Poster: Your Access Control List Is Recoverable Even if Your OS Is Compromised

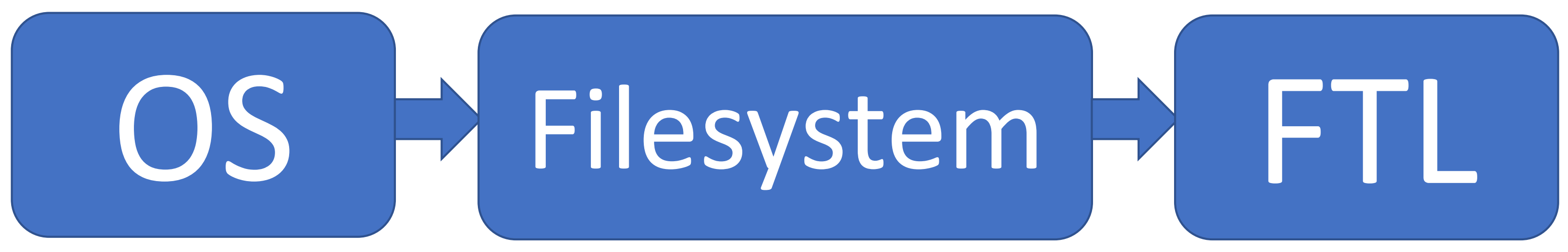
Caleb Rother and Bo Chen
Computer Science Dept., Michigan Technological University

Introduction

In the history of access control, nearly every system designed has relied on the operating system for enforcement of its protocols. If the operating system (and specifically root access) is compromised, there are few if any solutions that can get users back into their system efficiently. In this work, we have proposed a method by which file permissions (specifically EXT's Access Control Lists, or **ACL**) can be efficiently rolled back after a catastrophic failure of permission enforcement. Our key idea is to leverage the out-of-place-update feature of flash memory in order to collaborate with the flash translation layer to efficiently return those permissions to a state pre-dating the failure.

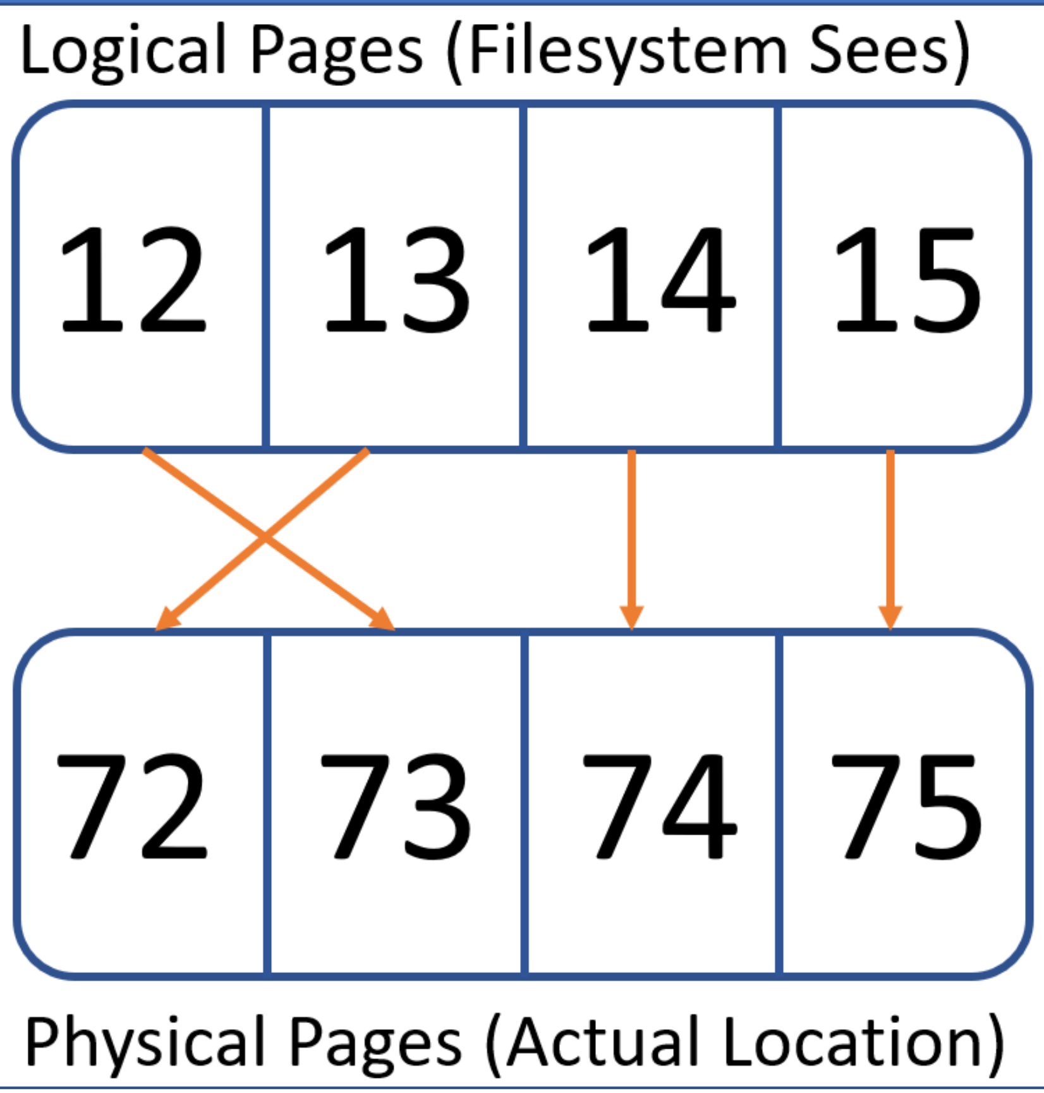
Background: Flash Translation Layer

NAND Flash has become nearly ubiquitous in the modern computing world. To remain compatible with traditional block file systems, a flash storage device (e.g., an SSD) introduces a new flash translation layer (FTL) between the file system and the NAND flash, transparently managing the special characteristics of NAND flash. This layer is helpful as it is baked into the flash storage media, making it difficult to be tampered with.



Background: Out-of-place updates

When the FTL writes, it does so to a new address and updates a Page Mapping Table (PMT) tracking the actual location of each "page" of the drive. Old copies are left on the drive for a short period of time.


Logical Pages (Filesystem Sees)
12
13
14
15
72
73
74
75
Physical Pages (Actual Location)

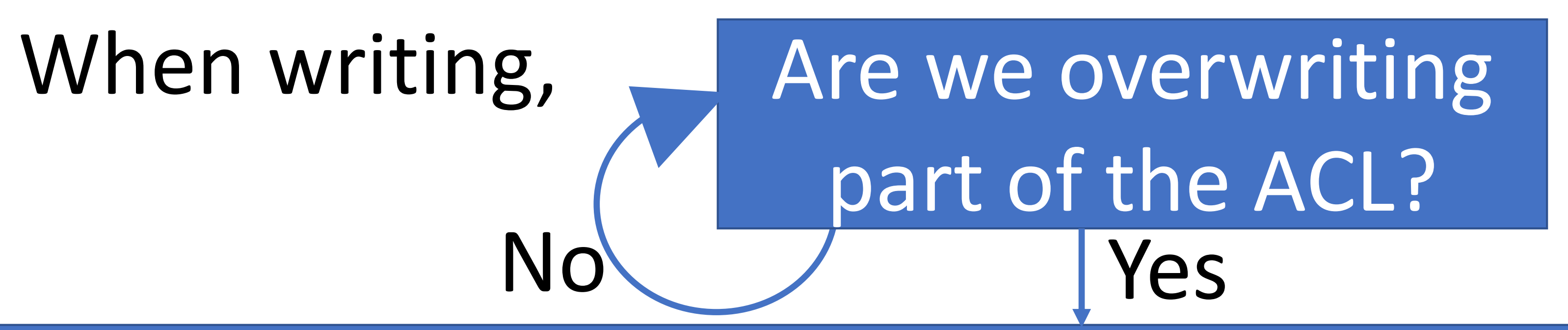
Threat Model

In our scenario, the adversary is able to compromise a computer's OS and obtain root access. The adversary tries to disrupt access control by modifying files' access permissions. Our system's job is to recover the permissions to a state before they were modified.

Design, Phase 1: Setup

The Access Control List is divided into sections (called "inode groups") and stored on the drive. The FTL needs to store three pieces of data: 1) the number of inode groups stored on the filesystem, 2) the logical page number for the first inode of each inode group (in an array), and 3) the size of each inode group, measured in pages.

Design, Phase 2: Monitoring

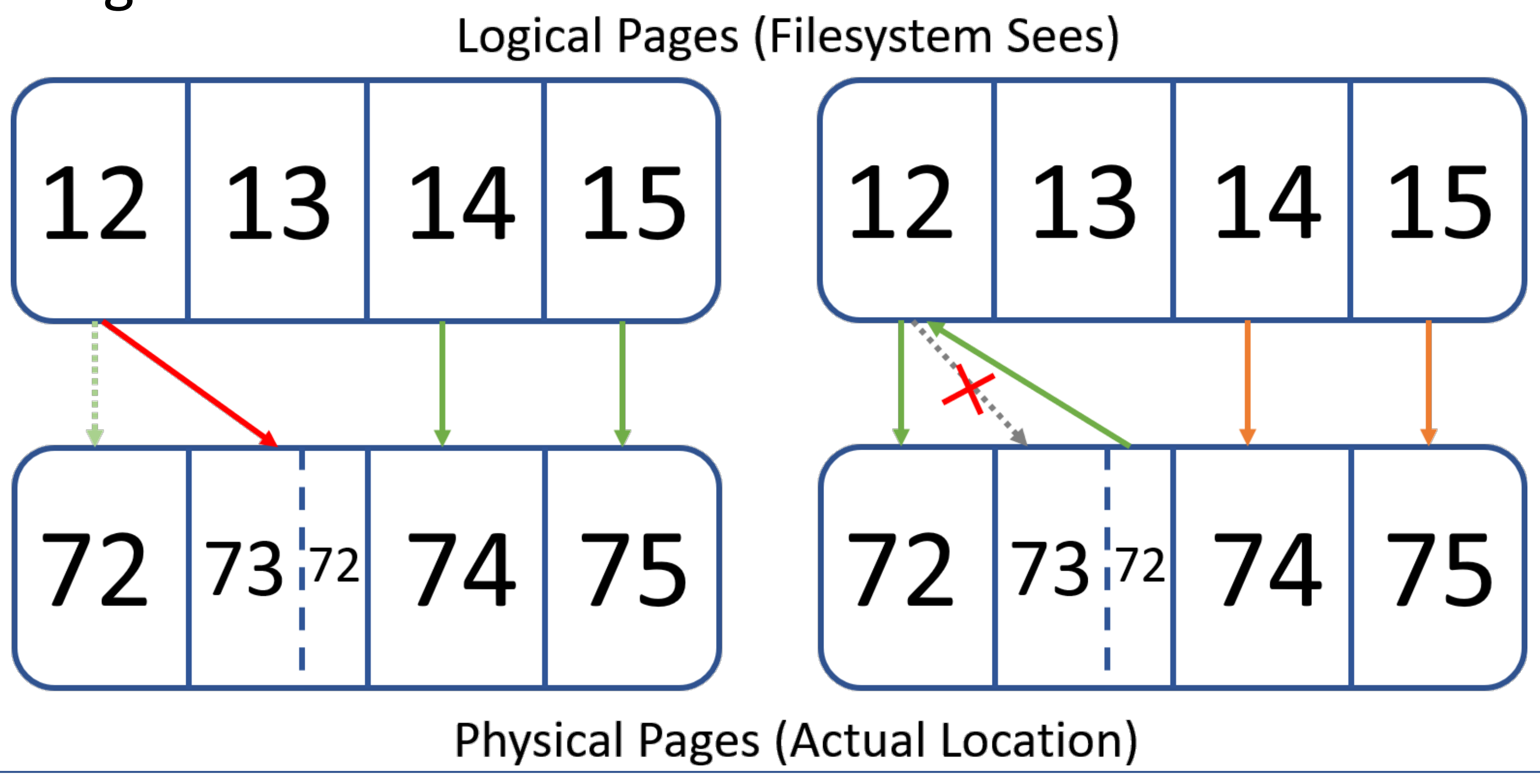


Store (in the FTL):

The current physical location of the page, in a small out-of-bounds section of the new page.	The logical location of the page, so that we know which changes are recent.

Design, Phase 3: Recovery

When we want to undo recent ACL changes, we can read the old location from the place we stored it in monitoring, then update the PMT to use it, undoing the change without having to write to the drive.


Logical Pages (Filesystem Sees)
12
13
14
15
12
13
14
15
72
73
72
74
75
72
73
72
74
75
Physical Pages (Actual Location)

Results

We were successfully able to implement our method on a linux system by modifying the free open-source FTL firmware OpenNFM. Our experiment showed that a rollback of the ACL entries is feasible. We also measured the time needed for rolling back an AVL entry to be around 11 milliseconds on average.

Acknowledgement

This work was supported by the US National Science Foundation under grant number 2225424-CNS, 1928349-CNS, and 2043022-DGE.