $ cat proc/1817/io
rchar: 4002903
wchar: 4557609
syscr: 775
syscw: 902
read_bytes: 3940352
write_bytes: 24576
cancelled_write_bytes: 0
```

Figure 6: I/O status when the ransomware is active.

3.2.2. Process Monitoring. The monitoring module uses information such as Processor share for each Process, Memory usage, I/O count, and Storage I/O count to detect suspected process among running processes. It also detects suspected process through monitoring file events. The operation of Process monitoring is based on the information of malicious/exceptional processes stored in the database by the configuration module. It takes advantage of the fact that ransomware process uses a lot of system resources in the process of encrypting files in the storage. The proposed technique checks whether Processor share, Memory usage, I/O count, and Storage I/O count are more than a threshold value based on statistical methods. It will transmit the information of the corresponding process into the Processing module when it is higher than a threshold value.

To prove the change of Processor usage, a sample ransomware is run. The name of the corresponding ransomware process is called "com.example. ***.Sample". Figure 3 shows the status of the process when a latent ransomware process is carried out. Processor share shows 0-1% so that users will not notice it.

Figure 4 shows the situation when the latent ransomware performs encrypting files in earnest. The Processor usage is changed to 11% at the moment of the encryption. It peaked at 46%.

The process of I/O usage of the same ransomware at latency is shown in Figure 5. The top of the figure is the initial stage of the process execution. The amount of bytes read is relatively small.

The process of I/O usage of the same latent ransomware that performs active encrypting is shown in Figure 6. As shown in Figure 6, the file I/O usage is sharply increased because the data of the files to be encrypted are read and written in earnest. The sharp increase in the CPU usage and I/O usage can be used to detect the ransomware.

3.3. Processing Module. The processing module forcibly stops the process suspicious of ransomware in the monitoring module and inquires users about the appropriate handling of the process. Once the handling is determined, the information of the corresponding process will be stored in the database and used in the configuration module subsequently. Database table structure used in the processing module is shown in Table 4.

ID is used to place the number of each tuple. PackageName is the name of an application. RiskType is a flag to determine whether it is safe/unsafe. Comment is prepared in case a separate explanation is needed.

The processing module also warns users about the risk of the ransomware through Android permission analysis.

(i) System Permission. The ransomware has permissions of the system. It seizes permissions of the device's administrator and prevents users from manipulating the device. This permission involves the risk of ransomware browsing the user's personal files stored in the device without user's permission. It uses administrator's permission.

(ii) SMS Permission. While a normal application provides convenience to users with SMS permission, the ransomware intercepts received messages to use them for illegal purposes by using SMS permission.

(iii) Contract Permission. Permission to access contacts is stored in the device. Typical examples of making ill use of this permission are phishing and smishing.

(iv) Network Permission. Permission to automatically find network connected to the device and allow the ransomware to operate. Ransomware seizes permission of the device so that

Table 4: Database table structure used in the processing module.

Table name	Column name	Data type	NULL constraint	Primary key
IDSDB	ID	INTEGER	NOT NULL	PK
	PackageName	VARCHAR	NOT NULL	PK
	RiskType	Integer	NOT NULL	
	Comment	VARCHAR		

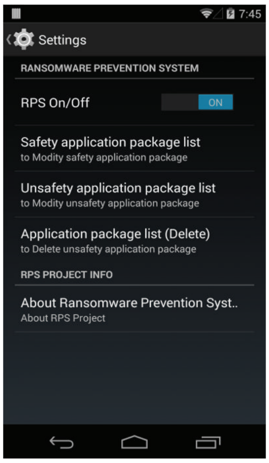
Table 5: Concerns of permission.

Permission	Concerns
GET_TASK	Rooting
WRITE_SETTINGS	Rooting
SYSTEM_ALERT_WINDOW	Rooting
RECEIVE_BOOT_COMPLETED	Rooting
READ_PHONE_STATE	Rooting
READ_EXTERNAL_STORAGE	Rooting, file accessing
WRITE_EXTERNAL_STORAGE	Rooting, file accessing
WAKE_LOCK	Rooting
READ_CONTACTS	Voice phishing, smishing, data spill
READ_CALL_LOG	Voice phishing, smishing, data spill
INTERNET	Rooting, cracking
ACCESS_NETWORK_STATE	Rooting, cracking
RECEIVE_SMS	Spying on sms, smishing, data spill
SEND_SMS	Spying on sms, Smishing, data spill
READ_SMS	Spying on sms, Smishing, data spill

users cannot operate the device. It has the risk of intercepting user's personal information stored in the device.

The processing module inquires of users about whether to keep or delete the corresponding program after stopping the process suspicious of ransomware. If the user shows his intention to delete the application, when the same process appears later, it is automatically removed without asking about user's thoughts because the user recognizes the corresponding application as ransomware. If he determines the process as normal, its safety is guaranteed so the process will not be forcibly stopped by the proposed technique. In addition, the proposed technique will let the user know if any part of the process is vulnerable. Concerns of permission are listed in Table 5.

3.4. User Interface. User Interface shown in Figure 7 provides users with easy access to the proposed method. UI is equipped with a basic format of the Android. It provides an interface of the configuration module. The proposed system functions can be turned on and off at any time using the corresponding interface. At the bottom, the names of the packages registered in the database so far can be checked. Addition, modification, and deletion of the information stored are also possible.



7:45

Settings

RANSOMWARE PREVENTION SYSTEM

RPS On/Off ON

Safety application package list
to Modity safety application package

Unsafety application package list
to Modity unsafety application package

Application package list (Delete)
to Delete unsafety application package

RPS PROJECT INFO

About Ransomware Prevention Syst..
About RPS Project

Figure 7: Implemented user interface.

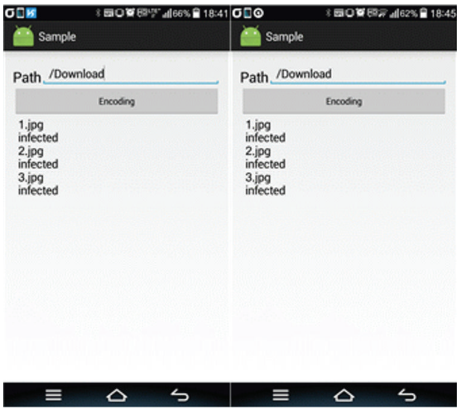
4. Evaluation

Unknown ransomware is used for evaluation of the proposed method compared to existing vaccine systems. A ransomware that encrypts files with 40-byte keys using the AES algorithm was made for testing. This sample ransomware has the function of opening all files on the input path and encrypting.

On the left of Figure 8 is the running result of a testing ransomware after running V3 Mobile one vaccine system that is famous in South Korea. On the right of Figure 8 is the result after running Avast made in the Czech Republic. These vaccine programs have no information about the new ransomware. They failed to detect the unknown ransomware. Therefore, files on /Download are encrypted. It is impossible to cure them either because there is no decryption key value.

In order to verify the proposed technique, PPA was set as /Download directory using the configuration module. Figure 9 shows the result of running the same sample ransomware after activating the proposed technique. In the device using the proposed method, users' files were protected because it found the ransomware before the encryption. Therefore, it stopped the ransomware process and asked about users' thoughts on deletion.

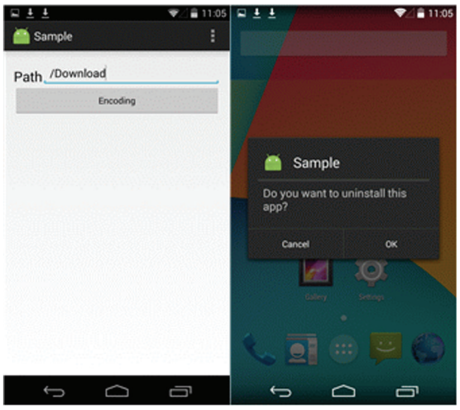
Results of evaluation of existing vaccine systems compared to the proposed technique are shown in Table 6. The proposed technique can deal with modified or new patterns of ransomware because it can detect ransomware using information such as Processor share, Memory usage or I/O count, and Storage I/O count. However, existing techniques need information of the ransomware to detect it. While traditional vaccines require updating the detection pattern



1.jpg
infected
2.jpg
infected
3.jpg
infected

1.jpg
infected
2.jpg
infected
3.jpg
infected

FIGURE 8: Running results of ransomware on existing vaccines (Avast, Ahnlab).



Do you want to uninstall this app?

FIGURE 9: Running results of ransomware using the proposed technique.

TABLE 6: Evaluation of existing vaccine systems compared to the proposed technique.

	Our method	V3	Avast
Protecting ransomware without ransomware info	O	X	X
Operating without updating	O	X	X
Operating without downloading	O	X	X
Operating without executing application	O	X	X

from time to time, the proposed method does not need so many updates because it can detect the ransomware based on its behavior. It does not need to install an application such as existing vaccines as it is implemented in the Android source. In this study, we found a slightly degraded performance of the device after using the proposed technique in order to protect sensitive information.

5. Conclusion

In this paper, a technique is proposed to reduce damage caused by unknown ransomware attacks on Android devices. The proposed method can effectively reduce damage caused by ransomware with modified or new patterns without obtaining information on the ransomware. It uses file input/output events and Processor status information based on the behavior of ransomware, unlike existing techniques that need information about the ransomware. It can automatically prevent damage caused by such ransomware attacks later based on information collected on the detected ransomware. It is possible to use the proposed method in all Android-based smart phones because this technique is added to the open source of Android source file. This technique is expected to allow users to minimize damage caused by attacks of ransomware that existing vaccine systems fail to detect.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Korea government (MSIP) (NRF-2015R1D1A1A01057680).

References

[1] X. Luo and Q. Liao, "Ransomware : a new cyber hijacking threat to enterprises," in *Handbook of Research on Information Security and Assurance*, IGI Global, 2009.

[2] "Worldwide Quarterly Mobile Phone Tracker," IDC, August 2015, http://www.idc.com/tracker/showproductinfo.jsp?prod_id=37.

[3] TREND Micro, *Ransomware Definition—Security Intelligence*, TREND Micro, Irving, Tex, USA, 2015, http://www.trendmicro.com/.

[4] D. Kim and S. Kim, "Design of quantification model for ransom ware prevent," *World Journal of Engineering and Technology*, vol. 3, no. 3, pp. 203–207, 2015.

[5] D. Lim, "Treats and countermeasures of malware," *Journal of IT Convergence Society for SMB*, vol. 5, no. 1, pp. 13–18, 2015.

[6] N. Andronio, S. Zanero, and F. Maggi, "HelDroid: dissecting and detecting mobile ransomware," in *Research in Attacks, Intrusions, and Defenses*, vol. 9404 of *Lecture Notes in Computer Science*, pp. 382–404, Springer, 2015.

[7] A. Beuhring and K. Salous, "Beyond blacklisting: cyberdefense in the era of advanced persistent threats," *IEEE Security & Privacy*, vol. 12, no. 5, pp. 90–93, 2014.

[8] P. Ducklin, "Reveton/FBI ransomware—exposed, explained and eliminated," NakedSecurity, August 2012, https://nakedsecurity.sophos.com/.

[9] J. Milletary, "Citadel Trojan Malware Analysis," Dell Secure Works Counter Threat Unit™ Intelligence Services, Dell Secure Works, September 2012.

[10] T. M. Marengereke and K. Sornalakshmi, "Cloud based security solution for android smartphones," in *Proceedings of the IEEE*

International Conference on Circuit, Power and Computing Technologies (ICCPCT '15), pp. 1–6, Nagercoil, India, March 2015.

[11] Y. Liu, Y. L. Sun, J. Ryoo, S. Rizvi, and A. V. Vasilakos, "A survey of security and privacy challenges in cloud computing: solutions and future directions," *Journal of Computing Science and Engineering*, vol. 9, no. 3, pp. 119–133, 2015.

[12] Ahnlab Security Issue, *How to Attack Us?, Ransomware 'CryptoLocker' That Hit South Korea*, 2015 (Korean), http://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?menu_dist=2&seq=23630.

[13] Ahnlab Security Report, "The latest mobile ransomware app and countermeasures," vol. 65, July 2015 (Korean), http://www.ahnlab.com/kr/site/securityinfo/asec/asecView.do?groupCode=VNI001&seq=23834.

[14] Ahnlab ASEC blog, "The ransomware that impersonate," NSB (National Security Bureau), Febuary 2015 (Korean), http://asec.ahnlab.com/1025.

[15] M. E. Wagner, *Behavior Oriented Detection of Malicious Code at Run-Time*, Florida Institute of Technology, 2004.

[16] P. Szor, *The Art of Computer Virus Research and Defense*, Symantec Press; Addison-Wesley Professional, 2005.

[17] J. Aycock, *Computer Viruses and Malware*, vol. 22, Springer Science & Business Media, 2006.

[18] D. Anderson, T. Frivold, and A. Valdes, "Next-generation intrusion detection expert system (NIDES): a summary," Tech. Rep. SRI-CSL-95-07, SRI International, Computer Science Laboratory, 1995.

[19] K. Daewon, "Automated Control Method and Apparatus of DDoS Attack Prevention Policy using The Status of CPU and Memory," Electronics and Telecommunications Research Institute(South Korea), US Patent, US 2012/0054823 A1, 2012.

[20] J. L. Lee and C. S. Hong, "Nonparametric detection methods against DDoS attack," *The Korean Journal of Applied Statistics*, vol. 26, no. 2, pp. 291–305, 2013.

[21] "System Permissions," API Guide, Android Developers, http://developer.android.com/intl/ko/guide/topics/security/permissions.html.

[22] virustotal, https://www.virustotal.com/en-gb/.