Poster: Incorporating Malware Detection into Flash Translation Layer

Wen Xie, Niusen Chen, Bo Chen
Department of Computer Science, *Michigan Technological University*
{wenxie, niusenc, bchen}@mtu.edu

***Abstract*—OS-level malware may compromise OS and obtain root privilege. Detecting this type of strong malware is challenging, since it can easily hide its intrusion behaviors or even subvert the malware detection software (or malware detector). Having observed that flash storage devices have been used broadly by computing devices today, we propose to move the malware detector to the flash translation layer (FTL), located inside a flash storage device. Due to physical isolation provided by the FTL, the OS-level malware can neither subvert our malware detector, nor hide its access behaviors from our malware detector.**

***Index Terms*—OS-level malware, detection, flash translation layer, dynamic analysis**

I. Introduction

Malware may compromise OS and obtain root privilege. We call this type of malware the OS-level malware. Detecting such malware has been a challenging problem, because: by obtaining root privilege, the malware can easily hide its intrusion behaviors, or even directly subvert malware detection software running in user/kernel space. This work aims to tackle this problem by moving the malware detection software, namely, malware detector, out of the regular user/kernel space.

Modern computing devices are increasingly using flash memory as external storage. For example, high-end desktops/laptops usually use solid state drives (SSDs), instead of hard disk drives (HDDs); mobile devices like smartphones, tablets, IoT devices extensively use flash memory in forms of UFS cards, eMMC cards, or SD/miniSD/microSD cards. Unlike mechanical drives, flash memory exhibits completely different characteristics, e.g., erase-then-write design, being vulnerable of wear. To handle this unique nature transparently, an independent firmware layer, flash translation layer (FTL), is usually integrated into a flash storage device, such that it can provide a block-access interface externally.

Having observed existence of extra firmware layer in a flash storage device, we propose to integrate the malware detector into the FTL. This new design is advantageous in terms of security, as the OS-level malware will not be able to subvert the malware detector located in the FTL which is transparent to the OS. However, it could be also risky, as a malware detector staying in a place under the OS may not be able to have access to malware behaviors. Fortunately, our preliminary experimental evaluation on various types of malware shows that malware running in the upper layer (e.g., application and OS) does exhibit some unique behaviors in the FTL.

II. Background on Flash Memory and FTL

Flash-based block devices (e.g., SSD drives, UFS cards, eMMC cards, SD cards) are used extensively in desktops, laptops, and smart devices. As a non-volatile storage medium which can be electrically erased and reprogrammed, flash memory contains NOR and NAND flash. NAND flash is usually cheap and has large capacity, and hence has been used broadly as mass storage. NAND flash stores information in an array of memory cells, which are grouped into blocks. Each block is further divided into a few (e.g., 32, 64, or 128) pages. NAND flash supports three basic operations: read, program (write), and erase. A read/program operation is usually performed on the basis of pages, while an erase operation is performed on the basis of blocks.

Flash memory has completely different nature compared to HDDs. To be compatible with traditional block file systems (e.g., EXT4, FAT32), a flash-based storage device is usually used as a regular block device by exposing a block access interface. This is achieved by introducing FTL (flash translation layer), which stays between the file system and NAND flash, transparently handling special characteristics of NAND flash.

III. System and Adversarial Model

System model. We mainly consider computing devices equipped with flash-based block devices. These include desktops/laptops equipped with SSDs, and mobile devices equipped with flash memory cards (e.g., eMMC, SD).

Adversarial model. We consider malware which has the ability of compromising OS of the host computer. Therefore, any conventional malware detectors which run in the application/OS layer may be subverted and cannot defend against this type of "high-privilege" malware. Note that we mainly consider malware which causes I/Os on the external storage, e.g., computer viruses, ransomware, etc; additionally, our malware detector aims to detect the malware in the FTL and once detects malware, it will inform users for further actions (e.g., malware blocking and removal, OS restoration, and system recovery [4]), which are not focuses of this work.

IV. Preliminary Design and Experiments

To combat the OS-level malware, we integrate the malware detector into the FTL, the internal software layer of a flash-based storage device. In this way, the OS-level malware will not be able to subvert the malware detector due to physical isolation provided by the FTL. In addition, the OS-level malware is not able to hide its intrusion behaviors utilizing its root privilege (e.g., modifying the system logs to eliminate all its traces). This is because, intuitively, special behaviors of malware in upper layers will likely cause special access behaviors on the underlying flash memory, which can be observed by the malware detector located in the FTL.

Our FTL-based malware detector consists of three components (Figure 1): I/O monitor, malicious pattern set, and detection engine. The I/O monitor will continuously watch the I/Os issued by the upper layers and extract access patterns. The malicious pattern set consists of patterns collected through performing dynamic analysis on known malware data sets. Detection engine compares access patterns sent by the I/O monitor with those stored in the malicious pattern set, and determines existence of malware.

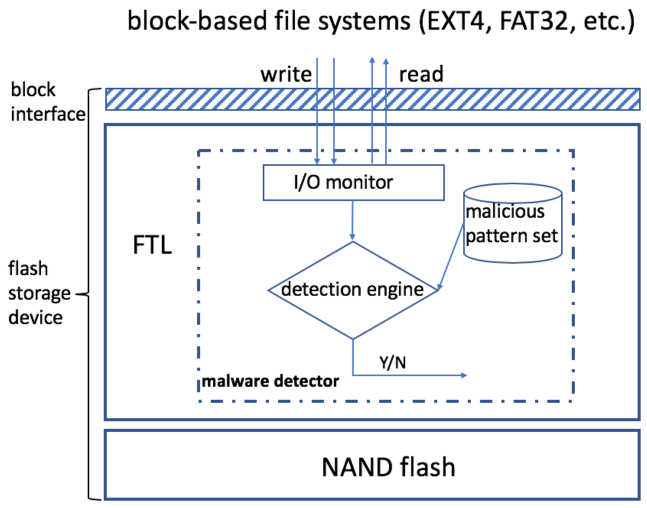

Fig. 1. A preliminary design of an FTL-based malware detector

Preliminary implementation and experiments. Our testbed consists of open-source flash firmware, OpenNFM [3], an electronic board LPC-H3131 [5] with 512MB NAND flash, and a host computer with Intel Core i7 8700K and 32GB RAM. The OpenNFM was ported to LPC-H3131, which can then be used as a regular flash-based block device via USB. OpenNFM contains 3 layers, FTL, UBI, and MTD. The FTL layer provides a few access interfaces including FTL_Read and FTL_Write, and we can capture each access caused by the upper layers by intercepting FTL_Read and FTL_Write. To extract access patterns of malware, we did the following: 1) We collected 31 ransomware families and 3 computer virus families [1], each contains a few malware samples. 2) For each malware sample, we performed dynamic analysis by running it on the host computer, and collecting all its corresponding I/Os in the FTL into a trace file (by modifying OpenNFM). We collected 62 I/O trace files in total. 3) We analyzed all the 62 trace files, extracting patterns. The trace files were analyzed as follows:

- View each trace file as a three-dimension sequence and use dynamic time warping to calculate distance between each pair of trace files, obtaining a distance matrix.
- Use the obtained distance matrix as input to hierarchical clustering algorithm to cluster all the trace files, obtaining clusters.
- Manually distill I/O patterns in each cluster.

The extracted malicious I/O patterns are provided in Table I, in which N_f denotes #families, and N_s denotes #samples, under each malicious I/O pattern (note that the computer virus families are in the rows corresponding to ID 8 and 9).

ID	N_f	N_s	I/O patterns in FTL
1	8	15	Reading is split into several sizes. Write the whole size into original logical page address
2	9	14	Writing size is smaller than reading size, starting page address is the same
3	4	7	Writing size is equal to reading size, starting page address is different
4	1	1	Reading and writing size is mostly 32 or 64, starting page address is not the same.
5	1	2	R/W size is mostly 1 or 2 at the beginning, and finally is almost 32.
6	1	4	Reading and writing is a sequence with size 16,2,2,2. starting page address is the same respectively.
7	7	13	Reading is immediately followed by writing, their size is equal, starting page address is the same
8	1	2	Writing corrupted file to original place(almost same size), then write typical size (virus payload) to new place
9	2	4	First write typical size (virus payload) to original place or new place, then write corrupted file to new place

TABLE I
MALICIOUS I/O PATTERNS EXTRACTED IN THE FTL

V. RELATED WORK

Most existing malware detectors rely on the assumption that the malware cannot obtain OS privilege, which is unfortunately not true for the OS-level malware. Especially, to enable data recovery after ransomware attacks, a few existing works [2], [6], [7] study ransomware behaviors in the FTL. However, their designs are purely for ransomware, a special type of malware. On the contrary, our malware detector aims to detect any types of malware which generate I/Os on NAND flash.

VI. ACKNOWLEDGMENT

This work was supported by National Science Foundation under grant number 1938130-CNS and 1928349-CNS.

REFERENCES

[1] Malware samples for ransomware and computer virus. Retrieved March 30, 2020, from https://snp.cs.mtu.edu/research/drm2/malware-samples-03302020.html.
[2] S. Baek, Y. Jung, A. Mohaisen, S. Lee, and D. Nyang. Ssd-insider: internal defense of solid-state drive against ransomware with perfect data recovery. In *ICDCS*. IEEE, 2018.
[3] Google Code. Opennfm. Retrieved May 17, 2019, from https://code.google.com/p/opennfm/, 2011.
[4] L. Guan, S. Jia, B. Chen, F. Zhang, B. Luo, J. Lin, P. Liu, X. Xing, and L. Xia. Supporting transparent snapshot for bare-metal malware analysis on mobile devices. In *ACSAC*. ACM, 2017.
[5] Mantech. Lpc-h3131. Retrieved May 17, 2019, from https://www.olimex.com/Products/ARM/NXP/LPC-H3131/, 2017.
[6] D. Min, D. Park, J. Ahn, R. Walker, J. Lee, S. Park, and Y. Kim. Amoeba: an autonomous backup and recovery ssd for ransomware attack defense. *IEEE Computer Architecture Letters*, 17(2):245–248, 2018.
[7] P. Wang, S. Jia, B. Chen, L. Xia, and P. Liu. Mimosaftl: Adding secure and practical ransomware defense strategy to flash translation layer. In *CODASPY*. ACM, 2019.