Project Almanac: A Time-Traveling Solid-State Drive[*]

Xiaohao Wang, Chance C. Coats, Jian Huang
Systems and Platform Research Group
University of Illinois at Urbana-Champaign

Abstract

Preserving the history of storage states is critical to ensuring system reliability and security. It facilitates system functions such as debugging, data recovery, and forensics. Existing software-based approaches like data journaling, logging, and backups not only introduce performance and storage cost, but also are vulnerable to malware attacks, as adversaries can obtain kernel privileges to terminate or destroy them.

In this paper, we present Project Almanac, which includes (1) a time-travel solid-state drive (SSD) named TimeSSD that retains a history of storage states in hardware for a window of time, and (2) a toolkit named TimeKits that provides storage-state query and rollback functions. TimeSSD tracks the history of storage states in the hardware device, without relying on explicit backups, by exploiting the property that the flash retains old copies of data when they are updated or deleted. We implement TimeSSD with a programmable SSD and develop TimeKits for several typical system applications. Experiments, with a variety of real-world case studies, demonstrate that TimeSSD can retain all the storage states for eight weeks, with negligible performance overhead, while providing the device-level time-travel property.

1 Background and Motivation

Retaining past storage states enables users to recover detailed information about updates in storage systems. This is useful for many purposes, such as debugging, data recovery and rollback, forensics, and error tracking.

Prior approaches rely on a variety of software-based techniques, such as journaling, logging, checkpointing, and backups. These techniques have become essential components of a vast majority of storage systems. However, they suffer from serious issues that significantly affect system reliability and security. To be specific, software-based solutions are vulnerable to malware attacks that can terminate the data backup processes or destroy the data backup itself. For instance, encryption ransomware can obtain kernel privileges and destroy data backups in order to prevent victims from recovering their encrypted data [1]. Furthermore, these software-based techniques inevitably impose storage overhead and additional I/O traffic.

A more lightweight and secure approach is desired, that retains the storage states and their lineage, without relying on software-based techniques, and without incurring overhead.

2 Threat Model

As discussed, malicious users could try to elevate their privilege to run as administrators and disable or destroy the software-based data backup solutions. We do not assume the OS is trusted, but instead, we trust the flash-based SSD firmware. We believe this is a realistic threat model for two reasons. First, the SSD firmware is located within the storage controller, underneath the generic block interface in computer systems. It is hardware-isolated from higher-level malicious processes. Second, SSD firmware has a much smaller TCB compared to the OS kernel, making it typically less vulnerable to malware attacks. Once the firmware is flashed into the SSD controller, commodity SSDs will not allow firmware modifications, which guarantees the integrity of TimeSSD. In this work, we only consider the situation where data on persistent storage is overwritten or deleted.

3 Design and Implementation

In this paper, we present a time-travel SSD (TimeSSD) that transparently retains storage states in flash-based storage devices, and maintains the lineage of states. TimeSSD enables developers to retrieve the storage state from a previous time, creating a time-travel property. We also present a toolkit called TimeKits which supports storage-state query and rollback, for developers to exploit the firmware-isolated time-travel property.

Flash-based SSD is an ideal candidate for TimeSSD for four reasons. First, since flash pages cannot be written without being erased, modern SSDs perform out-of-place updates to reduce the overhead generated by expensive erase operations. The out-of-place update routine inherently supports logging functionality since old versions of data are not immediately erased. Second, the SSD has to run garbage collection (GC) to reclaim obsolete data for free space. With a slight modification to the GC algorithm, we can reclaim older versions of storage state in time order. Therefore, we can maintain the lineage of storage states in an SSD by keeping track of older versions of data. Third, the SSD firmware is isolated from the operating system (OS) and upper-layer software. The isolated firmware has a much smaller Trusted Computing Base (TCB) than upper-level systems software, which is less vulnerable to malware attacks. It provides a last line of defense for protecting user data, even if the host OS is compromised. Lastly, SSDs have become prevalent in a vast majority of storage systems driven by their significantly increased performance and decreased cost.

TimeSSD is a lightweight firmware-level solution to fulfill the time-travel property in the hardware device. It balances the tradeoff between the retention time of past storage states and the storage performance by estimating the GC overhead. TimeSSD can dynamically adjust retention duration according to application workload patterns, while still enforcing a

[*]This work has been publihsed at EuroSys'19 [2].

guarantee of a lower bound on the retention duration (three days by default). To further retain past storage state for a longer time, TimeSSD compresses the storage state during idle I/O cycles. Our study with a variety of storage workloads, collected from both university and enterprise computers, demonstrates that TimeSSD can retain storage state for up to eight weeks, with minimal performance overhead.

To ensure performance, TimeSSD dynamically adjusts the retention duration, using GC overhead as a proxy for storage performance degradation. If the estimated GC overhead per write exceeds a given threshold, we reclaim some of the oldest invalid data. This reduces retention duration but improves SSD performance. A larger threshold results in lower SSD performance, but a longer retention duration. To fit with various workload patterns, TimeSSD dynamically adjusts the retention duration, in a workload-adaptive manner. For example, when workload writes become more intensive, the retention duration decreases accordingly.

However, TimeSSD will guarantee a lower bound for the retention duration in order to avoid malicious attacks. We chose a default minimum duration of three days, but this is configurable by SSD vendors. If the free storage space runs out and the retention duration has not reached three days, TimeSSD stops serving I/O requests, resulting in the failure of file system operations. The users or the administrator will quickly notice this abnormal behavior. Even in this extreme case, TimeSSD preserves the past storage state of the last few days, for further analysis.

To facilitate the retrieval of storage states, TimeSSD leverages the available hardware resources in commodity SSDs, such as the out-of-band (OOB) metadata of each flash page to store the reverse mapping from physical page in flash chip to logical page in the file system. It uses back-pointers to connect the physical flash pages that map to the same logical page, which enables TimeSSD to maintain the lineage of storage states in hardware. All OOB metadata operations are performed in SSD firmware, they are also firmware-isolated from upper-level software. To accelerate retrieving storage states, TimeSSD utilizes the internal parallelism of SSDs to parallelize queries among multiple flash chips.

Based on this time-travel property, we develop TimeKits, a tool that can answer storage state queries in both backward (e.g., what was the storage state a few hours ago?) and forward (e.g., how has a file been changed since some prior state?) manner. These basic functionalities enable features, such as protecting user data against encryption ransomware, recovering user files, retrieving update logs, and providing an evidence chain for forensics.

Project Almanac utilizes this firmware-level time-travel property to achieve the same goals as conventional software-based solutions, in a transparent and secure manner, with low performance overhead. Overall, we make the following contributions:

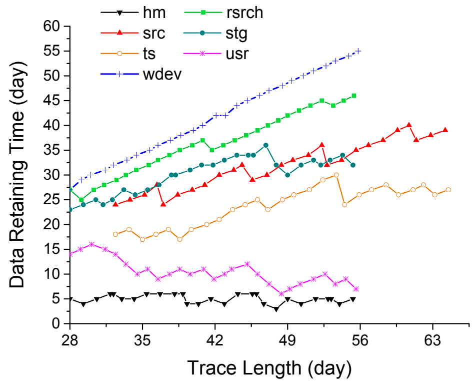

Figure 1. The data retention duration of TimeSSD. For applications running in company servers, the invalid data can be retained for up to 56 days.

- We present a time-travel SSD, named TimeSSD, which retains past storage states and their lineage in hardware, in time order, without relying on software techniques.
- We present a toolkit, named TimeKits, to exploit the firmware-isolated time-travel property of TimeSSD. It supports rich storage-state queries and data rollback.
- We quantify the trade-off between retention duration of storage states versus storage performance. We propose an adaptive mechanism to reduce performance overhead for TimeSSD, while retaining past storage states to a bounded window of time.

We implement TimeSSD in a programmable SSD, and develop TimeKits for a few popular system functions, such as data recovery and log retrieval of file updates.

4 Evaluation

We run TimeSSD against a set of storage benchmarks and traces collected from different computing platforms. Our experimental results demonstrate that TimeSSD retains the history of storage states for up to eight weeks (see Figure 1). TimeSSD fulfills these functions with up to 12% performance overhead. We also apply TimeSSD to several real-world use cases, and use TimeKits to retrieve past storage states or conduct data rollback , The performance results also show that TimeKits can accomplish the storage-state queries and data rollback instantly. The detailed experimental results can be found in [2].

References

[1] Jian Huang, Jun Xu, Xinyu Xing, Peng Liu, and Moinuddin K. Qureshi. 2017. FlashGuard: Leveraging Intrinsic Flash Properties to Defend Against Encryption Ransomware. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. Dallas, TX.

[2] Xiaohao Wang, Yifan Yuan, You Zhou, Chance C. Coats, and Jian Huang. 2019. Project Almanac: A Time-Traveling Solid-State Drive. In *Proceedings of the 14th European Conference on Computer Systems (EuroSys'19)*. Dresden, Germany.