Smart Contract Security Audit Report

Asterizm

1. Contents

2. General Information

This report contains information about the results of the security audit of the Asterizm (hereafter referred to as "Customer") smart contracts, conducted by Decurity in the period from 18/11/2024 to 20/12/2024.

2.1. Introduction

Tasks solved during the work are:

- Review the protocol design and the usage of 3rd party dependencies,
- Audit the contracts implementation,
- Develop the recommendations and suggestions to improve the security of the contracts.

2.2. Scope of Work

The audit scope included the contracts in the following repository: asterizm-contracts-sol (commit 6b2ff7). Review was done for commit 55509b.

The following contracts have been tested:

- programs/asterizm-token-example/src/lib.rs
- programs/asterizm-client/src/models/message.rs
- programs/asterizm-client/src/lib.rs
- programs/asterizm-relayer/src/lib.rs
- programs/asterizm-initializer/src/models/message.rs
- programs/asterizm-relayer/src/models/message.rs
- programs/asterizm-initializer/src/lib.rs
- programs/asterizm-client/src/models/client.rs
- programs/asterizm-relayer/src/models/settings.rs
- programs/asterizm-client/src/models/client_trusted_addresses.rs
- programs/asterizm-client/src/models/client_senders.rs
- programs/asterizm-relayer/src/models/chain.rs

- programs/asterizm-relayer/src/models/custom_relayer.rs
- programs/asterizm-initializer/src/models/settings.rs
- programs/asterizm-client/src/models/settings.rs
- programs/asterizm-initializer/src/models/blocked_account.rs
- programs/asterizm-client/src/models/mod.rs
- programs/asterizm-relayer/src/models/mod.rs
- programs/asterizm-initializer/src/models/mod.rs

2.3. Threat Model

The assessment presumes actions of an intruder who might have capabilities of any role (an external user, token owner, token service owner, a contract). The centralization risks have not been considered upon the request of the Customer.

The main possible threat actors are:

- Relayer
- Client Sender
- Protocol Owner
- User

2.4. Weakness Scoring

An expert evaluation scores the findings in this report, an impact of each vulnerability is calculated based on its ease of exploitation (based on the industry practice and our experience) and severity (for the considered threats).

2.5. Disclaimer

Due to the intrinsic nature of the software and vulnerabilities and the changing threat landscape, it cannot be generally guaranteed that a certain security property of a program holds.

Therefore, this report is provided “as is” and is not a guarantee that the analyzed system does not contain any other security weaknesses or vulnerabilities. Furthermore, this report is not an endorsement of the Customer’s project, nor is it an investment advice.

That being said, Decurity exercises best effort to perform their contractual obligations and follow the industry methodologies to discover as many weaknesses as possible and maximize the audit coverage using the limited resources.

3. Summary

During audit we have detected one high and multiple medium issues. The Asterizm team has given the feedback for the suggested changes and explanation for the underlying code.

3.1. Suggestions

The table below contains the discovered issues, their risk level, and their status as of October 10, 2024.

Table. Discovered weaknesses

Issue	Contract	Risk Level	Status
Anyone can block an account in initializer	asterizm-initializer/src/models/message.rs	**High**	Fixed
DOS via pre-funding of transfer account	programs/asterizm-client/src/lib.rs	**High**	Fixed
The unblock functionality doesn’t work	asterizm-initializer/src/lib.rs	**Medium**	Fixed
Refunds can be processed after execution	programs/asterizm-client/src/lib.rs	**Medium**	Fixed
Incorrect access control of settings	programs/asterizm-client/src/models/settings.rs	**Medium**	Fixed

Issue	Contract	Risk Level	Status
	programs/asterizm-initializer/src/models/settings.rs programs/asterizm-relayer/src/models/settings.rs		
Anyone can request a refund for any transaction	programs/asterizm-client/src/lib.rs	Medium	Fixed
No unblock functionality	programs/asterizm-initializer/src/lib.rs	Medium	Fixed
Client Creation Frontrunning	programs/asterizm-client/src/lib.rs	Low	Fixed
Initialization Frontrunning		Info	Fixed

4. General Recommendations

This section contains general recommendations on how to improve overall security level.

The Findings section contains technical recommendations for each discovered issue.

4.1. Security Process Improvement

The following is a brief long-term action plan to mitigate further weaknesses and bring the product security to a higher level:

- Keep the whitepaper and documentation updated to make it consistent with the implementation and the intended use cases of the system,
- Perform regular audits for all the new contracts and updates,
- Ensure the secure off-chain storage and processing of the credentials (e.g. the privileged private keys),
- Launch a public bug bounty campaign for the contracts.

5. Findings

5.1. Anyone can block an account in initializer

Risk Level: High

Status: Fixed in commit 0deefa9f144e598bb304fd1a77a5a42f40988562

Contracts:

asterizm-initializer/src/models/message.rs

Location: Lines: `55, 62, 182, 189`.

Description:

In the `asterizm-initializer` program, when an account is being blocked, the corresponding data account is initialized via the `block_account()` function and deposited with required amount of lamports.

Structures `InitSendMessage` and `InitTransferMessage` use constraints `blocked_src_account.lamports() == 0` and `blocked_dst_account.lamports() == 0` to ensure that the source or destination account is not blocked when invoking the `send_message()` and `init_transfer()` functions. These constraints are unsafe, since an attacker can compute the addresses of `blocked_src_account` and `blocked_dst_account` and deposit a minimal amount of lamports into them, even if these accounts have not been initialized. This can lead to a DoS attack by preventing message transfers from and to blocked accounts.

Remediation:

Check the `is_blocked` field in the blocked data account instead of its balance to determined, whether an address is blocked or not.

5.2. DOS via pre-funding of transfer account

Risk Level: High

Status: Fixed in commit 9b467a0728b727c065608822adbcece91f44f0ac

Contracts:

programs/asterizm-client/src/lib.rs

Location: Lines: `355, 433`. Function: `confirm_incoming_refund, init_receive_message`.

Description:

In case there are any lamports in the transfer account, the `asterizm-client` program assumes that the transfer account already exists and tries to modify its state. However, a malicious user may pre-calculate the address of the transfer account and fund it, e.g., with 1 lamport, thus resulting in a failure when trying to modify account data. This will allow an attacker to perform a DOS attack, which will result in a failure of processing every incoming message and refund confirmation.

Remediation:

Consider checking that there are enough lamports for rent on account. Example reference logic can be found in spl-token

5.3. The unblock functionality doesn't work

Risk Level: Medium

Status: Fixed in commit 55509bb74d544c2ac84a71feecca59b0404d979a.

Contracts:

asterizm-initializer/src/lib.rs

Location: Lines: `66-81`. Function: `unblock_account()`.

Description:

The accounts unblock functionality was added in the `unblock_account()` function of the `asterizm-initalizer` program. However, it doesn't work, since the transfer CPI context isn't signed. Additionally, in current logic, the account data isn't deleted after transfer.

Remediation:

Close the account in `unblock_account()` function to delete account data and transfer remaining lamports properly.

5.4. Refunds can be processed after execution

Risk Level: Medium

Status: Fixed in commit 9b467a0728b727c065608822adbcece91f44f0ac

Contracts:

programs/asterizm-client/src/lib.rs

Location: Lines: 312. Function: `process_refund_request`.

Description:

The `process_refund_request` function checks that `transfer_account` is not refunded yet, but it doesn't check whether it has already been executed or not, thus allowing a refund to be processed after execution.

Remediation:

Since there is a `transfer_sending_result` that alerts about successful message processing, we would recommend adding a flag to the transfer account that will be set as true in this function and will not allow processing a refund request after that.

5.5. Incorrect access control of settings

Risk Level: Medium

Status: Fixed in commit 9b467a0728b727c065608822adbcece91f44f0ac

Contracts:

programs/asterizm-client/src/models/settings.rs

programs/asterizm-initializer/src/models/settings.rs

programs/asterizm-relayer/src/models/settings.rs

Description:

`UpdateSettings` structure has the following constraint:

```
#[account(
constraint = program_data.upgrade_authority_address == Some(authority.key())
|| program_data.upgrade_authority_address == Some(SystemProgramId)
|| authority.key() == settings_account.manager
)]
pub program_data: Account<'info, ProgramData>,
```

In case `upgrade_authority_address` is set to `SystemProgramId` , anyone will be able to update settings of asterizm-client, asterizm-initializer and asterizm-relayer. This was probably added for testing purpose, because of

```
[test]
upgradeable = false
```

in `Anchor.toml` that sets `upgrade_authority_address` to `SystemProgramId` by default. However, in case this constraint is used in production, it may become potentially dangerous in case `upgrade_authority_address` will be mistakenly set to `SystemProgramId`.

Remediation:

Consider removing this constraint for production.

5.6. Anyone can request a refund for any transaction

Risk Level: Medium

Status: Fixed in commit 9b467a0728b727c065608822adbcece91f44f0ac

Contracts:

programs/asterizm-client/src/lib.rs

Location: Lines: 298. Function: `add_refund_request`.

Description:

The `add_refund_request` function in the `Asterizm-client` program allows any user to create refund requests without proper authorization checks.

This could lead to unauthorized refund requests for any valid transfer and potential fund loss if there is another bugs in offchain scope.

Remediation:

Consider adding constrains to `AddRefundRequest` struct to ensure that signer is

The original user address

The client account owner

to prevent unauthorized refund request creation.

5.7. No unblock functionality

Risk Level: Medium

Status: Fixed in commit 9b467a0728b727c065608822adbcece91f44f0ac

Contracts:

programs/asterizm-initializer/src/lib.rs

Description:

The `asterizm-initializer` program lacks the ability to unblock accounts once they have been blocked. It could lead to DoS for legitimate users if they were blocked by mistake.

Remediation:

Consider adding unblock functionality.

5.8. Client Creation Frontrunning

Risk Level: Low

Status: Fixed in commit 9b467a0728b727c065608822adbcece91f44f0ac

Contracts:

programs/asterizm-client/src/lib.rs

Location: Lines: 51. Function: `create_client`.

Description:

The `create_client()` function lack the caller validation and can be frontrunned by anyone with malicious parameters, like different relayer address and disabled hash checks.

Remediation:

Consider adding constrain `authority.key() == user_address` to prevent frontrunning.

5.9. Initialization Frontrunning

Risk Level: Info

Status: Fixed in commit 9b467a0728b727c065608822adbcece91f44f0ac

Location: Function: `initialize()`.

Description:

The `initialize()` function in a `Asterizm-Client`, `Asterizm-Initializer`, `Asterizm-Relayer` is vulnerable to frontrunning attacks, since there is no caller validation. An attacker could watch the mempool and initialize the program with his own settings.

Remediation:

Consider hardcoding the authority to ensure that only the designated authority can perform initialization.

6. Appendix

6.1. About us

The Decurity team consists of experienced hackers who have been doing application security assessments and penetration testing for over a decade.

During the recent years, we've gained expertise in the blockchain field and have conducted numerous audits for both centralized and decentralized projects: exchanges, protocols, and blockchain nodes.

Our efforts have helped to protect hundreds of millions of dollars and make web3 a safer place.