Slope Finance

Investigation

Presented by:

OtterSec contact@osec.io

Robert Chen notdeghost@osec.io

Harrison Green hgarrereyn@osec.io

Contents

01 | Executive Summary

Overview

Slope Finance engaged OtterSec to perform an analysis of critical logging infrastructure following a potential breach. This assessment was conducted between August 4th and August 10th, 2022.

Slope Finance provided OtterSec with a snapshot of the logging server taken on August 3rd, hours after the initial attack was discovered. Additionally Slope Finance provided OtterSec access to the Slope Wallet codebase.

Key Findings

1. We did not discover any evidence to suggest that the log storage server had been breached.
2. We confirmed that the Slope wallet since version 2.2.0 (release June 24th) was vulnerable to leak private keys and mnemonics in Sentry logs which would be sent to the log server.
3. Based on docker container activity, we believe that logging ingress via Kafka was not functioning correctly prior to July 28th, meaning that no logs were stored on the server between May 3rd and July 28th.
4. Analyzing the logs stored from July 28th to August 3rd (which represent 14% of the total potential vulnerable period), we discovered private keys for 12% of the hacked addresses.

Slope's server stored sensitive information for 15% of the exploited wallets (1444 out of 9229). However, it also only received data for 12% of the time the applications were trying to send sensitive data (7 of 41 days). Thus, the number of mnemonics that were sent was likely much greater than the number stored on the server.

Since log history is non-existent on the log server prior to July 28th, we do not think it would be possible for an attacker with direct access to the log server to obtain these logs. We did not evaluate the remainder of the client to server pipeline and cannot comment on vectors of compromise.

02 | Timeline

- June 24th: v2.2.0 released (first vulnerable version)
- July 15th: v2.2.1 released
- July 21st: v2.2.2 released
- July 28th: Docker metadata shows that the Kafka and Zookeeper containers are restarted which re-enables normal ingress and logs begin to be stored on the log server.
- August 3rd: Attacker starts exploit

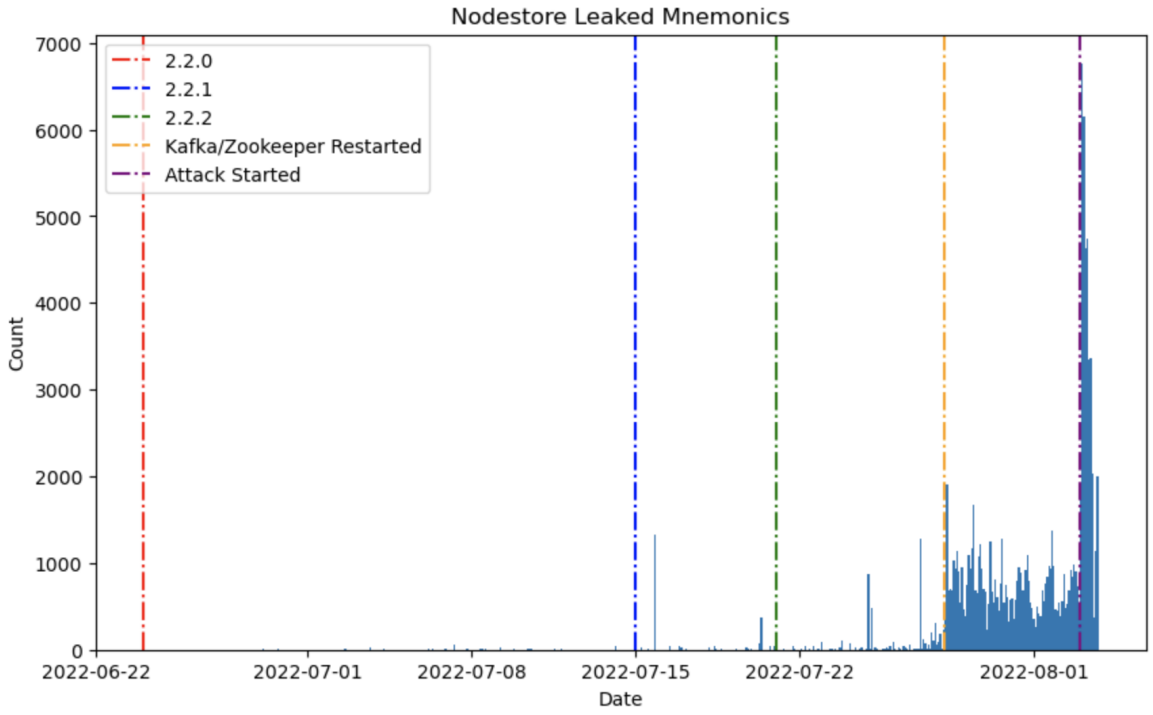

Figure 02.1: Slope Log Server Timeline

03 | Data Analysis

Nodestore Timestamp

The stored logs contain both a nodestore timestamp (when the log was stored in the database) and a sentry timestamp (when the log was generated). We observed large discrepancies between these two values.

External nodestore timestamps have a hard cutoff on July 28th (when the Kafka container was restarted):

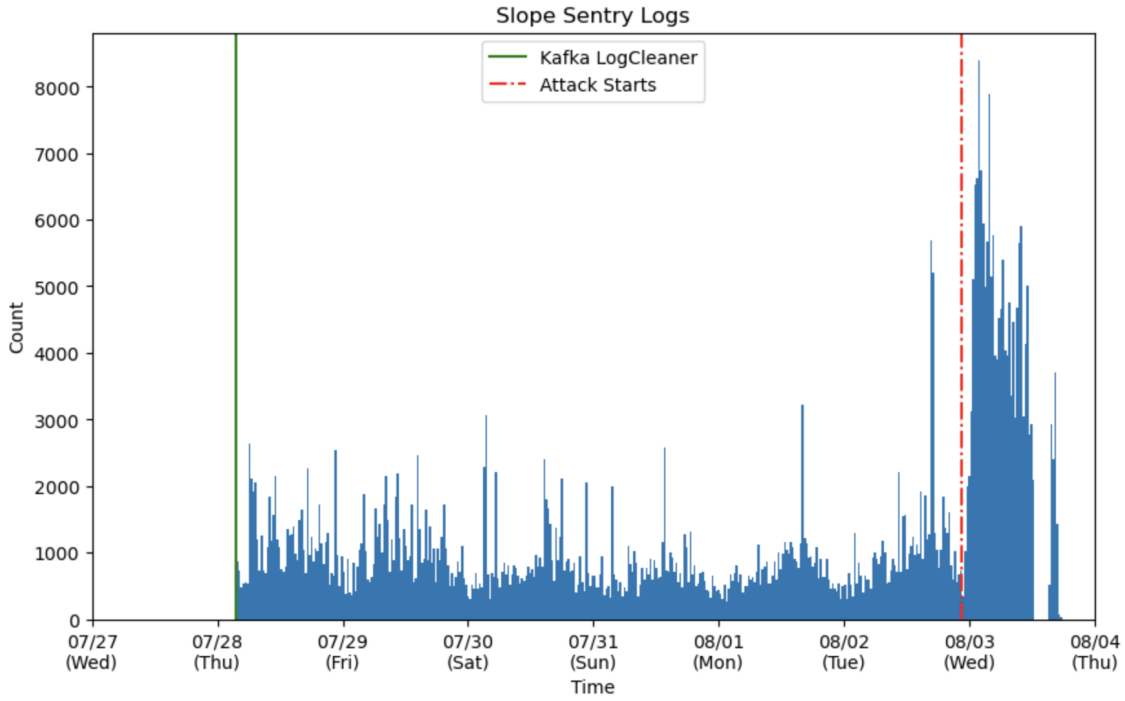

Figure 03.1: Nodestore Timestamps

Internal sentry timestamps are back-dated up to a month and have a soft cutoff on July 28th:

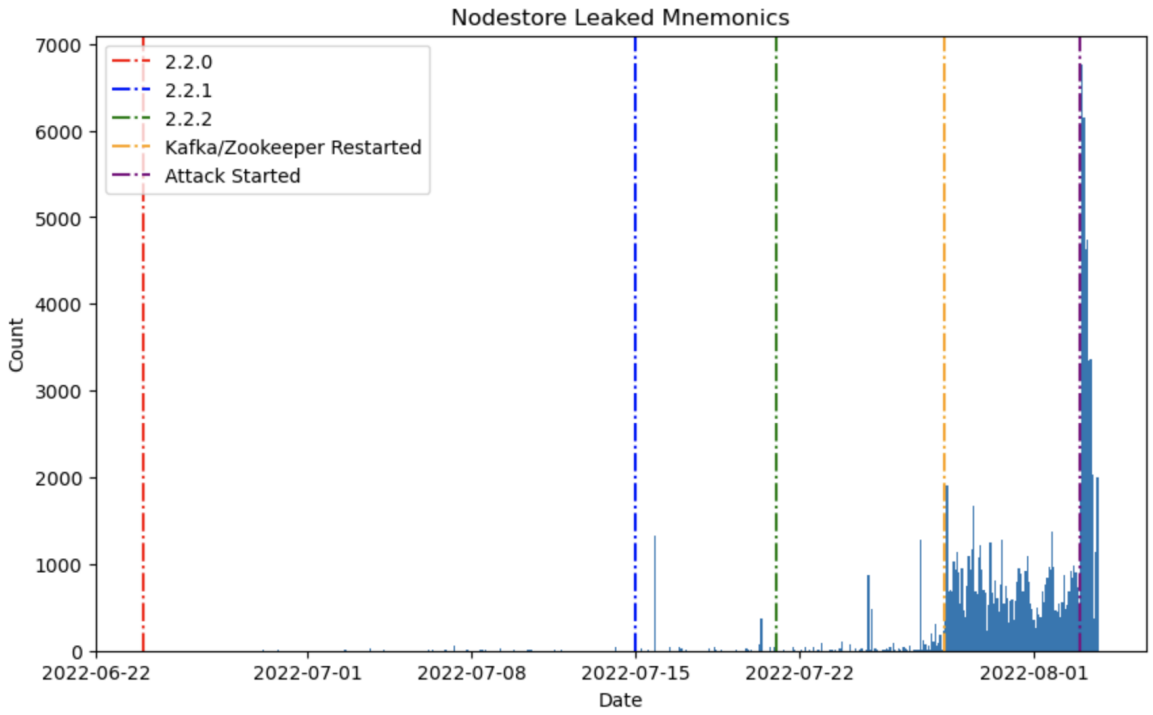

Figure 03.2: Sentry Timestamps

We analyzed the difference between the two timestamps. Most timestamps roughly correlate 1:1 (as shown by the diagonal line on the right). However, internal sentry timestamps may lag nodestore timestamps by up to one month.

Additionally, we do not observe any sudden influx of logs in the period after Kafka was restarted which suggests that logs generated prior to July 28th were not cached en masse, but rather most of them were dropped.

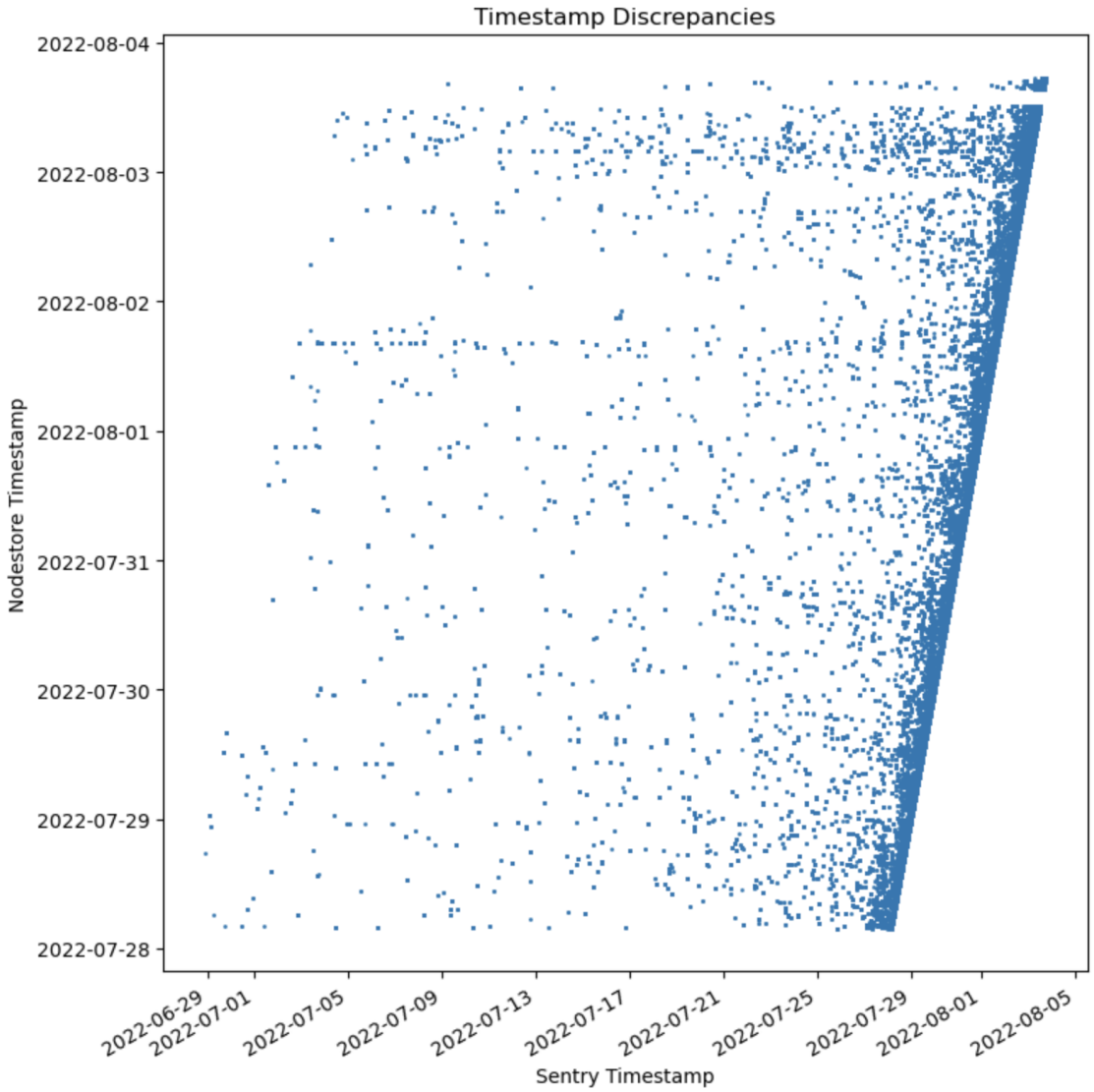

Figure 03.3: Timestamp Differential

Docker Containers

We identified existing metadata for 6 docker containers:

1. `75a83b0d0f029b72f715bc41c69da98b58b76d2877a60387b8de55f98976f54c`: Running zookeeper, started `2022-07-28T03:38:21.643529317Z`.
2. `815d897a90f7b715e1d551c289ba2224eda310b564f8cef07221aacee6457502`: Running ClickHouse, started `2022-01-19T07:12:30.903547637Z`.
3. `893aa1ac8f11a7dc851081952314621ba9ded3d1b6ef86682e93927039401ba6`: Running postgres, started `2022-01-19T07:11:12.323572135Z`.

4. `adaf8f9dfe71f3567295b6d35d52cf88ba8eea0512faa6a818b772920ac67b02`: Running a container unpacking a solana validator state, started on May 3rd.
5. `bb4380ae47dd7ebec1171d155f6e654bb4581cbf281fb5b7a604af290ed51a79`: Running Kafka, started `2022-07-28T03:38:52.400743414Z`.
6. `effae60a37bea1f05bf212d0b0c473a87c191bc24d94fb0c3f27bf81b6fbf85e`: Running redis, started `2022-01-19T07:11:04.027390996Z`

Several of these containers wrote logs to a persistant log file. Using the timestamps associated with each line in the log file, we were able to generate an estimate of service activity:

Postgres

Postgres was started on January 19th and restarted four times during a maintenance period on April 23rd.

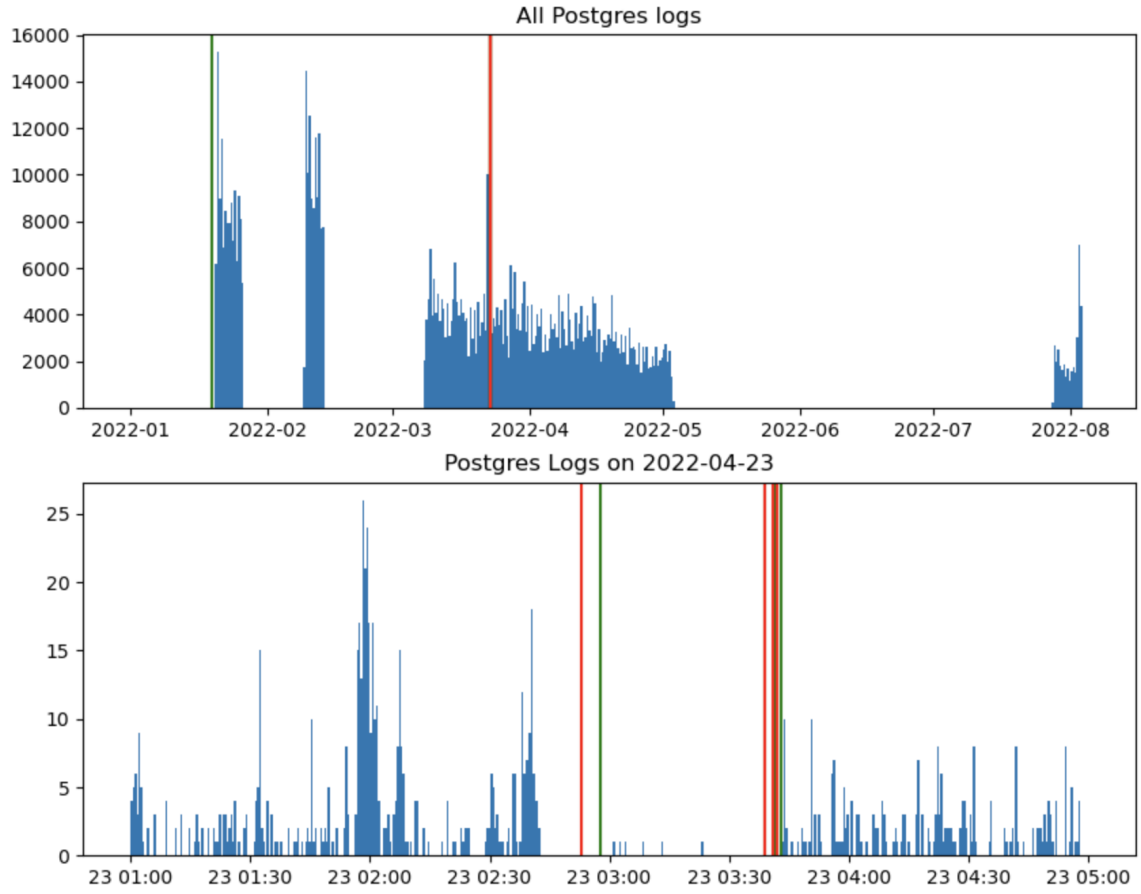

Figure 03.4: Postgres Activity

Clickhouse

Clickhouse was started on January 19th and restarted twice on April 23rd.

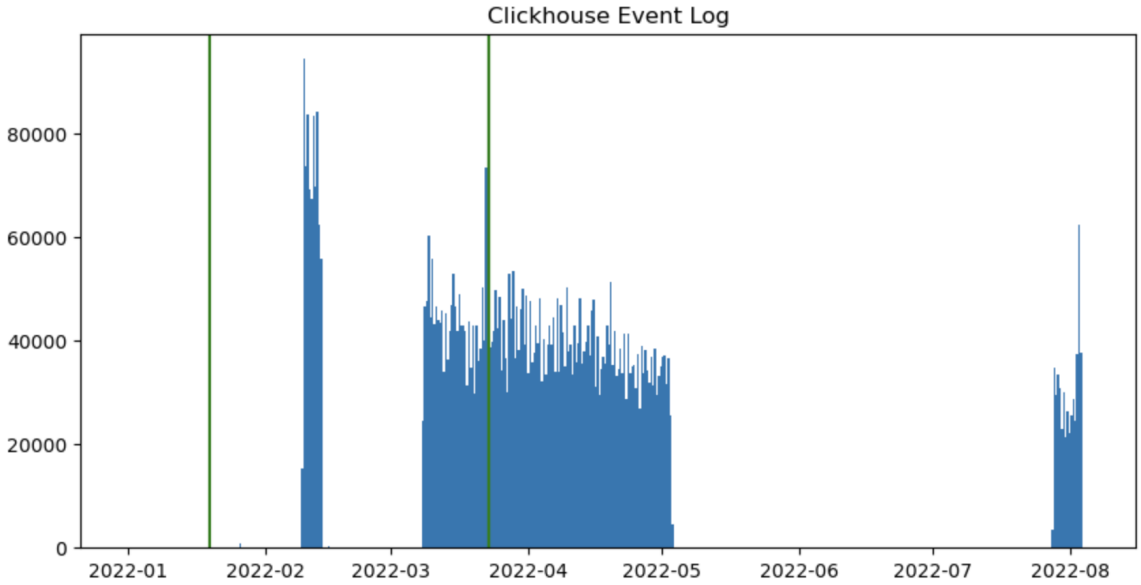

Figure 03.5: Clickhouse Activity

Redis

Redis was started on January 19th and restarted twice on April 23rd. Redis logs also contain a frequent pulse message showing that the service was up throughout the entire period from January to August.

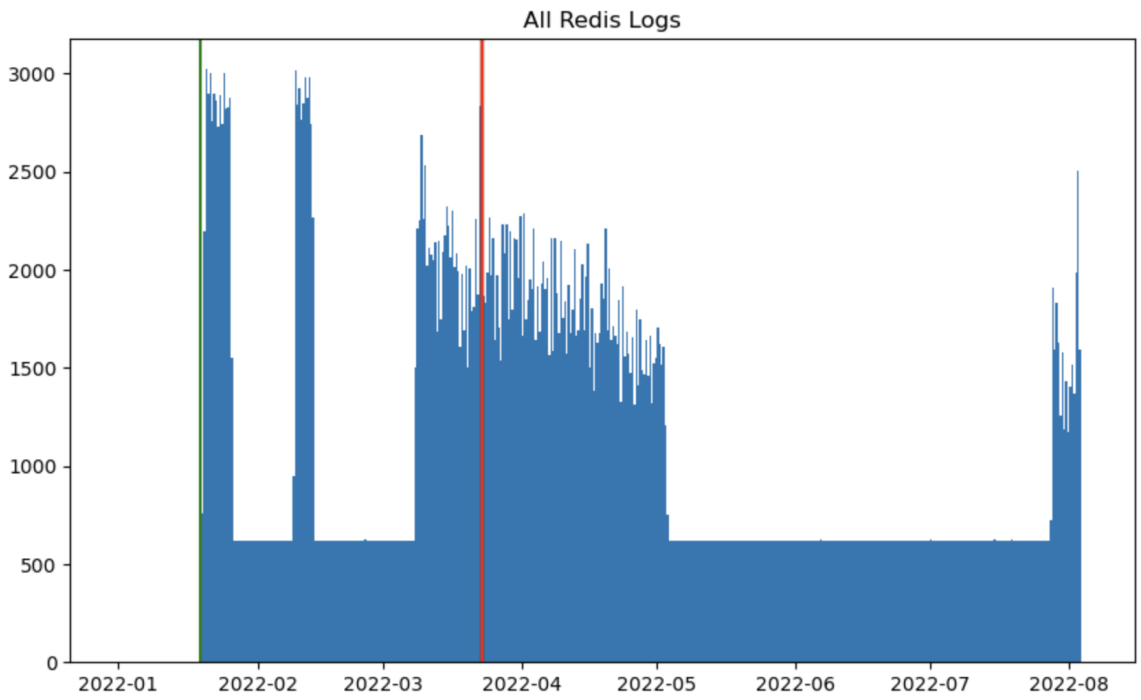

Figure 03.6: Redis Activity

Leak Source

While Slope Wallet code did attempt to filter out sensitive information in logs, some sensitive information was able to escape unfiltered.

Specifically, while the `message` field was filtered (generated in console events), sensitive information included in navigation events would not be filtered.

JSON

```
{
  "category": "navigation",
  "data": {
    "state": "didReplace",
    "to": "[Filtered]",
    "to_arguments": {
      "pageData": "Instance of 'ivb'",
      "type": "[Filtered]",
      "wallet": "WalletEntity{privateKey: 5Vd..."
    }
  },
  "level": "info",
  "timestamp": 1658990294.618,
```

```
  "type": "navigation"
},
{
  "category": "console",
  "level": "debug",
  "message": "[WalletEntity{privateKey: [****], address: ByZ..."d
  "timestamp": 1658990299.523,
  "type": "debug"
},
```