Incorporating Malware Detection into the Flash Translation Layer

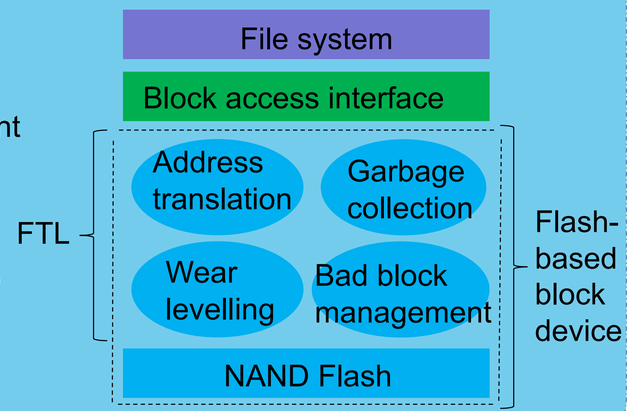
Wen Xie, Niusen Chen, Bo Chen
Department of Computer Science, Michigan Technological University

Motivation

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).

Isolated Environment in Flash Memory

- Modern computing devices are increasingly using flash memory as external storage
- Flash memory exhibits completely different characteristics compared to traditional mechanical disks
- To handle unique nature of flash memory, an independent firmware layer, flash translation layer (*FTL*), is usually integrated into a flash storage device
- FTL is transparent to the OS



A flash-based storage system

Malware Detection Comparison

	OS-based	FTL-based
Pros	Being able to have access to rich behaviors of malware	Located in an isolated environment where the OS-level malware cannot "touch"
Cons	Could be compromised by the OS-level malware	Can only detect malware which has perform I/Os on the external storage

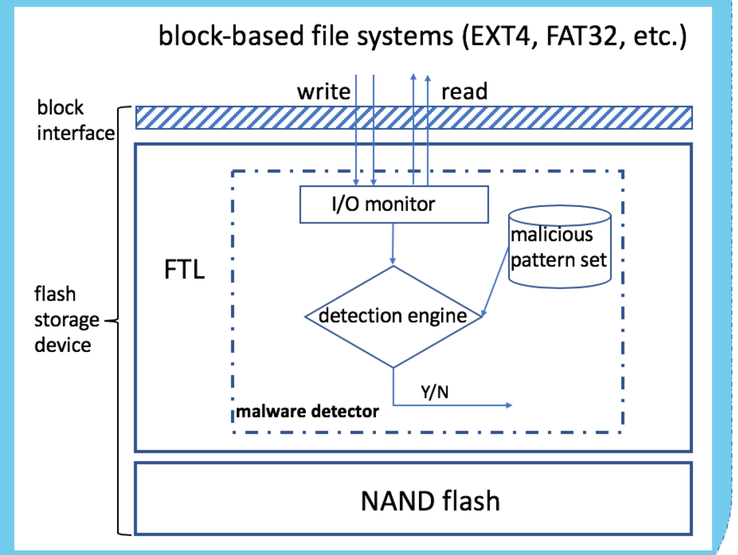
Adversarial Model

- The malware can compromise OS of the host computer
- The malware causes I/Os on the external storage, e.g., computer viruses, ransomware, etc
- 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 removal)

Preliminary Design

An FTL-based malware detector:

- **I/O Monitor:** Observes the I/Os issued by the upper layers and extracts access patterns
- **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 sets, and determines existence of malware



The design of an FTL-based malware detector

Experimental Results

- Ported an open-source flash controller, OpenNFM, to an electronic board LPC-H3131, which can then be used as a USB device
- Collected 62 malware samples from 31 ransomware families and 3 computer virus families
- Ran each sample on a host computer and collected corresponding I/Os in the FTL of the USB device, generating 62 trace files
- Analyzed all the 62 trace files, extracting patterns

C	N_f	N_s	Pattern
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 different
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 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

Malicious I/O patterns extracted. C: Cluster; N_f: Number of families; N_s: Number of samples

Acknowledgment

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