Poster: Self-Defensible Storage Devices based on Flash memory against Ransomware

Joon-Young Paik
Computer Engineering Dept.
Ajou University
Suwon, South Korea
lucadi@ajou.ac.kr

Keuntae Shin
Samsung Electronics Co.
Suwon, South Korea
keuntae.shin@gmail.com

Eun-Sun Cho
Dept. of Computer Sci. and Engineering
Chungnam National University
Daejeon, South Korea
eschough@cnu.ac.kr

***Abstract*—In this work, a storage-level approach for ransomware attacks is proposed, which enables a storage device to perform detection and data recovery independently of the host-level solution. It detects the intrusion of ransomware based on storage-access activity and recovers original data from encrypted data without any decryption key with our ransomware-aware garbage collector.**

Keywords—ransomware, detection, recovery, Solid State Device (SSD), garbage collector

I. INTRODUCTION

In recent years, ransomware has emerged as one of the most troublesome threats to cyber security. Ransomware locks users' files by encrypting them and then demands payment for the decryption key. It is almost impossible to retrieve the encrypted files without the key held by the attackers.

Most solutions, such as detection tools and recovery systems (e.g. backup), have been proposed to protect against ransomware in the host system. However, studies on storage devices have not been carried out, even though storage-level approach is an attractive method of detecting ransomware, since storage devices can transparently monitor the storage-access activity of ransomware for file encryption. Moreover, flash-based Solid State Devices (SSDs) have a hidden space that cannot be recognized by an operating system [1]. The space can be exploited in a similar way to backup devices, thereby recovering the original data from the encrypted files. Moreover, commercial flash-based SSDs have much greater processing ability than traditional hard disk devices.

In this poster, we propose a storage-level solution, called the *Self-Defensible SSD*, to protect against ransomware by exploiting both the storage-access activity of ransomware and the property of flash memory. Our proposal detects the intrusion of ransomware and enables the original data of the encrypted data to be recovered without the decryption key.

II. BACKGROUND

A. Ransomware

Ransomware is a type of malware used to extort money for locked files from users. After communicating with Command-and-Control (C&C) servers, ransomware searches files with targeted formats (e.g. pdf, doc, jpg etc.), then encrypts and stores the files. As reported in [2], ransomware generates a common storage-access pattern; it *reads* the victim files, encrypts them, and then either *rewrites* the encrypted or dummy files on the original files or *deletes* the original files during a very short period of time [2], which we call *Reading-and-Rewriting/Deleting* pattern.

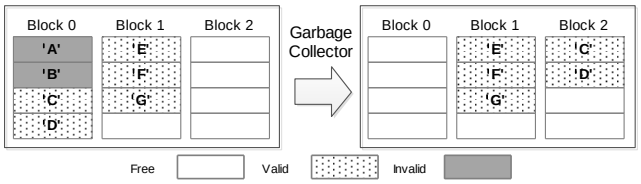


Fig. 1. Simple example of garbage collector.

B. Garbage Collector in Flash-based SSDs

Flash-based SSDs have extra space called *overprovision space*, in addition to accessible space called the *data space* of the operating system. Since no in-place update is performed in the flash memory [1], the overprovision space is populated by new data of update requests, and the corresponding overdue data in the data space is marked as *invalid* status. Eventually, the invalid data encroaches on the space, which results in the depletion of free space. Then, the *garbage collector* reclaims the invalid space to a free space for new write requests. At this time, the overdue data in the invalid space disappears forever. In Fig. 1, the garbage collector selects Block 0 as a victim for reclamation, migrates the valid data of 'C' and 'D' to Block 2, and then recycles it to a free block, in which the overdue data of 'A' and 'B' are eliminated forever. The garbage collector is not exposed to the host system. Thus, it is allowed to be managed only at the storage-level.

III. SELF-DEFENSIBLE SSDS AGAINST RANSOMWARE

The self-defensible SSD aims at supporting the detection and recovery against ransomware independently of the host-level solution. For this, two challenges emerge; (1) *how to detect ransomware with no semantic information* and (2) *how to recover the original data without decryption key and additional backup devices.*

Architecture: Typical flash-based SSDs have three essential software components: Address allocator, Address Translator and Garbage Collector.

To allow SSDs to defend against ransomware, we insert a ransomware detector and modify the garbage collector to create a ransomware-aware garbage collector, as shown in Fig. 2. The detector detects ransomware attacks based on the storage-access-activity, while the proposed garbage collector attempts to retain the original data corresponding to the encrypted data as long as possible for data recovery.

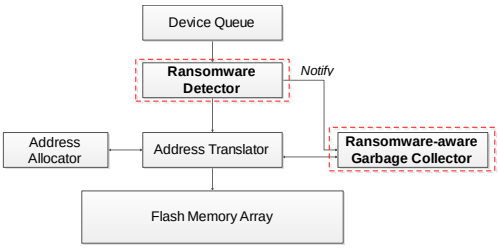


Fig. 2. System overview of self-defensible SSD.

Ransomware detection: Our ransomware detector is designed to be placed in SSDs. As part of SSDs, the detector can transparently monitor the storage-access activity but must use the limited information of type, address, and size of the I/O requests. To detect suspicious access patterns of the ransomware under the limited semantic information, we exploit *repetitive reading-and-rewriting/deleting patterns in a short period of time* during the encryption performed by ransomware.

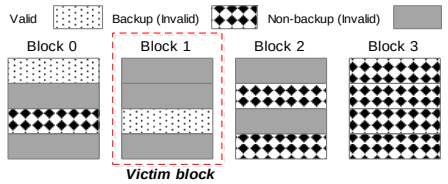
For this, we introduce a new data structure called the *Recently-Requested-Reads (RRR)* list in SSDs. The RRR list stores information about read requests for a short period of time, because the reading-and-rewriting/deleting pattern occurs very frequently by ransomware for a brief period of time.

When a read request is received at SSDs, its abstracted information is inserted into the RRR list. When the RRR list is full, its least recently used element is deleted from the list. On the other hand, when a write request or a trim request[1] is received, it is compared to the elements of the RRR list in order to check whether it rewrites or deletes the data that have been read within a certain period of time. When the incoming write or trim request is matched with the information on the element in RRR list, the matched element is marked as *backup* status because it signifies a reading-and-rewriting/deleting pattern. Then, the frequency of the reading-and-rewriting/deleting pattern in a short period of time is quantified by the ratio of the backup-marked elements to the total elements in the RRR list. If the ratio is larger than a certain threshold, it is detected as a ransomware attack. The detector notifies the ransomware's intrusion to the ransomware-aware garbage collector for data recovery.

Ransomware-aware garbage collection: For data recovery in flash-based SSDs, it is very important to manage the garbage collector because it determines the longevity of overdue data in the invalid pages, which indicate the original version data of the files encrypted by ransomware. The original version data provide an opportunity to *recover the original data without the decryption key and additional backup devices* in spite of being in the invalid pages.

The goal of the ransomware-aware garbage collector is to retain the backup-marked data as long as possible for data recovery. In general, the greedy victim selection policy [3] of the garbage collector is used for high performance. In that policy, the block with the lowest value of $\frac{\#\ of\ valid\ pages}{total\ \#\ of\ pages}$ is selected as a victim block. However, it prohibits the recovery of the *backup* data because it tends to remove the data promptly. On the other hand, in the ransomware-aware garbage collector, we separate invalid data into *Backup* data and *Non-backup* data. To prohibit the garbage collector from removing the *Backup* data, our proposal considers *Backup* pages as well as valid pages for victim selection. Thus, it selects a victim block based on the following equation.

[1] Most operating systems can generate trim commands [4] when files are deleted, in which the space has an invalid status.



	Block 0	Block 1	Block 2	Block 3
Greedy Cost	0.25	0.25	0	0
RaC	0.5	**0.25**	0.5	1

Fig. 3. Example of victim block selection based on RaC in ransomware-aware garbage collection.

$$Ransomware\text{-}aware\ Cost\ (RaC) = \frac{\#\ of\ valid\ and\ Backup\ pages}{total\ \#\ of\ pages}$$

The block with the lowest value of *Ransomware-aware Cost (RaC)* is chosen. In addition to the valid pages, the *Backup* pages migrate to the free space. Fig. 3 shows the ransomware-aware garbage collector selecting Block 1 of the lowest value of RaC as a victim, resulting in no *Backup* pages being removed. On the other hand, the greedy garbage collector cannot recover the four *Backup* pages when Block 3 is chosen as a victim due to its lowest greedy cost.

Our garbage collector helps to improve the recovery of data from the encrypted files without the decryption key by retaining the original version data as long as possible.

IV. Discussion and Conclusion

In our proposal, the ability of data recovery depends on the size of the overprovision space in the SSDs. In commercial SSDs, the overprovision space is from 7% to 37% of the data space [1]. In spite of the limitation on the space, it would be sufficient because the sizes of the files targeted by ransomware are relatively small.

The proposed self-defensible SSD detects ransomware attacks and recovers the original data from the encrypted data without a decryption key with a ransomware detector and ransomware-aware garbage collector. Our novel storage-level approach enables storage devices to defend against ransomware attacks independently of the host-level solution.

References

[1] K. Smith, Understanding SSD over Provisioning, Flash Memory Summit 2012.

[2] A. Kharra, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: a look under the hood of ransomware attacks," *in Proceedings of Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2015.

[3] M. Wu, W. Zwaenepoel, "eNVy: a non-volatile, main memory storage system," *in Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 1994.

[4] Information Technology - ATA/ATAPI Command Set-2 (ACS-2). INCITS Working Draft T13/2015-D Rev. 7, June 2011. http://www.t13.org/documents/UploadedDocuments/docs2011/d2015r7-ATAATAPI_Command_Set_-_2_ACS-2.pdf.