Kubernetes & Etcd Are Not Secure, until Now

2025-10-14

Abstract

Modern computation is increasingly trending towards Internet cloud computing performed by containers on shared rented powerful server computers managed from a central third-party cloud server provider (Amazon Web Services, Microsoft Azure, Google Cloud Pages, and more) serving content and computation to many client computers, as it is much easier to outsource all server hosting needs to a third-party (especially for individuals and smaller organizations with tight budgets). The traditional security measure to protect sensitive data in-transit and at-rest from both network eavesdroppers and the cloud providers themselves is end-to-end (E2E) encryption, but because of modern inference/frequency analysis attacks discovered in the early 2010s, we nowadays need oblivious computation in addition to the standard end-to-end (E2E) encryption for safe and scalable cloud computing. To add oblivious computation to Kubernetes and Etcd for safe and scalable cloud computing is the aim of my Master Capstone Project, Locker 2.0.

Problem Motivation

Kubernetes is a container manager, designed by the Linux Foundation for the easy scalability of distributed computing resources on shared computer servers, and it is commonly used in a cloud environment for resource management [1]. Etcd is a key-value data storage, designed by the Linux Foundation for the easy scalability of reliable key-value data storage, and is commonly used in a cloud environment for data storage [2]. Modern Internet connections are end-to-end (E2E) encrypted with the Hypertext Transfer Protocol Secure (HTTPS) protocol, which is the Hypertext Transfer Protocol (HTTP) over Transport Layer Security (TLS) protocol using public key/digital certificates signed by widely trusted third party certificate authorities (CAs) [3]. HTTPS provides end-to-end (E2E) encryption for data traveling over the Internet, only leaking metadata (which is still a concern, as we shall see later). Kubernetes does have an option to enable at-rest encryption for Secrets (password, tokens, and keys), which encrypts data such that it is stored to cloud providers as ciphertext (which is still a concern, as we also shall see later) [4]. Etcd is, by default, unencrypted without the

usage of a third-party encryption measure (such as Kubernetes Secrets), which makes it unsuitable to store any sensitive information without such measures. Therefore, by default, Kubernetes and Etcd over the Internet is not safe for cloud computing. This vulnerability is due to cloud providers being able to use frequency/inference analysis attacks on ciphertext and metadata using the information leaked from data access patterns, like in Muhammad Naveed et al. [5].

Solution Introduction

Oblivious computation is a subfield of cybersecurity that focuses on obscuring data access patterns to prevent the leakage of data access patterns, assuming the existence of one-way functions. It is typically accomplished by the client continuously encrypting and shuffling data around, such that the adversary cannot tell if any given data access was "real"/desired or "fake"/deceptive, thus preventing the adversary from performing any inference/frequency analysis attacks (although there is still the possibility of performing side-channel attacks and currently unknown attacks) [6]. It was first founded by Oded Goldreich & Rafail Ostrovsky, who wanted to safeguard commercial software systems against software piracy in the 1980s and 1990s. They devised the concept of oblivious random-access memory (ORAM), which is a create-read-update-delete (CRUD) data structure designed to hide data access patterns from any adversary that can observe, repeatedly run, and tamper with the computer system. It worked, but it was prohibitively slow for any practical usage (especially for the much slower computers of those times), as it had a time complexity of $O(t \log t^3)$. There was a long gap in interest in oblivious computation from the global cybersecurity community until the early 2010s, with Mohammad Saiful Islam et al. in 2014 publishing a research paper containing their new inference/frequency analysis attack, called the IKK attack, that broke encrypted databases' security [7]. The cybersecurity community quickly took notice of the new IKK attack, with Muhammad Naveed et al. in 2015 and Paul Grubbs et al. in 2017 both publishing follow-up research papers that refined the IKK attack's technique and managed to break more encrypted databases [8]. This was becoming a perturbing problem for the cybersecurity community with no clear defense against such inference/frequency analysis attacks, so urgent work was needed on this pressing problem.

Solution Definition

Emil Stefanov et al. published a research paper in 2013 that used a simple yet innovative binary tree data structure, called PathORAM, to make Oded Goldreich & Rafail Ostrovsky's ORAM performance practical, reducing its time complexity to just $O(\log n^2)$ [9]. This was a major advancement for the subfield of oblivious computation that arrived at precisely the right time, as the Mohammad Saiful Islam et al. paper was published the following year and PathORAM was the only viable defense against the IKK attack. Cybersecurity

researchers rushed to follow-up on the PathORAM data structure, but we will be highlighting only two follow-up research papers for now, Daniel S. Roche et al's 2016 vORAM+HIRB oblivious data structure (ODS) and Alexandra Boldyreval & Tianxin Tang's 2022 Encrypted Multi-Map (EMM) oblivious data structure (ODS), as we will need to know them to understand how Locker 2.0 adds oblivious computation to Kubernetes and Etcd [10] [11]. Daniel S. Roche et al. focused on extending the practical usability of PathORAM to be a generic oblivious map/dictionary that leaks few data access patterns while still retaining a time complexity of $O(\log n^2)$ with minor additional overhead. On the other hand, Alexandra Boldyreval & Tianxin Tang concentrated on extending the practical usability of PathORAM and vORAM+HIRB to be a generic oblivious multi-map that leaks no data access patterns while being slightly slower than PathORAM with a time complexity of $O(m \log n^4)$. These data structures are integrated into Locker 2.0 to provide the backbone of the oblivious computation needed for safe and scalable cloud computing [12]. For more information, my Master Capstone Report has several Sections discussing the technical details of these research papers.

Solution Implementation

Locker 2.0 was implemented in 5 Phases, which took a total of 10 weeks to implement. The source code of Locker 2.0 is free and open source (FOSS) under the Creative Commons Attribution 4.0 International license and is available on GitHub. For additional information, my Master Capstone Report has several Sections discussing the technical details and performance of my Locker 2.0 implementation.

1. Phase 1: Develop a generic plaintext Etcd client-server communication protocol in Golang, suitable for integration with Kubernetes.
2. Phase 2: Incorporate an existing C++ PathORAM implementation into the plaintext Etcd protocol as an oblivious RAM (ORAM) data structure.
3. Phase 3: Extend the PathORAM implementation into using vORAM+HIRB as an oblivious map/dictionary.
4. Phase 4: Include the Python vORAM+HIRB implementation into using EMM as an oblivious multi-map (the EMM implementation had to be written from scratch in Golang, as at the time there were no FOSS implementations of EMM available on the Internet).
5. Phase 5: Re-engineer the generic plaintext Etcd client-server communication protocol to use EMM to secure its data from any possible attack, except for side-channel attacks and currently unknown attacks.

Conclusion

The normal Kubernetes container manager & Etcd key-value data storage cannot be used securely in an Internet cloud computing environment, as their end-to-end (E2E) encryption measures are not good enough to prevent cloud

providers from performing inference/frequency analysis attacks on encrypted data, thus compromising the security and privacy of all users. Therefore, an additional subfield of cybersecurity, called oblivious computation, must be used together with the standard end-to-end (E2E) encryption policies for safe cloud computing. That is the reason behind my Master Capstone Project and Master Capstone Report, Locker 2.0, which is designed as a drop-in Etcd client-server communication protocol for usage with Kubernetes. Locker 2.0, when used with standard information security practices (such as the NIST Cybersecurity Framework 2.0), provides secure and scalable cloud computing that is currently invulnerable to any attack except for side-channel attacks and currently unknown attacks [13].

References (IEEE Style)

1. "Overview", Kubernetes. [Online]. Available: https://kubernetes.io/docs/concepts/overview/.
2. "What Is etcd?", IBM. 17-Apr.-2025. [Online]. Available: https://www.ibm.com/think/topics/etcd.
3. "What is HTTPS?", [Online]. Available: https://www.cloudflare.com/learning/ssl/what-is-https/.
4. "Secrets", Kubernetes. [Online]. Available: https://kubernetes.io/docs/concepts/configuration/secret/.
5. M. Naveed, S. Kamara, and C. V. Wright, "Inference Attacks on Property-Preserving Encrypted Databases," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver Colorado USA: ACM, Oct. 2015, pp. 644–655. doi: 10.1145/2810103.2813651.
6. O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," Journal of the ACM, vol. 43, no. 3, pp. 431–473, May 1996.
7. M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Inference attack against encrypted range queries on outsourced databases," in Proceedings of the 4th ACM Conference on Data and Application Security and Privacy. San Antonio Texas USA: ACM, Mar. 2014, pp. 235–246.
8. P. Grubbs, T. Ristenpart, and V. Shmatikov, "Why Your Encrypted Database Is Not Secure," in Proceedings of the 16th Workshop on Hot Topics in Operating Systems. Whistler BC Canada: ACM, May 2017, pp. 162–168.
9. E. Stefanov, M. van Dijk, E. Shi, T.-H. H. Chan, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path ORAM: An Extremely Simple Oblivious RAM Protocol," Journal of the ACM, vol. 65, no. 4, pp. 1–26. 10. D. S. Roche, A. Aviv, and S. G. Choi, "A Practical Oblivious Map Data Structure with Secure Deletion and History Independence," in 2016 IEEE Symposium on Security and Privacy (SP). San Jose, CA: IEEE, May 2016, pp. 178–197.
10. D. S. Roche, A. Aviv, and S. G. Choi, "A Practical Oblivious Map Data Structure with Secure Deletion and History Independence," in 2016 IEEE Symposium on Security and Privacy (SP). San Jose, CA: IEEE, May 2016, pp. 178–197.
11. A. Boldyreva and T. Tang, "Encrypted Multi-map that Hides Query,

Access, and Volume Patterns," in Security and Cryptography for Networks, C. Galdi and D. H. Phan, Eds. Cham: Springer Nature Switzerland, 2024, vol. 14973, pp. 230–251.
12. I. Ahmed, "Master Capstone Project: Locker 2.0 - Oblivious Computation for the Etcd Key-Value Datastore". Jun. 2025. [Online]. Available: https://personal-website-3bm.pages.dev/CSE_247B_Master_Capstone_Project_Ismail_Ahmed.pdf.
13. C. Pascoe, S. Quinn, and K. Scarfone, "The NIST Cybersecurity Framework (CSF) 2.0", NIST. 26-Feb.-2025. [Online]. Available: https://www.nist.gov/publications/nist-cybersecurity-framework-csf-20.

Footnotes

- I am a recent honors graduate from the University of California, Santa Cruz (UCSC) with a Master of Science in Computer Science & Engineering, a Bachelor of Arts in Economics, and a Bachelor of Science in Computer Science.
- To contact me with any further questions, please see my LinkedIn, GitHub, and email for reference. The best way to reach me is through email.
- For a permanent downloadable PDF verison of this blog post, here is the link.
- Thank you, reader, for taking the time out of your day to read my personal blog. I hope to provide you with informative and useful content.