ChainLight

initia

Initia Security Audit

: opinit-bots security audit

Oct 16, 2024

Revision 1.0

ChainLight@Theori

Table of Contents

Executive Summary

Starting on September 3, 2024, ChainLight of Theori audited the Executor and Challenger components of the in a state of ongoing development opinit-bot. These functions are critical parts of the OPinit Stack that relay transactions and data between opHost (L1) and opChild (L2). The bot handles tasks such as processing L1 token deposit transactions to L2, submitting L2 output roots to L1, relaying oracle data, submitting block data to data availability layers, and providing withdrawal proofs to users.

ChainLight has designed the Threat scenarios. Among these, some of the main threat scenarios are as follows.

- Is the message encoding and decoding process in opinit-bot (Challenger, Executor) functioning correctly? (INITIA-OPINIT-BOT-001)
- Can the system effectively handle bottlenecks when sending and receiving large volumes of messages? Is it resilient against spam attacks? (INITIA-OPINIT-BOT-003)
- Does the system recover from updates, initialization, or physical environment failures? (INITIA-OPINIT-BOT-002)
- What potential issues could arise if opinit-bot is executed on remote services by users other than the designated Minita operator?
- Censorship of the Minita Challenger functionality in opinit-bot operations. (INITIA-OPINIT-BOT-004)

During the audit, ChainLight discovered a critical issue arising from inaccurate message processing that handles all transactions in a block regardless of their execution results, leading to potential inconsistencies. The system delaying issues occurring during node sync state were also identified. Additionally, improvements for design flaws were found in the message processing system causing performance bottlenecks, and security risks were noted in the challenge process.

Audit Overview

Scope

Name	Initia Security Audit
Target / Version	• Git Repository (opinit-bots): commit `108d0f8165ea0195d086cdbb443cdc5715cb0413`
Application Type	Layer2 Bots (Executor, Challenger)
Lang. / Platforms	Bot System [Go]

Code Revision

N/A

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.
Informational	Any informational findings that do not directly impact the user or the protocol.
Note	Neutral information about the target that is not directly related to the project's safety and security.

Status Categories

Status	Description
Reported	ChainLight reported the issue to the client.
WIP	The client is working on the patch.
Patched	The client fully resolved the issue by patching the root cause.
Mitigated	The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations.
Acknowledged	The client acknowledged the potential risk, but they will resolve it later.
Won't Fix	The client acknowledged the potential risk, but they decided to accept the risk.

Finding Breakdown by Severity

Category	Count	Findings
Critical	1	• `INITIA-OPINIT-BOT-001`
High	1	• `INITIA-OPINIT-BOT-004`
Medium	1	• `INITIA-OPINIT-BOT-002`
Low	0	• N/A
Informational	2	• `INITIA-OPINIT-BOT-003` • `INITIA-OPINIT-BOT-005`
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	INITIA-OPINIT-BOT-001	`txHandler()` must not handle a failed transaction	Critical	Patched
2	INITIA-OPINIT-BOT-002	Some messages may be delayed if the deposit events are not sent more than 10 times	Medium	Patched
3	INITIA-OPINIT-BOT-003	Potential degradation in message processing designs	Informational	Mitigated
4	INITIA-OPINIT-BOT-004	Design issues related to the challenge process	High	Acknowledged
5	INITIA-OPINIT-BOT-005	Minor Suggestions	Informational	Patched

#1 `INITIA-OPINIT-BOT-001` `txHandler()` must not handle a failed transaction

ID	Summary	Severity
`INITIA-OPINIT-BOT-001`	The `txHandler()` function processes all transactions in a block without verifying their execution results. This leads to conflicts by handling failed transaction data and processing inaccurate data.	**Critical**

Description

In the `blockProcessLooper` function of `/node/process.go`, the latest `blockResult` values are retrieved through `fetchNewBlock()` based on the current height. The block data is then processed via `process.handleNewBlock()` using `txHandler()` implemented in each child or host.

However, the `txHandler()` function does not check whether the transaction's message result was successful.

The inadequate checks are as follows:

1. The `txHandler()` function in `challenger/host/handler.go` only verifies `args.TxIndex == 0`. This ensures that only the first transaction is reflected but does not handle failed transactions. Consequently, if a failed transaction is in the first of the block, invalidated values are enqueued in `h.eventQueue` within `oracleTxHandler()`.
2. Similarly, the `txHandler()` function in `challenger/executor/handler.go` does not check for failed transactions. If a failed transaction is processed, incorrect `MsgUpdateOracle` message values are included in the `ProcessedMsgs` queue.
3. In `challenger/child/handler.go`, the `txHandler()` function only performs signature checks through `authsigning`. Therefore, if a transaction included in the block has a valid signature, it can pass the check regardless of whether the message failed. It then checks whether the message is of a specific type via `msgs[0].(*opchildtypes.MsgUpdateOracle)`. As a result, incorrect values are assigned to `ch.eventQueue` and `ch.lastUpdatedOracleL1Height` in `oracleTxHandler()`.

Impact

Critical

An attacker can broadcast transactions by creating fake events of the `MsgUpdateOracle` type, constructing messages as demonstrated:

```
const lcd = new LCDClient(
  "https://maze-rest-sequencer-beab9b6f-d96d-435e-9caf-5679296d8172.ue1-prod.newmetric.xyz/",
  {
    chainId: "landlord-1",
    gasPrices: "0.16l2/afaa3f4e1717c75712f8e8073e41f051a4e516cd25daa82d948c4729388edefd",
    gasAdjustment: "2.0",
  }
);
const wallet = new Wallet(lcd, rawKey);
msgs = [new MsgUpdateOracle(rawKey.accAddress, 1337, "data1234CL")];
async function main() {
  const signedTx = await wallet.createAndSignTx({ msgs });
  lcd.tx.broadcastSync(signedTx).then((res) => console.log(res));
}
main();
```

This allows the attacker to enqueue numerous failed messages into the `eventQueue`, causing delays or conflicts in Oracle updates between the child and host. During the challenge event process, when querying pending events up to the last Oracle event's L1 height via `getOraclePendingEvents()`, the presence of many failed messages can lead to processing delays due to linear handling.

Recommendation

Modify the `txHandler()` function to include the condition `if args.TxResult.code != 0 { return nil }`, ensuring that failed transactions are not processed.

Perform checks not only on the message type but also on the properties within the message.

Remediation

Patched

The issue has been resolved as recommended.

#2 `INITIA-OPINIT-BOT-002` Some messages may be delayed if the deposit events are not sent more than 10 times

ID	Summary	Severity
`INITIA-OPINIT-BOT-002`	Broadcasting of `processedMsgs` to the child is delayed during node and data synchronization when the `msgQueue` lacks sufficient messages.	Medium

Description

In the `endBlockHandler()` function of `executor/host/handler.go`, the condition `blockHeight != args.LatestHeight && len(msgQueue) > 0 && len(msgQueue) <= 10` only passes when the `msgQueue` contains more than 10 items. However, there is no guarantee that the `initiate_token_deposit` event handling will occur more than 10 times during node and data synchronization. Consequently, if the `msgQueue` is insufficient, this condition fails, delaying `endBlockHandler()` from broadcasting the oracle update messages in `processedMsgs` to the child.

Impact

Medium

Delays occur in broadcasting Oracle updates to the child until deposit transactions exceed 10, even after reaching the latest block height.

Recommendation

The `msgQueue` for Oracle update messages should be updated in the `txHandler()` function of `executor/host/handler.go`.

Remediation

Patched

By removing `blockHeight != args.LatestHeight && len(msgQueue) > 0 && len(msgQueue) <= 10`, the system has been modified to broadcast even during node synchronization.

#3 `INITIA-OPINIT-BOT-003` Potential degradation in message processing designs

ID	Summary	Severity
`INITIA-OPINIT-BOT-003`	Design flaws in the message processing system may lead to performance bottlenecks and system overload due to unbuffered channels and fixed retry intervals.	Informational

Description

Several design issues have been identified in the message processing system:

1. **Unbuffered channels causing performance issues with high message volumes:**

   a. In the `Start()` function of `node/broadcaster/process.go`, the `txChannel` is unbuffered and can process only one message at a time.

   b. Unbuffered channels block the sender when no receiver of the channel is available, leading to performance issues when a large number of messages occur rapidly.

   c. As the number of Minita nodes increases, the number of OPInit bots required to handle events also increases proportionally, potentially causing bottlenecks in event processing.

2. **System overload due to fixed retry intervals:**

   a. If message processing in `txChannel` fails, the system retries up to 10 times with a fixed 30-second interval between attempts.

   b. Retrying failed message processing at fixed intervals can result in concentrated load if transaction failures persist, causing bottlenecks and degrading overall system performance.

Impact

Informational

The system may experience performance degradation and bottlenecks as it scales, affecting its ability to process events efficiently. Persistent transaction failures could lead to system overload and

reduced performance.

Recommendation

1. **Use buffered channels:**

   a. Introduce buffering in the `txChannel` to allow the sender to store a certain amount of data even if the receiver does not immediately process it. This approach enables the sender to transmit data while the receiver processes messages later, reducing bottlenecks and improving performance.

   b. A fallback system should be designed to address potential data loss if the node service fails and the buffer is cleared. Implementing a recovery mechanism to safely store and reprocess data that fails to be received is recommended.

2. **Apply an exponential backoff algorithm:**

   a. Implement an exponential backoff strategy where the retry interval increases progressively upon each failure. This reduces system load during persistent failures. If the number of retries reaches a certain threshold, the operation should be canceled to prevent unnecessary burden on the system.

Remediation

Mitigated

- The client responded that they are currently using unbuffered channels due to difficulties in sequence management.
- A backoff algorithm strategy has been added as recommended.

#4 `INITIA-OPINIT-BOT-004` Design issues related to the challenge process

ID	Summary	Severity
`INITIA-OPINIT-BOT-004`	Design flaws in the challenge process may lead to risks such as malicious behavior by permissioned challengers, insufficient recovery procedures, and potential censorship by the Minita operator.	**High**

Description

Several design issues have been identified in the current challenge process:

1. **Malicious behavior by permissioned challengers:**

   a. The system operates with a permissioned challenge mechanism controlled by a single entity, similar to the OPStack Bedrock version.

   b. If the permissioned challenger acts maliciously, the resolution relies on governance voting to remove or replace them, which may not be swift enough to prevent system disruptions.

2. **Reduced user experience due to insufficient recovery procedures when challenges against incorrect proposers fail:**

   a. While data can be recovered through governance, this process can extend the total waiting time to 14 days, negatively impacting the user experience.

3. **Risk of censorship by the Minita operator:**

   a. The Minita operator has structural control over the challenger system, posing risks of direct data manipulation and censorship.

   b. Because the voters of the governance system rely on the challenger's http responses.

There is a possibility that the Minita team could exploit this centralized control to manipulate or censor data.

Impact

High

Malicious actions by permissioned challengers and potential censorship by the Minita operator could compromise system integrity, cause significant delays, manipulate data, and erode user trust.

Recommendation

1. **Delegate the permissioned challenger role to the L1 validator group:**

   a. To prevent self-challenging by the Minita operator and reduce centralized control, the permissioned challenger role should be performed by the L1 validator group.

2. **Establish oversight mechanisms on the L1 side:**

   a. Implement a review process to monitor the Minita L2 Challenger Group for collusion or malicious activities, ensuring transparency and accountability.

Remediation

Acknowledged

The client responded that they would consider this issue during opinit development.

#5 `INITIA-OPINIT-BOT-005` Minor Suggestions

ID	Summary	Severity
`INITIA-OPINIT-BOT-005`	The description includes multiple suggestions for preventing incorrect settings caused by operational mistakes, mitigating potential issues, improving code maturity and readability, and other minor issues.	Informational

Description

Operational Risk Mitigation / Sanity Check

- In the `initCmd()` function of `/cmd/opinitd/init.go`, if the botType argument is not set or does not match the specified values, a panic occurs. As a result, an error code is displayed to the user. Therefore, verifying that a predefined value has been entered is recommended.
- In the `Initialize()` function of `executor/batch/batch.go`, when creating the `batch` file with `bs.batchFile, err = os.OpenFile(bs.homePath+"/batch", fileFlag, 0666)`, permissions are set to `0666`. This grants write permissions to 'Other' users, which can pose a security risk. Therefore, it is appropriate to change the file creation permissions to `0660` so that, after applying `umask`, the final permissions become `0640`. This prevents unnecessary permission grants.
- The event parsing logic should be enhanced by adding checks to ensure that all required attributes are present, and any that are missing should be logged for easier debugging. Although it is unlikely that a transaction could cause attributes to be missing, this defense-in-depth approach will improve the robustness of the system. These changes will help maintain the stability and reliability of the bot, even in unexpected situations.
  - i.e., https://github.com/initia-labs/opinit-bots/blob/86916b3de6f6362e245c156d9a20971125ee834e/challenger/child/withdraw.go#L31

Impact

Informational

Recommendation

Consider applying the suggestions in the description above.

Remediation

Patched

The issue has been resolved as recommended.

Revision History

Version	Date	Description
1.0	Oct 16, 2024	Initial version

Theori, Inc. (“We”) is acting solely for the client and is not responsible to any other party. Deliverables are valid for and should be used solely in connection with the purpose for which they were prepared as set out in our engagement agreement. You should not refer to or use our name or advice for any other purpose. The information (where appropriate) has not been verified. No representation or warranty is given as to accuracy, completeness or correctness of information in the Deliverables, any document, or any other information made available. Deliverables are for the internal use of the client and may not be used or relied upon by any person or entity other than the client. Deliverables are confidential and are not to be provided, without our authorization (preferably written), to entities or representatives of entities (including employees) that are not the client, including affiliates or representatives of affiliates of the client.