LogSentinel Summary White Paper

I. Motivation

Almost every system needs to keep audit logs – information on "who did what and when". For example:

- Who changed the patient's medical history
- Which bank employee viewed the customer bank account
- Which CCTV camera has taken a particular video and when

It's not just important who did it, but also what exactly they changed or viewed and when. The whole set of details (actor, action, entity modified, details of the action) has to be stored. But simply storing it won't be sufficient, as the stored data can be modified – there's a need for securely storing it and guaranteeing its integrity.

Whenever confronted with the need to store an audit log (which is the case for almost all applications), companies go for custom home-grown solutions. In the best case scenario they use some plug-in to their database access technology (e.g. an ORM) that automatically handles modifications. Very rarely these solutions cover the following set of requirements:

- Business-logic level events as well as insert/update/delete/get events have to be stored – it's not sufficient to just see the database rows that were updated, sometimes high-level operations like "basket checkout", "bank transfer initiated" or "medical examination performed" need to be stored as well, as they carry more meaning to the business than a bunch of database table updates.
- The audit-log has to be tamper-proof, i.e. nobody, even system administrators, can alter it without being detected. That way the audit log has a more significant legal strength in courts. And the business can make certain guarantees to their clients.
- The audit-log has to be easily searchable and navigable – anyone with proper access (a high level manager, line manager or even external auditor) can easily see sequences of events that lead to a particular issue

And when speaking of issues – data manipulation attacks, both from insiders and outside attackers, are a serious threat nowadays. Wired has put it in the top security threats prediction for 2016. Without taking additional measures, no business is safe from having their data manipulated for the benefit of other parties. And often without even detecting that until it's too late (this is why NIST recommends using audit logs).

Last, but not least, public sector entities, as well as regulated businesses, need to comply with a set of security regulations. Ticking the "compliance" box may not be that hard, but the liabilities for not protecting customers'/citizens' data are still a serious risk to be taken into account.

Speaking of regulations, the General Data Protection Regulation in the EU mandates that every system that processes personal data (which is true for most systems) must have an audit log and not complying with the regulation may lead to significant fines for any business.

While many companies think they have audit logs, they are almost never properly protected. And as one book on ISO 27001 warns:

> ***System logs need to be protected, because if the data can be modified or data in them deleted, their existence may create a false sense of security.***

There needs to be a product that addresses all of the above concerns and observations, and LogSentinel aims to be such a product.

II. Features

LogSentinel offers the following core set of features

- Easy integration – the RESTful API is very simple and can be plugged in new and existing systems painlessly. Using one of the provided client libraries simplifies this process even further. No need for complex setups or lengthy integration processes.
- LogSentinel runs in the cloud, but can also be deployed on-premise. This gives sufficient flexibility, based on the particular buainess case – small businesses may not need or be able to manage their own additional servers (even though the setup is easy), whereas big enterprises may not wish to trust their sensitive audit logs to a cloud solution.
- The audit log is protected by a technology similar to the way the integrity of the blockchain is protected. No modification to the log can be made without becoming evident almost immediately. This also bears legal strength – e.g. if a given event data is not tampered with, it is guaranteed that the event really happened.
- The authentication-specific API endpoints provide a way for actors to digitally sign their logins, for example, which according to the relevant legislation may give additional legal strength in case an issue is brought to court.
- A user-friendly dashboard is provided for searching and navigating through the audit events – for example “who modified that user account at 6 pm yesterday” or “what exactly did employee X do on his last shift”
- Fraud Detection Capabilities – a set of features that allow the user to define domain-specific rules for detecting anomalous behavior. As soon as suspicious activity occurs notifications are pushed through e-mail or SMS, allowing for further investigation and rapid reaction. Rules can be easily configured within the LogSentinel dashboard by business experts using either the rule engine or LogSentinel’s intuitive Wizard. Administrators or business users can specify a wide range of industry-relevant parameters and deviations that trigger notifications in case of anomalous events, turning LogSentinel into a powerful and versatile monitoring and fraud detection tool.
- Protecting the integrity of all the data – while the integrity of the data itself is not the goal of LogSentinel (as opposed to protecting the integrity of the audit log), having a full secure audit log on all data changes means that the audited data is also protected – you can always trace the data modifications in the log and verify whether a given change occurred “naturally”, or was altered inappropriately.

III. Technology

The technology used is based on a number of peer-reviewed papers, e.g. Audit Logs to Support Computer Forensics by Bruce Schneier and John Kelsey and Efficient Record-Level Keyless Signatures for Audit Logs by Ahto Buldas et al. The implementations is customized to account for the various business cases that should be addressed. Some properties have been dropped or moved to the client/client libraries, and the need for scaling the product and providing multi-tenancy has led to a bit more complicated processing logic.

Notably, the technology relies on consecutively hashing all incoming audit log events, where each subsequent hash is formed by the data of the current entry combined with the hash of the previous entry. That way the audit log entries form a hash chain that cannot be "broken" – i.e. any manipulation to any of the entries will result in invalid hashes from that moment on.

In addition to forming a hash chain, entries form groups, which are timestamped using a client-provided timestamping authority. The group is represented by a single value, which is the root of the merkle tree of the hashes of all the entries in a group. A merkle tree is a data structure used in the blockchain to guarantee the integrity of each block. The timestamping provides additional integrity guarantees and legal strength (EU Regulation 910/2014 defines the rules for trusted timestamping)

The integrity of the log is guaranteed even if a malicious actor gains access to the log database. So you don't have to trust the LogSentiel cloud service, or even your own employees who manage the hosted solution. In order to achieve that, there is one "catch" – the latest hash in the hash chain at a given moment has to be kept in an unmodifiable way. When you have a given hash, the fact that it is present in the audit log guarantees that it hasn't been tampered with. And vice-versa – if a hash that was previously stored is missing from the audit log, it means the log has been tampered with. There are multiple ways to store these latest hashes in an unmodifiable way, and LogSentinel supports:

- Print it on paper – it can be published in newspapers (which is less practical), or printed on a blank paper and stored in physically protected cases, or snail-mailed to multiple stakeholders, including auditors.
- Store it on a write-only medium. Be it a CD-R, or more generally – any WORM storage. LogSentienl returns the latest known hashes which you can decide when and how to store.
- Email it to multiple stakeholders – while email can be manipulated as well, having it distributed to multiple people, potentially with different email servers, increases the complexity of changing the hash in all places.
- Store it on the Ethereum blockchain (alpha). Blockchains are the perfect candidate, as they are immutable – once an entry is stored there, it cannot be removed. Transaction fees are relatively cheap, so LogSentinel can regularly push the last known hashes to the Ethereum blockchain.

IV. Why not …?

The problem of securely storing audit logs is not a strictly defined one. The reasons why certain aspects were not implemented in a particular way are discussed below.

Why not provide built-in anonymization of actors and privacy mechanisms for the data? One of the features of LogSentinel is the ability to easily search, visualize and analyze audited events. Using actor pseudonyms, encrypting the data or using bit masks when computing the trees and hashes (all as suggested in the literature) would yield the search features useless. The premise of most of the papers includes only preserving the integrity of the logs, not analyzing them. Not having the entries fully encrypted would allow machine-learning based risk analysis and alerting on malicious activities. However, the privacy features are not ignored – they can easily be achieved on the client-side, before sending. The RESTful API acknowledges that opportunity and provides an endpoint for that.

Why not provide binary serialization support, in addition to the HTTP API? LogSentniel uses HTTP2, which is a binary protocol. Adding gzip on top of that reduces much of the overhead of typical RESTful APIs and allows for high performance.

Why not use merkle tree for everything? The data model chosen wouldn't benefit much from merkle trees over hash chains – we need to store all the data for all the entries anyway, and looking up each hash is pretty quick due to the performed indexing. We also don't use merkle proofs in the typical use case, as normally there is no need for a thin (verification) client. We do use merkle trees as an additional safeguard against tampering, by having one merkle tree per application and exposing merkle tree primitives like consistency and inclusion proofs. These require additional integration effort but are useful when it comes to trust.

Why not use an entirely public blockchain? The blockchain has features that are not needed in the "tamper-evident audit log" scenario. The distributed consensus algorithm (with proof or work, for example) and the distributed storage are not necessary in order to achieve the goal of ensuring the integrity of audit logs, therefore using a full-blown blockchain would be an overdesign (even though it would sound cool) and would make setting up and maintaining the system more complex. That's why we use only the relevant bits – the hash chaining / merkle trees.

Why not use a custom solution or syslog instead of LogSentinel? Custom solutions rarely cover the necessary features and take time and resources to implement. Using syslog or something like Splunk or Logstash again doesn't cover the security requirements. One can get hash chaining ontop of syslog, as shown by one of the cited papers, but it requires additional development and knowledge on the syslog server internals. LogSentinel is a "drop-in" solution, which is used by a very simple and straightforward API.

Why not use just timestamping? Timestamping guarantees the integrity of the timestamped groups (blocks) of entries, but does not guarantee that no record was inserted with a date in the past or that no group was deleted. The hash chain provides a strong guarantee that there were no modifications on the entire log.

V. Conclusion

LogSentinel is a secure audit log service that is simple to integrate and guarantees the integrity of all your audit data.