ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

A Secure, Fast, and Resource-Efficient Serverless Platform with Function REWIND

Jaehyun Song[1], Bumsuk Kim[1]*, Minwoo Kwak[2], Byoungyoung Lee[3], Euiseong Seo[1], Jinkyu Jeong[2]
[1]*Sungkyunkwan University,* [2]*Yonsei University,* [3]*Seoul National University*
{jaehyun.song, bumsuk.kim}@csi.skku.edu, {minwoo.kwak, jinkyu}@yonsei.ac.kr,
byoungyoung@snu.ac.kr, euiseong@skku.edu

Abstract

Serverless computing often utilizes the warm container technique to improve response times. However, this method, which allows the reuse of function containers across different function requests of the same type, creates persistent vulnerabilities in memory and file systems. These vulnerabilities can lead to security breaches such as data leaks. Traditional approaches to address these issues often suffer from performance drawbacks and high memory requirements due to the extensive use of user-level snapshots and complex restoration process.

The paper introduces REWIND, an innovative and efficient serverless function execution platform designed to address these security and efficiency concerns. REWIND ensures that after each function request, the container is reset to an initial state free of any sensitive data, including a thorough restoration of the file system to prevent data leakage. It incorporates a kernel-level memory snapshot management system, which significantly lowers memory usage and accelerates the rewind process. Additionally, REWIND optimizes runtime by reusing memory regions and leveraging the temporal locality of function executions, enhancing performance while maintaining strict data isolation between requests. The prototype of REWIND is implemented on OpenWhisk and Linux and evaluated with serverless benchmark workloads. The evaluation results have demonstrated that REWIND provides substantial memory savings while providing high function execution performance. Especially, the low memory usage makes more warm containers kept alive thereby improving the throughput as well as the latency of function executions while providing isolation between function requests.

1 Introduction

Serverless computing, also known as Function-as-a-Service (FaaS), has significantly gained traction in cloud computing, evident from its adoption by major cloud vendors. This shift from traditional virtual machine-based cloud computing or Infrastructure-as-a-Service (IaaS) offloads system operation responsibilities like auto-provisioning, auto-scaling, and load-balancing from the client to the platform itself [26, 36]. This model frees clients from the complexities of managing the entire software stack, from operating systems (OSs) to applications, allowing them to focus more on application or function development. Driven by the benefits of faster development and continuous integration/continuous delivery (CI/CD) processes [31], serverless computing has become increasingly crucial.

The FaaS platforms usually process requests for a serverless function in containers dedicated to that function [39]. This design allows function's processing capability to dynamically scale in or out, responding to changes in request demand. To support this, function containers must be stateless, and function's data should be ephemeral, being maintained only for the duration of a single request execution. This architecture fundamentally provides robust security. The use of multiple containers offers isolated execution environments across requests. Consequently, even if one container is compromised, its impact does not extend to requests being processed in other containers. Additionally, function containers' ephemeral nature further enhances security, as any residual memory or file data within a container disappears quickly.

Containers require hundreds of milliseconds to launch [17]. To remove this startup latency of function containers out of the function execution, serverless platforms typically reuse these containers for subsequent requests of the same function [6, 18, 19, 33, 37, 53, 58]. Unfortunately, this practice of employing *warm* containers may create security vulnerabilities, such as the potential exposure of sensitive data or the risk of malware attacks from previous requests [4, 13].

One of the most straightforward solutions to address these security threats is to restore function container's state to its initial one before processing the next request. The FaaS platform should checkpoint the initial state of the container as a snapshot when it is first launched, and restore function container's state from this snapshot after the execution of

*Currently at Samsung Electronics

the sandbox's file system or in the background tasks. This demonstrates REWIND 's capability to effectively remove both persistence points—the file system and tasks.

5 Related Work

Snapshot Function Sandbox Booting. Many studies have proposed to reduce the startup latency of a function container by exploiting the snapshot approach [5,59]. The use of a snapshot can avoid container initialization time. REAP [59] and Faasnap [5] have improved the cold startup time of a container by characterizing and prefetching pages actually used during snapshot booting. Although these approaches make use of a snapshot, their goal is not to provide isolation, but to accelerate the booting time of a function container. Such snapshot boot methods are usually integrated with microVMs [5,59], which are lightweight VM-based sandboxes [1]. REWIND is orthogonal to these approaches. A snapshot can contain REWIND's feature, which always rewinds the function's state to the pristine state before serving a function request.

Serverless Security. A few studies tried to address the security vulnerabilities of serverless computing [12, 13]. Valve [12] proposed to prevent privacy-sensitive data leakage by tracking taints and allowing them to pass through only authorized communication paths. ALASTOR [13] extends serverless security by auditing behaviors of the entire functions/sandboxes distributed servers. A few studies have proposed to build secure containers from privileged components using Intel SGX [3,8,49]. REWIND's purpose is orthogonal to confidential computing (e.g., Intel SGX), which would further strengthen the security of REWIND if combined.

Secure Sandbox. Enforcing isolation in a shared sandbox is important and many studies proposed to provide isolation across various resources [2,25,41,64]. Chancel [2] proposed multi-client isolation in a single enclave using per-thread memory region and software fault isolation. This work is similar to our work in terms of enforcing isolation between multiple end users. However, this work assumes the concurrent handling of multi-end users in a single container, which is not allowed in serverless computing. TxBox [25] exploited system transactions for secure execution in a sandbox. It detects and aborts any system transactions by malicious users using the TxOS's system transaction feature [48]. This work preserves privacy-sensitive data created or modified by allowed system transactions, which do not prevent persistence properly, but possible in REWIND. PRIVEXEC [41] allows private execution in a shared sandbox by providing secure and isolated storage access. This work is limited to preventing the persistence of a file.

Checkpoint and Restore. Taking a snapshot of a file system is supported in various file systems, including btrfs [51] and ocfs2 [40], as well as user-level tools, such as git [20]. Checkpoint and restore (C/R) [7] is a more general technique to take a snapshot of a set of processes and restore them when it is necessary. Many studies have been conducted to utilize C/R for fault tolerance of long-running high-performance computing applications [14,15,38,47]. CRIU [11] is a user-level tool for Linux that provides C/R of processes and containers. The original CRIU is slow because the snapshot is stored in the secondary storage. However, VAS-CRIU [60] extended CRIU to store a snapshot in memory, reducing the C/R latency. Both are general-purpose C/R techniques that freeze the entire set of processes and take a snapshot of all the resources and all possible kernel-provided states processes might hold. Recently, Groundhog [4] has proposed C/R specialized to serverless computing. As our evaluation results have shown, the memory and performance cost of C/R-based approaches are high as compared to REWIND since REWIND's snapshot feature is supported by the OS kernel and REWIND exploits the temporal locality of function execution to accelerate the next function execution.

6 Conclusions

The lack of temporal isolation exposes modern serverless computing to potential data leakage or exfiltration attacks. The reuse of warm function sandboxes is effective in improving the function execution performance. However, this comes at the risk of allowing persistence points in a sandbox that can be exploited by attackers. The main cause of this problem is the lack of temporal isolation within a shared reused sandbox.

In this paper, we propose REWIND to address the lack of temporal isolation. REWIND rewinds the state of a function sandbox back to its pristine state whenever it completes handling a function request. This allows us to eliminate the persistence points mistakenly produced by the use of a warm sandbox. This eliminates the possibility of leaving privacy-sensitive data in a sandbox. Consequently, REWIND provides the temporal isolation between consecutive function requests. Our evaluation results demonstrated that REWIND provides the temporal isolation at a low cost as compared to the state-of-the-art approach.

Acknowledgments

We would like to thank the anonymous reviewers and shepherd, for their valuable comments. This work was supported in part by Institute for Information & communications Technology Promotion (IITP) grant and by the National Research Foundation of Korea (NRF) grant all funded by the Korea government (MSIT) (No.2021-000773, RS-2023-00321688). Jinkyu Jeong is the corresponding author.

Availability

The source code of the REWIND prototype is available at https://github.com/s3yonsei/rewind_serverless.