Decentralized backup of logging data in cloud applications using IPFS

Mahid Atif Hosain
Department of Computer Science & Engineering
Brac University
Dhaka, Bangladesh
mahid.atif.hosain@g.bracu.ac.bd

Zunaira Khan
Department of Computer Science & Engineering
Brac University
Dhaka, Bangladesh
zunaira.khan@g.bracu.ac.bd

Tarik Bin Shams
Department of Computer Science & Engineering
Brac University
Dhaka, Bangladesh
tarik.bin.shams@g.bracu.ac.bd

Department of Computer Science & Engineering
Brac University
Dhaka, Bangladesh
@g.bracu.ac.bd

***Abstract*: Cybercriminals can make use of application logging data to find vulnerabilities in a system. To protect the user or organization from security breaches, identity theft and data leaks, encrypting logging data is crucial for data accountability and privacy. In this paper, we propose a model utilizing InterPlanetary File System protocol (IPFS) to build a more secure system for distribution and backup of access logs. The peer-to-peer nature of IPFS helps reduce complexity of external control mechanisms. The data is encrypted using AES 256 bit. We create a private IPFS cluster utilizing swarm key that will allow it to recognize only nodes that are part of the private network. A private IPFS means we control the nodes and our encrypted data as some laws require complete deletion of user information which can include logging data.**

Keywords— IPFS, cloud storage, decentralized, peer-to-peer, encryption

I. Introduction

More and more companies are using cloud servers to launch their applications, which has led to the need for increased security measures to protect against malicious activity. Cloud service providers (CSPs) store their own databases and parameters for logging network activities performed by their customers using virtual machines (VMs). In the event of a forensic investigation, the CSP, user, and Cloud Forensic Investigator (CFI) are all involved. However, the investigator must blindly trust the credibility of the logs provided by the CSP and base their conclusions on them in court. There is a risk of collusion between the parties involved, and if tampering is discovered, it is impossible to recover the original data or logs. According to the SOX law, financial institutions must keep a trustworthy log of their IT-based activities [3], which can be difficult to ensure if they use an in-house cloud platform where the IT department is in control of the logs and could potentially tamper with them. Therefore, a trustworthy logging service is necessary.

It is not only a privacy concern but it can also be used to find vulnerabilities in a system and reveal crucial information required to infiltrate a system. Logging data is usually labeled for it to be useful for monitoring of different services but this type of data can become dangerous in the hands of a bad actor. different studies have been made to make use of the logging data for data leaks [1].

Logging data can contain personal information included as metadata like the IP address of a request . but according to the 'right to erasure' article of GDPR [2]. If any user wants to have their data removed, the CSPs are required to delete all and any data immediately which includes log files but it is quite difficult to implement in a distributed system. This is where the IPFS (The InterPlanetary File System) comes into play as an autonomous decentralized file system IPFS stores data as immutable objects and assigns a CID (content identifier) we store the CID in a system called IPNS (InterPlanetary Name System) which provides a mutable pointer for the immutable CID.

The InterPlanetary File System (IPFS) is a protocol that allows for the creation of a decentralized file system through a peer-to-peer network. It provides a reliable and secure method of storing and sharing data, without the need for a central authority or trust between nodes. IPFS is particularly useful for storing data that cannot be easily stored on a distributed ledger technology (DLT). When a file is published on the IPFS network, it becomes an IPFS object and can be identified by a unique file digest, which is generated through the use of a hash function. To access an IPFS object, one only needs to use the file digest as a reference [4].

AES (Advanced Encryption Standard) is a widely used symmetric encryption algorithm. AES uses a fixed block size of 128 bits and supports key sizes of 128, 192, or 256 bits. It is a block cipher, which means that it operates on fixed-sized blocks of data, and uses the same key for both encryption and decryption. One of the main features of AES is that it is very fast and efficient, making it suitable for a wide range of applications. It is also very secure, as it has been extensively analyzed and tested by cryptanalysts around the world [5].

II. Research Objective

In this paper, we propose a logging backup mechanism using IPFS. We build a private IPFS cluster using Docker compose that allows us to provide reliability by hosting the IPFS nodes privately using any cloud provider as we create the service using cloud agnostic technology like Docker. The logging data files are stored encrypted so our data is secured even if any bad actor gets access to the data. The main contributions of the proposed work are as follows.

- Develop a cloud backup solution for logging data.
- Reduce information abuse, hacking, fraud, and other financial crimes.
- Encrypt logging data using the RSA 256 bit encryption algorithm.
- Minimize the burden of complying with regulations when handling clients' sensitive information.

III. Literature Review

The authors in [5] have proposed a method for breaking the 9-round version of the AES-256 encryption algorithm in which the entire 256-bit key can be determined in a time of 2^{39} by only using the most basic type of related keys (where the plaintext being encrypted is done so using two keys with a variable XOR difference) which is much smaller than 2^{256}, the time it should take to break the cipher. The methods were not practical as it requires special conditions to be met.

In reference [6], the authors suggested a method for protecting against tampering and preserving privacy in cloud forensics. This approach creates evidence of the existence of process records. The proposed architecture includes a Merkle root for creating blockchain receipts, which can aid in data validation.

In this paper [7], the authors suggested a method for ensuring the security and traceability of data in the cloud, which involves using two layers of encryption in order to increase privacy. This approach requires more computational resources than other methods.

In reference [8], the authors introduced a system called Provchain, which aims to track and verify the origin of data in the cloud by incorporating it into blockchain transactions. To do this, the architecture employs the open-source cloud platform ownCloud to gather provenance information, and utilizes the Tieron data API [9] to interface with the blockchain network.

Tosh et al. [10] created a system called ProvChain that uses blockchain technology to record the history and origin of data stored in the cloud, while also prioritizing user privacy and ensuring the availability of the data. They tested the system using open source storage applications, but did not evaluate it using a cloud operating system.

In reference [11], the authors described a method for maintaining the confidentiality and integrity of provenance data using encryption and digital signatures. This method is compatible with existing provenance systems and is not dependent on the specific format or model used for the provenance data. However, the SPROVE-2 system does not have the ability to query the provenance data.

Patil et al. [12], the authors proposed a data provenance assurance mechanism that utilizes both Blockchain and IPFS technology. The data is stored in IPFS and the hash of the data is stored in Ethereum based distributed ledger. They also show the scalability and performance analysis of the proposed mechanism.

IV. Proposed Methodology

The design that we propose in this paper is based on IPFS as the main storage layer as it is distributed and immutable as seen in Figure 1. The data is encrypted by AES 256 bit and the hash of the data is stored in IPNS, which is a system for storing mutable pointers for our immutable data.

A. IPFS Cluster

We have used docker compose for setting up 3 IPFS clusters, each having a single node on a local machine. docker allows us to create and deploy IPFS nodes or clusters independent of cloud providers as it is OS agnostic. We have set the replication factor to maximum as we want our data to be available from all the nodes. The nodes are secured using secret keys and to make sure the nodes do not connect to other nodes on the internet we have set a swarm key which would allow us to control our nodes privately and delete the data on demand. We have also set all the CID to be pinned so that the garbage collector does not delete any data without permission.

Using external IPFS nodes would mean we lose complete control over the data , even though the data is encrypted, according to GDPR , if a user requests their data to be deleted, we need to delete the data immediately. The API for the cluster also supports Basic Authentication. In the cloud environment where most of the logging data is not more than 10 MB, IPFS can handle much larger files as it converts the data into smaller chunks internally.

In implementations where public IPFS nodes are used, they do not have complete control over the replication factor and as most of the public nodes are not replicated efficiently, it would cause latency when requesting the files from different regions [13].

B. IPFS Client Application

We use Node.js as our backend runtime as it is one of the most widely used backend runtimes. The application uses the MFS (Mutable File System) which is a tool included with IPFS that allows us to do normal file operations like copy, edit and delete and rename. The application connects to the local IPFS cluster and encrypts a local file or files on the user computer or any path that is defined in the application. The encryption is done using AES 256 bit rather than an asymmetric key algorithm like RSA because the time required for it is much higher than AES and it would require us to store the private key of each of our machines in a separate record and it would not be practical in cloud environments where the containers are temporary.

The AES key is stored as an environment variable so that it can be injected into containers when it is deployed on the cloud. When using AES all the applications will use the same key so it ensures that duplicate data will produce the same output when using the same key but in RSA a different public key would mean different output regardless of the date being encrypted so it is more efficient to use AES.

The file will be encrypted and its data will be uploaded to an IPFS node. Each time the file is changed, a different CID (Content Identifier) will be produced. This CID will be added to IPNS (InterPlanetary Name System) to keep track of the latest version of the file. The CID can be mapped to a human readable DNA record. If the CID currently associated with the file does not match the CID that is resolved through IPNS, it indicates that the data has been altered. This process helps to ensure the integrity of the log files by detecting any changes to the data. The IPNS node can be local or it can be a public node because the CID will not be found on any public node.

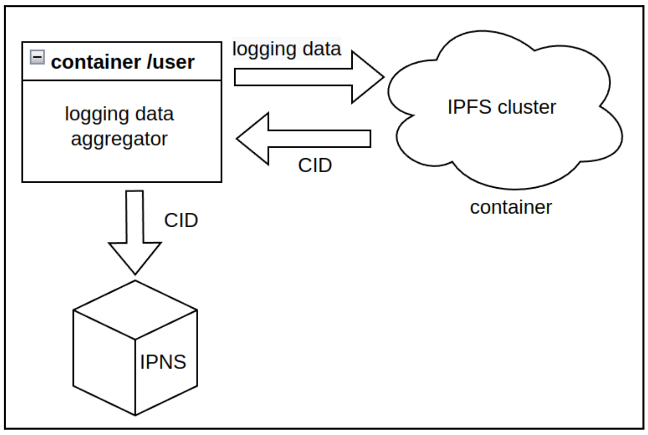

Fig. 1. An overall view of the system architecture

V. Conclusion

In this paper we presented a design and implementation of a decentralized backup system of logging data in cloud applications using IPFS. We developed the client application using Node.js. It uploads the encrypted logging data to the IPFS nodes, which were privately hosted and published the returned CID to IPNS.

As for future work, we would like to implement the client application in Golang as it scales with multiple processor cores and work on better distribution of workload of the cluster nodes. We aim this model will be able to contribute to the field of distributed computing systems in terms of its use of low resources.

References

[1] Ávila, R., Khoury, R., Khoury, R., & Petrillo, F. (2021). Use of security logs for data leak detection: a systematic literature review. Security and communication networks, 2021..

[2] Council of European Union, “Regulation (eu) 2016/679 - directive 95/46,” pp. 1–88.

[3] United States Code (2002). Sarbanes-Oxley Act of 2002, PL 107-204, 116 Stat 745 Codified in Sections 11, 15, 18, 28, and 29 USC .

[4] Trautwein, D., Raman, A., Tyson, G., Castro, I., Scott, W., Schubotz, M., Gipp, B., & Psaras, Y. (2022). Design and evaluation of IPFS. Proceedings of the ACM SIGCOMM 2022 Conference. https://doi.org/10.1145/3544216.3544232

[5] Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., & Shamir, A. (2010). Key Recovery Attacks of Practical Complexity on AES-256 Variants with up to 10 Rounds. Advances in Cryptology – EUROCRYPT 2010, 299–319. https://doi.org/10.1007/978-3-642-13190-5_15

[6] Tosh, D., Sengupta, S., Kamhoua, C. A., & Kwiat, K. A. (2018). Establishing evolutionary game models for cyber security information exchange (cybex). Journal of Computer and System Sciences, 98, 27-52.

[7] Asghar, M. R., Ion, M., Russello, G., & Crispo, B. (2011, June). Securing data provenance in the cloud. In International Workshop on Open Problems in Network Security (pp. 145-160). Springer, Berlin, Heidelberg.

[8] Aditya, C., Akash, M., Akash, P., Amitkumar, M., Nagarathna, K., Suraj, D., G., Narayan & Meena, S. M. (2020). Claims-Based VM Authorization on OpenStack Private Cloud using Blockchain. Procedia Computer Science, 171, 2205-2214.

[9] Tierion: Blockchain Proof Engine. (n.d.). https://tierion.com

[10] Tosh, D., Shetty, S., Liang, X., Kamhoua, C., & Njilla, L. L. (2019). Data provenance in the cloud: A blockchain-based approach. IEEE consumer electronics magazine, 8(4), 38-44.

[11] Hasan, R., Sion, R., & Winslett, M. (2009). SPROV 2.0: A Highly-Configurable Platform-Independent Library for Secure Provenance.

[12] Patil, A., Jha, A., Mulla, M. M., D.G., N., & Kengond, S. (2020). Data Provenance Assurance for Cloud Storage Using Blockchain. 2020 International Conference on Advances in Computing, Communication &Amp; Materials (ICACCM). https://doi.org/10.1109/icaccm50413.2020.9213032

[13] Costa, P. Á., Leitão, J., & Psaras, Y. (2022). Studying the workload of a fully decentralized Web3 system: IPFS. arXiv preprint arXiv:2212.07375.