International Journal of Security and Its Applications
Vol. 9, No. 5 (2015), pp. 61-68
http://dx.doi.org/10.14257/ijsia.2015.9.5.06

Secure Data Sanitization for Android Device Users

Na Huang, Jingsha He and Bin Zhao

School of Software Engineering
Beijing University of Technology
Beijing 100124, China

Abstract

To protect the data confidentiality of Android device users, this paper proposed a user-level secure data sanitization method which can make sensitive data stored in flash memory unrecoverable. The proposed method using reversed-scanning algorithm to reduce time and it don't need overwriting or erase the whole flash memory. After sensitive data pages are found, the program targetedly overwriting some blocks, erase them, and insert them into free block list waiting to be located again which means the blocks will be twice overwrited by new data.

Keywords: *Android Device, Confidentiality, Sensitive, Secure, Sanitization*

1. Introduction

Owing to the growing popularity of Android smartphones and mobile application, more and more sensitive personal information is being stored in smartphone flash memory. People create huge data in everyday life and they usually do normal delete operation to the sensitive personal data thinking these data will not be found by others wishfully. Thus far, however, many studies have developed data recovering methods for flash memory which has two main types, NAND and NOR. Among them, the NAND flash is widely used in Android smart devices for its high read-write speed and it usually load Yaffs2 file system. The normal delete operation in Yaffs2 will not erase user data from flash memory indeed, which is completed by function yaffs_DeleteChunk (yaffs_Device * dev, int chunkId, int markNAND, int lyn).

The basic unit of writing operation for NAND flash is one page while data has to be erased in blocks. There is a limit on the number of times that NAND flash blocks can be reliably programmed and erased. A technique known as wear ensures that all physical blocks are exercised uniformly. To maximize the life of NAND, it is critical to implement both wear leveling and garbage collection. An effective strategy is out-of-place updates in which new data cannot be overwritten at the same location occupied by old data [7]. As we can see, when a block is made up of only deleted chunks, that block can be erased and reused. However, YAFFS2 will invoke the garbage collection process only when most of the flash chip is used and there is data waiting to be written. The strategies above are not friendly to user data secure since deleted files still exist in NAND and can thus be recovered [7]. Secure deletion is the act of deleting data from a storage medium such that the data is afterwards irrecoverable from the storage medium [8]. The time between deleting data and it becoming irrecoverable is called the deletion latency in [8].

This paper proposed a user-level secure data sanitization method and the structure of this paper is as follows. In Section 2, we review some related work. In Section 3, we describe some international mainstream data sanitization algorithm and standard. In Section 4, we describe our proposed method which includes a flow diagram and a step-by-step description of the method. In Section 5, we present some experiment results to show the performance advantages of the proposed method. Finally, in Section 6, we conclude this paper in which we also discuss our plan for future work.

ISSN: 1738-9984 IJSIA

2. Related Work

There has been some research effort on data sanitization. Paper [3] modified YAFFS to support secure deletion uses encryption to delete files and forces all keys of a specific file to be stored in the same block. In [4], a method to make the contents of the deleted blocks in the flash memory unreliable was proposed, it can be used for any data security algorithm in terms of error detection and correction. The method involves introducing appropriate number of errors in the data and control bits of the blocks of data making data recovery impossible. S. M. Diesburg, *et al.*, summarized and compared existing methods of providing confidential storage and deletion of data in personal computing environments [6]. Paper [8] shows that these systems provide no temporal guarantees on data deletion and that deleted data still persists for nearly 44 hours with average phone use and indefinitely if the phone is not used after the deletion. They proposed three mechanisms for secure deletion on log-structured file systems. R. Joel, *et al.*, presented three user-level solutions for secure deletion on log-structured file systems assuming a coercive attacker capable of compromising both the storage medium and any secret keys required to access it [9]. The principle behind their three solutions—purging, ballooning, and our hybrid solution—is that they reduce the file system's available free space to encourage more frequent garbage collection, thereby decreasing the deletion latency of deleted data. Ballooning and purging work in different ways and the hybrid solution combines them and relieves the disadvantages of each. Paper [11] analyses the attack module of the data of NAND Flash and explores the relationship between the safety of sanitization and the overwriting character and times, and then the paper has thorough research on the technology of storage data sanitization of NAND Flash based on the total overwriting and jump times. Reardon, J, *et al.*, further presented taxonomy of adversaries differing in their capabilities as well as systematization for the characteristics of secure deletion approaches [12]. Characteristics include environmental assumptions, such as how the interface's use affects the physical medium, as well as behavioural properties of the approach such as the deletion latency and physical wear. They performed experiments to test a selection of approaches on a variety of file systems and analyze the assumptions made in practice.

3. The Mainstream Data Sanitization Algorithm and Standard

There are two types of secure deletion methods, overwriting and encryption. In overwriting, predetermined or random sequences of data are written over the file which has to be deleted securely [3]. It is known that even after the data on magnetic media is overwritten, there is a chance that it can potentially be recovered by special techniques. Therefore, many overwriting based secure deletion methods overwrite data multiple times. The National Institute of Standards and Technology (NIST) recommend that magnetic media should be overwritten at least three times. Given this, achieving secure deletion by overwrite data multiple times is too expensive since it will wears out the storage medium. The mainstream standard of data sanitization is listed in Table 1. There is also huge data produced by the Android system and applications as well as user data. To find the sensitive data by scanning all of the NAND flash pages from the beginning to the end will cost much time.

In encryption-based secure deletion methods, data can be securely deleted by erasing a key which has been used to encrypt the data. In order to use these methods, every file should be encrypted with a different key. There are many file systems that ensure confidentiality of files by encrypting the file data with individual file or

directory keys. If these encryption-based secure deletion methods are used, users have to protect keys try their best. In addition, to the data already stored in the NAND flash, the plaintext can still be found in storage device even though you encrypted it after the writing operation.

Table 1. The Mainstream Data Sanitization Standard

Data sanitization standard	Description
DoD 5220.22-M-Sup 1(1995)	1.Erase all of the data; 2.Overwrite with single characters; 3.Overwrite with the complement of single characters; 4.Overwrite with random single characters; 5.Additional check procedures if necessary.
DoD 5220.22-M(1997）	1.Overwrite with single characters; 2.Erase the overwriting space.
NSA/CSS storage device Declassification manual	1.Overwrite with template predetermined without encryption; 2.Check the overwriting space with random reread.
Media Clearing, Purging, and Destruction	1.Erase all of the data; 2.Overwrite two times with Pseudo random values; 3.Overwrite with template predetermined once; 4.Additional check procedures if necessary.
IRIG 106-07,Ch.10	1.Erase all of the data; 2.Overwrite with 0x55； 3.Erase all of the data again; 4.Overwrite with 0xAA, erase the third time; 5.Overwrite with a random sequence.

4. The Proposed Method

Author names and affiliations are to be centered beneath the title and printed in Times New Roman 12-point, non-boldface type. Multiple authors may be shown in a two or three-column format, with their affiliations below their respective names. Affiliations are centered below each author name, italicized, not bold. Include e-mail addresses if possible. Follow the author information by two blank lines before main text. In YAFFS2, there are thus two types of chunks: data chunks and object header chunks [5]. Data chunks contain the actual file content while object header chunks contain file/directory metadata and descriptor information, such as file size, object name and creation time [10]. Each chunk has Tags in the spare area that holds additional information such as the chunk ID, serial number, number of bytes and object ID. Files stored on the NAND chips have unique identities, *i.e.*, the object IDs, since they are regarded as objects by the file system.Yaffs2 defined a data structure yaffs_ExtendedTags to store the above information.

```
typedef struct {
  unsigned validMarker0;
  unsigned chunkUsed;    /* Status of the chunk: used or unused */
  unsigned objectId;      /* If 0 then this is not part of an object (unused) */
  unsigned chunkId;      /* If 0 then this is a header, else a data chunk */
  unsigned name[YAFFS_MAX_NAME_lenth+1] ;   /*Object name*/
  unsigned byteCount;   /* Only valid for data chunks */
  unsigned chunkDeleted;   /* The chunk is marked deleted */
  unsigned sequenceNumber;   /* The sequence number of this block */
  unsigned extraParentObjectId;    /* The parent object */
  unsigned extraIsShrinkHeader;    /* Is it a shrink header? */
  yaffs_ObjectType extraObjectType;   /* What object type? */
```

```
unsigned extraFileLength;      /* Length if it is a file */
unsigned validMarker1;
} yaffs_ExtendedTags;
```

For better performance, we choose the reversely-scanning algorithm to find sensitive data pages and general steps are as follows. Let $Block_n$ denote the block with sequence number n and $Page_m$ denote the mth page in the block. We begin with the block with the largest sequence number and scan the whole block from the last chunk to the first one.

Reversely-scanning algorithm:

Step 1: Scan $Block_n$;
Step 2:Scan $Page_m$ in $Block_n$. If $Page_m$ is the first one in the block, jump to Step 4;
Step 3: m=m-1, go to Step 2;
Step 4: If n=0, then end; else n=n-1, return to Step 1.

As a user of the device, you only need to input the file name, confirm that you want to sanitization this file after the system scanning completely. Sanitization algorithm is as follows:

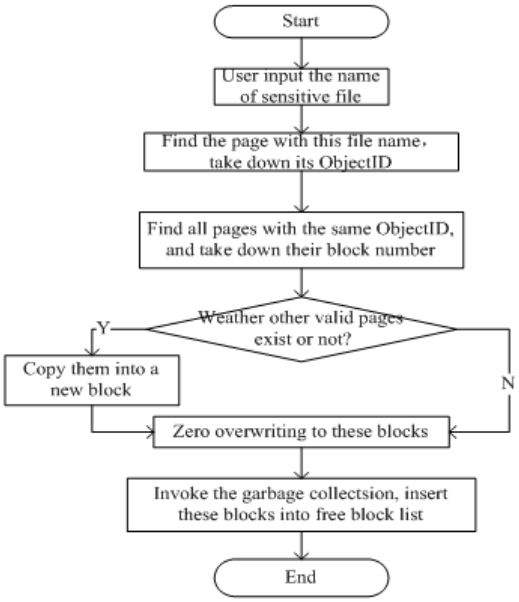

Figure 1. The Sanitization Algorithm Steps

Step 1: User input the file name which is going to be sanitization;

Step 2: System scan NAND flash using the reversely-scanning algorithm, read the YCHAR name in yaffs_tags from the spare area of each page and match it with the file name user input;

Step 3:To find all of the pages which has the same ObjectID as soon as the file name of one page matched successfully. Get the block number and ChunkID of these pages, scan end;

Step 5：Display message to the user, confirm weather to continue sanitization or not;

Step 6：If there are other valid data pages stored in these blocks, copy them into a new block;

Step 7： Do zero overwriting to these blocks;

Step 8：Erase these blocks, and insert them into free block list. In order to ensure that they will be allocated as a free block in the shortest time, we insert them in the front of free block list.

Since the basic unit of erase operation in Yaffs2 is block, it is necessary to copy valid

data to a new location before performing an erasure. The detailed operation of zero overwriting is that the program call writing function to converts all ones to zeros. In block erasure the contents of the block are erased by converting all zeros to ones in the memory. Then the program encourages garbage collection to recycling these blocks and inserts them into the front of the free block list. When there has new data waiting to be wrote, the erased blocks will be allocated and overwrote by new data. We will do a detailed analysis of this in the following section.

5. Evaluation

The proposed methods do only once overwriting operation then erased the blocks directly. After have been erased, the blocks are inserted into the free block list waiting to be allocated as free blocks. When new data be wrote, the blocks will be overwrote again. In a word, these methods not only reduce time but also can be applied with only a small additional wear to the NAND flash. The evaluation of time and secure performance in our proposed method is given below.

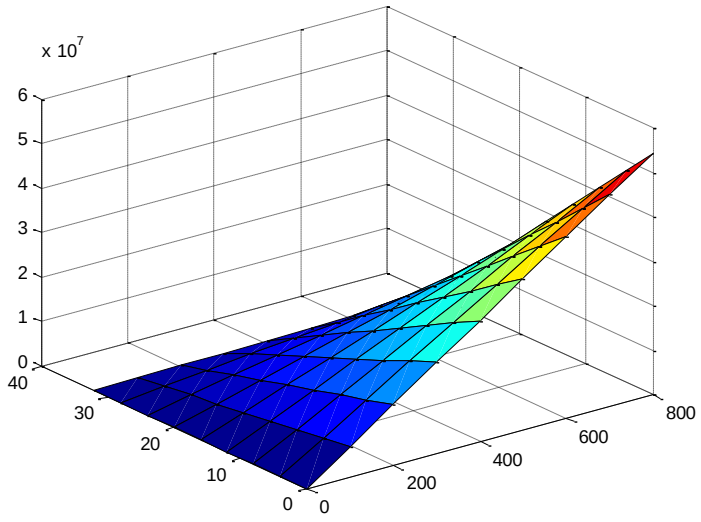

Figure 2. Time Performance of Proposed Method

Typically each small block NAND flash memory has a capacity of 32 pages of 528 bytes each with 16 bytes of additional information. Assuming that the sensitive data is distributed in each block evenly and each block consists of n pages to be deleted. Some constraints were made in the time performance evaluation.

Time reading/writing one page: Tr, Tw;

Time erasing one block: Te;

The number of blocks which contain sensitive data pages: N;

The total cost time is: T_{erase}=n*Tr+(32-n)*(Tr+Tw)*N+Te*N, in which n*Tr is the time spent in reading sensitive data pages; (32-n)*(Tr+Tw)*N is the time spent in copying other valid pages into a new block; Te*N is the time spent in erasing blocks. Theoretically Tr=30us,Tw=200us,Te=1800us, the three-dimensional view shown in Figure 2.

The process of erasing operation is from 0 to 1 which resulting in residual charge reduces. And from 1 to 0 will resulting in residual charge increases. Chang from 0 to 0 or from 1 to 1, the residual charge can be considered has no change. The residual charge changes can affect the threshold voltage. The numbers of write cycles can be calculated

from the threshold voltage changes, and then extrapolate the data before. The threshold voltage changes with the increase of written number as shown in Figure 3. We can see that when the write cycle to a certain extent, the threshold voltage is stabilized, the recovery rate is reduced as follow. Assuming that the flash memory size is nMB, the time period user write nMB to the device is T. Remember the destruction algorithm ends at the moment zero, three times overwrite operation has been carried out in our approach. The recovery rate is shown in Figure 4.

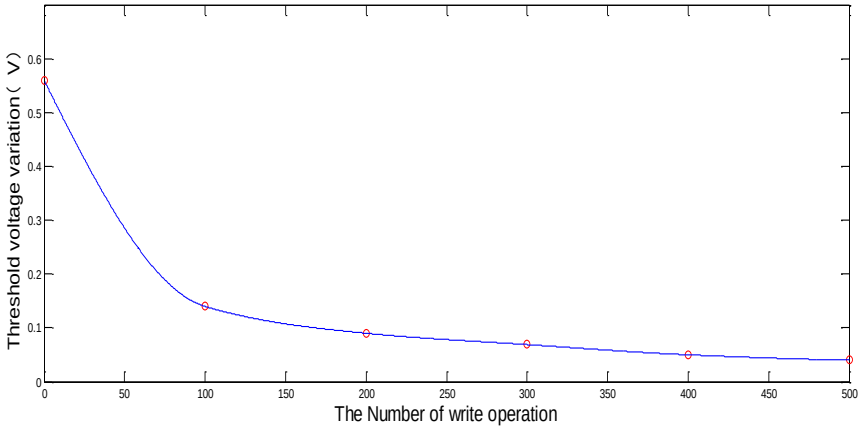

Figure 3. Threshold Voltage Variation with the Written Number

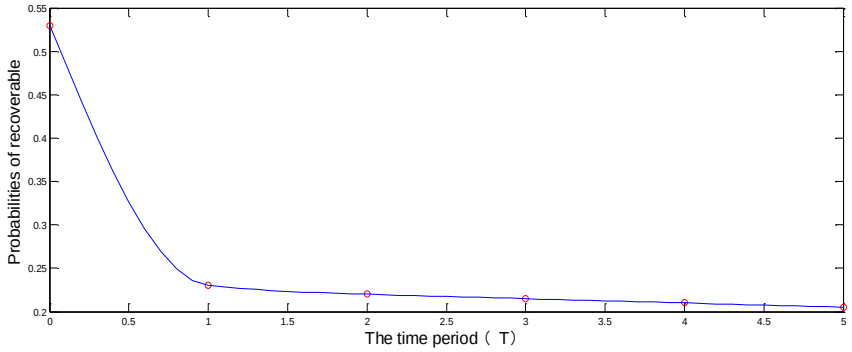

Figure 4. The Recovery Rate

6. Conclusion

Secure data sanitization enables users to protect the confidentiality of their data since the Android mobile devices contain lots of user data now. In this work, we present a user-level data sanitization method which overwrite sensitive data, erase the blocks and insert them into the front of free blocks list. This method use special scanning algorithm to reduce time and also take the lifetime of flash memory into account so it can be applied with only a small additional wear to the NAND flash.

ACKNOWLEDGEMENTS

The work in the paper has been supported by National Natural Science Foundation of China (61272500) and Beijing Natural Science Foundation (4142008).

References

[1] "Defense Security Service. National Industrial Security Program Operating Manual (NISPOM)", chapter 8: Automated Information System Security. U.S. Government Printing Office, **(1995)** January.
[2] T. Grance, M. Stevens and M. Myers, "Guide to Selecting Information Technology Security Products", chapter 5.9: Media Sanitizing. National Institute of Standards and Technology (NIST), **(2003)** October.
[3] J. Lee, J. Heo, Y. Cho, J. Hong and S. Y. Shin, "Secure Deletion for NAND Flash File System", ICCE Digest of Technical Papers. International Conference on, Consumer Electronics, **(2008)**, pp. 1-2.
[4] D. Amrit, "An algorithm for Secure formatting of memory", International Journal of Computer and Distributed System, vol. 1, no. 2, **(2013)**.
[5] C. Manning, "How YAFFS Works", **(2010)** Available at: http://www.yaffs.net/documents/how-yaffs-works.
[6] S. M. Diesburg and A. A. Wang, "A Survey of Confidential Data Storage and Deletion Methods", ACM Computing Surveys, vol. 43, no. 1, **(2010)**.
[7] C. Zimmermann, M. Spreitzenbarth, S. Schmitt and F. C. Freiling, "Forensic Analysis of YAFFS2", Lecture Notes in Informatics, vol. 195, **(2012)**, pp. 59-70.
[8] J. Reardon, C. Marforio and S. Capkun, "Secure Deletion on Log-structured File Systems", **(2011)**.
[9] J. Reardon, C. Marforio, S. Capkun and D. Basin, "User-level secure deletion on log-structured file systems", Proceedings of the 7th ACM Symposium Information, Computer and Communications Security, **(2012)** May, pp. 63-64.
[10] M. Xu, X. Yang, B. Wu, J. Yao, H. Zhang, J. Xu and N. Zheng, "A Metadata-based Method for Recovering Files and File Traces from YAFFS2", Digital Investigation, vol. 10, no. 1, **(2013)**, pp. 62–72.
[11] G. Zheng, "A Research on the Sanitization Technology for data in NAND Flash", PLA Information Engineering University, **(2013)**.
[12] J. Reardon, D. Basin and S. Capkun, "SoK: Secure Data Deletion, Security and Privacy (SP)", IEEE Symposium on, **(2013)**, pp. 301-315.

Authors

Na Huang, is currently a M.S. student in the School of Software Engineering at Beijing University of Technology in China. Her research focuses on digital forensic.

He Jingsha, received his B.S. degree from Xi'an Jiaotong University in Xi'an, China and his M.S. and Ph.D. degrees from the University of Maryland at College Park in USA. He is currently a professor in the School of Software Engineering at Beijing University of Technology in Beijing, China and an associate director in the Low Carbon Research Center at Beijing Development Area Co., Ltd. in Beijing, China. Professor He has published over 200 research papers in scholarly journals and international conferences and has received over 40 patents in the United States and in China. His main research interests include information security, network measurement, and wireless ad hoc, mesh and sensor network security.