Socarrat: Building Cost-Effective Secure WORM Devices Following the Reverse File System Approach

Gorka Guardiola Múzquiz
Enrique Soriano-Salvador
Universidad Rey Juan Carlos, Spain

Introduction & Motivation. WORM (Write Once Read Many) devices permit data to be written only once. Subsequently, the system can read the data an unlimited number of times. These devices are used for logging and other purposes (like backups) and are essential for a wide range of applications. A recurring theme in data regulations is the necessity for regulatory-compliant storage to offer WORM assurances that enable guaranteed retention of the data, secure deletion, and compliant migration [9]. In addition, the data should be tamper-evident: If the data has been manipulated, the auditor must to be able to detect it. Although it may appear straightforward, implementing secure WORM devices is highly complex. The traditional approaches use continuous feed printers [3] optical devices [7], content addressed storage [8] or specially designed hardware not available for the general public [5]. There are also numerous distributed approaches to address this problem (e.g. [6]).

In addition to these considerations, the possibility of a cyberattack on the machine storing the data must be taken into account. If the attacker takes control of the system, she should not be able to delete or modify the WORM data of the log. At this point, all solutions based on file system capabilities may fail: If the attacker elevates privileges, she can change the file system configuration or simply modify or delete the files, the data blocks (directly from the block device), the address of the data blocks if it is content-addressed or format the file system. Distributed systems also become a problem: a denial of service attack compromises the ability to write the to the logs when they are most needed.

We propose Socarrat, a radical and cost-effective solution to address this problem locally with a simple external usb device: A small single board running Linux with USB OTG support. For example, a Raspberry Pi with Socarrat can export a 1 TB USB mass storage WORM device, which can be mounted on any regular operating system as an ext4 or exFAT file systems (i.e. without running special software), for approximately $100.

Socarrat is based on a novel if somewhat contorted approach: The ***Reverse File System***. This approach consists of analyzing the blocks written to the storage device to infer the file system operations executed at the upper layers. Note that what is being exported is a block volume, which will be mounted in the user's machine as a common file system. Socarrat only takes into account write operations at the end of the logs, ignoring any other operation.

Example Scenario. Alice connects a USB black box (i.e. Socarrat running on a Raspberry Pi) to her server. The operating system of the server detects a USB mass storage device formatted as an ext4 or exFAT file system. Then, this drive is mounted in the system. In the mount point, there is a file named `log` (there can be more than one, we use this one as an example). The applications running in the server are able to use this volume as a regular one, by performing the traditional system calls to work with the files. The only difference is that, when the applications access the `log` file, only read operations and append-only write operations are effective; the rest of the write operations on it are discarded. Later, she can extract the `log` file from the device, along with an additional file that authenticates all the entries added to the log during this period. Using a diagnostic tool, she or a third party (the auditor) can verify that the log data corresponds to the entries made during the operational period, ensuring that no records have been deleted or tampered with.

Research Problem. It is worth noting that building a secure WORM is not a trivial problem. Within the USB black box, the volume served to Alice's machine cannot be mounted locally to check for differences and discard non valid writes, because there would be serious concurrency problems: The volume would be mounted in two systems at the same time (i.e. in the Raspberry Pi and the user's machine).

Therefore, we must follow the reverse file system approach: we have to analyze the block updates carefully and infer the write operations being performed on the data structures and blocks of the file system.

This is complex. For example, when a block write appears in an ext4 file system, we have to determine if it modifies the inode or the blocks (be it data blocks or intermediate data structure blocks like extents) of one of the log files for which we give WORM guarantees. In case it is, we have to check for the integrity of the tree and make sure we are not in the middle of a partial update. In that case, we can infer the data of a write (probably coalescing more than one) operation on the file. Journaling can help, but it also makes thing even more complicated.

The Continuous Printer Model. Our goal, in case of a total compromise of the user's machine after a remote logical attack, is to minimize the attack surface to modify the data already stored in the log file. It is restricted to the USB link. Moreover, if the attacker ultimately succeeds in modifying the data already in the log file, the attack must be detected. In order to provide such forward-integrity properties, it uses a previous system named SealFS [4, 10].

One of the challenges of such a system is modeling precisely the guarantees our system offers. Given the integrity of the USB link (it can only be used as a mass storage device), the guarantees Socarrat provides, are formalized by what we have named the The Continuous Printer Model (CPM):

1. Once committed, data cannot be deleted/rewritten.
2. Liveness (now and then we commit something while the system works: writes do not wait forever).
3. We do not guarantee that spurious or bad data is not committed in the future, or that the system cannot be stopped from *printing* by breaking it, but guarantee 1 is always preserved: What is *printed* cannot be *unprinted*.

Architecture. The general architecture of our system is depicted in Figure 1:

- `Soca` is the our main component, written in Go, that implements the reverse file system approach.
- The `nbd` protocol is used to export the block device over the OTG USB link.
- A file system image (ext4 or exFAT) is used as backend for `soca`. It contains the file system exported to the user's machine.
- `SealFS` [4, 10] is the component that provides tamper-evident mechanisms for the protected log.

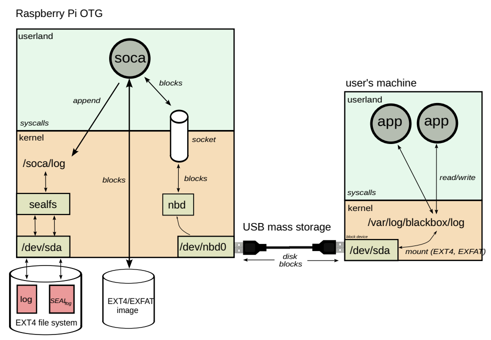

Figure 1. General architecture of the Socarrat system.

Related work. The idea of a reverse file system is quite unconventional and there is very little in terms of comparable systems or devices. The closest two we found are related to one another (one was inspired by the other). The first one is vvfat [2]. Vvfat is a block driver served by qemu which serves a virtual VFAT file system in which the files represent the files of an underlying directory and follow them when read an written. Inspired on this, there is the nbdkit floppy plugin [1]. These two devices implement a kind of *synthetic reverse file system*. They have two main differences with Socarrat:

1. These systems are unconstrained by a liveness property. They only need to guarantee the underlying files are completely synchronized when they are unmounted. Our reverse file system needs to extract valid operations continuously in order to work.
2. The second is an implementation difference. A *reverse file system* observes the data structures present in a block device image. A *synthetic reverse file system* invents these data structures on the fly.

Performance. Note that our system is designed just for secure logging, it may not be suitable for high performance IO purposes.

To measure the if and how well the system works, we have the following challenges to meet:

1. The system is correct and works well on different operating systems when not under attack, and fails gracefully under attack (under the CPM constraints).
2. The system is sufficiently fast to be able to write logs at the speed needed for different applications.
3. We preserve the liveness guarantee and there is no bottleneck on the system which stops the logs from getting updated.

Currently, we have a working prototype on a Raspberry Pi 4 and are in the process of testing and measuring it.

References

[1] Nbd virtual floppy disk from directory. URL https://libguestfs.org/nbdkit-floppy-plugin.1.html. [Online; accessed jan-2025].
[2] Qemu block driver for virtual vfat. URL https://github.com/qemu/qemu/blob/master/block/vvfat.c. [Online; accessed jan-2025].
[3] M. Bellare and B. S. Yee. Forward integrity for secure audit logs. Technical report, University of California at San Diego, 1997.
[4] G. Guardiola-Múzquiz and E. Soriano-Salvador. Sealfsv2: combining storage-based and ratcheting for tamper-evident logging. *Int. J. Inf. Secur.*, 22(2):447–466, Dec. 2022. ISSN 1615-5262. doi: 10.1007/s10207-022-00643-1.
[5] J. Leppäniemi, T. Mattila, T. Kololuoma, M. Suhonen, and A. Alastalo. Roll-to-roll printed resistive worm memory on a flexible substrate. *Nanotechnology*, 23(30):305204, 2012.
[6] M. Li, C. Lal, M. Conti, and D. Hu. Lechain: A blockchain-based lawful evidence management scheme for digital forensics. *Future Generation Computer Systems*, 115:406–420, 2021.
[7] S. Quinlan. A cached worm file system. *Software: Practice and Experience*, 21(12):1289–1299, 1991. doi: https://doi.org/10.1002/spe.4380211203.
[8] S. Quinlan, J. McKie, and R. Cox. Fossil, an archival file server. *World-Wide Web document*, 2003.
[9] R. Sion. Strong worm. In *2008 The 28th International Conference on Distributed Computing Systems*, pages 69–76, 2008. doi: 10.1109/ICDCS.2008.20.
[10] E. Soriano-Salvador and G. Guardiola-Múzquiz. Sealfs: Storage-based tamper-evident logging. *Computers and Security*, 108:102325, 2021. ISSN 0167-4048. doi: 10.1016/j.cose.2021.102325.

Funding. This work is funded under the Proyectos de Generación de Conocimiento 2021 call of Ministry of Science and Innovation of Spain co-funded by the European Union, project PID2021-126592OB-C22 CASCAR/DMARCE.