CoinFabrik

Security Audit Report

XLink - Staking

November 2024

V202410

Executive Summary

CoinFabrik was asked to audit some contracts for the XLink project.

During this audit we found one critical issue, one high issue, one medium issue and one minor issue. Also, several enhancements were proposed. The critical, high and minor issues were resolved. The medium issue was acknowledged.

After the audit concluded, a new high issue was found by Coinfabrik's QA team. This new issue was also resolved.

Scope

The audited files are from the git repository located at https://github.com/xlink-network/xlink-dao. The audit is based on the commit `47cbf8db47579c8afc59d4f5a73ef53caa507d58`. Fixes were checked on commit `69c4974db88a87930a183cb53396b1a7f4b55cf5`. A new issue was found on commit `db5500de46f17e93e25e4941e5ccf4d5b3544987`. Its fix was checked on commit `67848619b86be302a18d8f9b4dfe5526e46864cb`.

The scope for this audit includes and is limited to the following files:

- `contracts/aux/xlink-staking.clar`: contract that tracks the staking of several tokens.
- `contracts/liabtc/liabtc-mint-endpoint.clar`: `contracts/aux/xlink-staking.clar` façade handling the lifetime of the staking and burning for the LiaBTC token.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit. In particular, it must be noted that neither `contracts/liabtc/liabtc-mint-registry.clar` nor `contracts/liabtc/token-liabtc.clar` were audited even though they are dependencies of the audited contracts.

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

Each severity label is detailed in the Severity Classification section. Additionally, the statuses are explained in the Issues Status section.

Id	Title	Severity	Status
CR-01	Steal Tokens on finalize-burn	🟥 Critical	**Resolved**
HI-01	Authentication with tx-sender on revoke-burn	🟧 High	**Resolved**
HI-02	Single Validator Can Add Rewards	🟧 High	**Resolved**
ME-01	Lack of Admin Separation	🟨 Medium	**Acknowledged**
MI-01	Authentication with tx-sender on mint	🟨 Minor	**Resolved**

Critical Severity Issues

CR-01 Steal Tokens on finalize-burn

Location

- `contracts/liabtc/liabtc-mint-endpoint.clar:103`

Classification

- CWE-285: Improper Authorization[1]

Description

Any account may invoke the `finalize-burn` function stealing the abtc tokens, as any one can invoke this function and the abtc tokens are sent to the `tx-sender` in this line

```
(as-contract (try! (contract-call? .liabtc-mint-registry transfer (get amount
request-details) sender 'SP2XD7417HGPRTREMKF748VNEQPDRR0RMANB7X1NK.token-abtc)))
```

where `sender` is the `tx-sender` as set in the let-binding in line 99.

Recommendation

Send the tokens to the requester of the burn instead

[1] https://cwe.mitre.org/data/definitions/285.html

Status

Resolved. Fixed according to the recommendation.

High Severity Issues

HI-01 Authentication with tx-sender on revoke-burn

Location

- `contracts/liabtc/liabtc-mint-endpoint.clar:92`

Classification

- CWE-303: Incorrect Implementation of Authentication Algorithm[2]

Description

Using `tx-sender` to authenticate may lead to phishing attacks and exposure to other contract's bugs[3]. This is especially impactful when no tokens are transferred, as the Stacks' postconditions mechanism does not apply. Post-conditions also do not make any difference when the `tx-sender` is not the EOA that initiated the transaction.

Recommendation

Use `contract-caller` to authenticate instead. Given that the account compared against is stored in the `requested-by` field of the `request-details`, the `contract-caller` should also be stored there instead of the `tx-sender`.

Status

Resolved. `contract-caller` is used unless the `tx-sender` is the dao or an extension, which are expected to properly handle being used as `tx-sender` in authorization. See `contracts/liabtc/liabtc-mint-endpoint.clar:54` in the fixes commit.

HI-02 Single Validator Can Add Rewards

Location

- `contracts/aux/xlink-staking.clar:183`

[2] https://cwe.mitre.org/data/definitions/303.html
[3] https://www.coinfabrik.com/blog/tx-sender-in-clarity-smart-contracts-is-not-adviced/.

Classification

- CWE-294: Authentication Bypass by Capture-replay[4]

Description

When validating the signature for the message in the `add-rewards` function it does not check that the signature packs are not repeated. This may be exploited by a single validator, repeating its signature multiple times to add the rewards even when no other validator approves.

This issue was found by CoinFabrik's QA team on commit `db5500de46f17e93e25e4941e5ccf4d5b3544987` after the audit concluded.

Recommendation

Check that the signature packs do not repeat.

Status

Resolved. The issue was resolved by checking that the `signature-packs` list is equal to the `signature-packs` list with the duplicated items removed. Checked on commit `67848619b86be302a18d8f9b4dfe5526e46864cb`.

Medium Severity Issues

ME-01 Lack of Admin Separation

Location

- `contracts/aux/xlink-staking.clar`

Classification

- CWE-863: Incorrect Authorization[5]

Description

The actions that an administrator should do (`set-required-validators`, `set-paused`, `set-block-threshold`, `set-accrued-rewards`, `set-approven-token`, `add-validator`, `remove-validator`, `withdraw`), can be executed by the same accounts that can normally operate

[4] https://cwe.mitre.org/data/definitions/294.html
[5] https://cwe.mitre.org/data/definitions/863.html

with the contract (actions `stake` and `unstake`). The check used for the two roles is implemented in the `is-dao-or-extension` function.

This gives the normal operations the option to take actions that would give its executors more profit than just staking and unstaking. Some examples are:

1. Changing all the validators and manipulating the different tokens minting.
2. Calling `withdraw` instead of `unstake`, keeping the registered stake while retrieving the funds.
3. Unpausing a paused contract, so attackers may circumvent the pausing mechanism if they take control of a single extension.

Recommendation

Separate administrative actions to a different domain than the usual operations.

Status

Acknowledged. The development team informed us that both governance and privileged functions (and governance functions are a kind of privileged functions) are accessed by either dao itself or its extensions (which are addresses "trusted" by dao). So they say that this is by design.

Minor Severity Issues

MI-01 Authentication with tx-sender on mint

Location

- `contracts/liabtc/liabtc-mint-endpoint.clar:59`

Classification

- CWE-303: Incorrect Implementation of Authentication Algorithm[6]

Description

Using `tx-sender` to authenticate may lead to phishing attacks and exposure to other contract's bugs[7]. While postconditions mitigate this, it is still possible for a user of the contract to fall to this

[6] https://cwe.mitre.org/data/definitions/303.html
[7] https://www.coinfabrik.com/blog/tx-sender-in-clarity-smart-contracts-is-not-adviced/.

attack. For example, post-conditions also do not make any difference when the `tx-sender` is not the EOA that initiated the transaction.

Recommendation

Use `contract-caller` to authenticate instead.

Status

Resolved. As in HI-01, `contract-caller` is used unless the `tx-sender` is the dao or an extension, which are expected to properly handle being used as `tx-sender` in authorization.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

Id	Title	Status
EN-01	Automated Tests	Not implemented
EN-02	Token Burned on Request	Not implemented
EN-03	Non-Standard Transfer	Not implemented

EN-01 Automated Tests

Neither of the audited contracts has any automated tests. This makes it more difficult to reason about the code, as it is more complicated to make new tests checking for weird behavior. Also, the `pnpm test` command fails in the audited version.

Recommendation

1. All automated tests should pass on all the versions in the repository. In order to enforce this, add a continuous integration process to your development cycle.
2. All the smart-contract code in the repository should have proper test coverage including, at least, the happy path and checks to ensure that all the privileded roles actions are properly checked.

Status

Not implemented. This is acknowledged by the development team.

EN-02 Token Burned on Request

Location

- `contracts/liabtc/liabtc-mint-endpoint.clar:76`

Classification

- CWE-670: Always-Incorrect Control Flow Implementation[8]

Description

While the stated intent of the `request-burn` function is to request a burn at a later time, the action is actually taken when it is requested in line 76.

```
(as-contract (try! (contract-call? .xlink-staking unstake
'SP2XD7417HGPRTREMKF748VNEQPDRR0RMANB7X1NK.token-abtc amount message
signature-packs)))
```

It must be noted that this is the kind of issue that should be detected by just doing a proper happy path test.

Recommendation

All the unstaking logic should be in the `finalize-burn` function.

Status

Not implemented. The development team informed us that this behavior is by design. Actual unstaking will happen when the `request-burn` function is invoked. If at a later time `revoke-burn` is invoked, then the tokens will be staked again, losing the rewards for the time between `request-burn` and `revoke-burn` are invoked.

EN-03 Non-Standard Transfer

Location

- `contracts/aux/xlink-staking.clar: 1, 146, 193, 215`

[8] https://cwe.mitre.org/data/definitions/670.html

The 'SP2XD7417HGPRTREMKF748VNEQPDRR0RMANB7X1NK.trait-sip-010.sip-010-trait trait defines the transfer-fixed function but it is not defined in the standard[9].

This limits the token contracts that can be used in the contract.

Recommendation

Use the standard transfer function instead.

Status

Not implemented. This is acknowledged by the development team.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Use of tx-sender for Authentication of DAO

While in general it is not advisable to use tx-sender to authenticate roles, and that is specially important when no tokens are transferred[10], we did not mark the use of tx-sender in the is-dao-or-extension functions in any of the analyzed contracts[11].

The function code is (reformated for extra clarity)

```
(define-read-only (is-dao-or-extension)
   (ok (asserts! (or
      (is-eq tx-sender 'SP2XD7417HGPRTREMKF748VNEQPDRR0RMANB7X1NK.executor-dao)
      (contract-call? 'SP2XD7417HGPRTREMKF748VNEQPDRR0RMANB7X1NK.executor-dao
is-extension contract-caller)
   ) err-unauthorised))
)
```

There it can be seen that tx-sender is compared against a specific contract, so the usual phishing problems that are normally introduced when tx-sender is used for authentication are not present.

[9] See https://github.com/stacksgov/sips/blob/main/sips/sip-010/sip-010-fungible-token-standard.md
[10] See https://www.coinfabrik.com/blog/tx-sender-in-clarity-smart-contracts-is-not-adviced/
[11] See contracts/aux/xlink-staking.clar:49-50 and contracts/liabtc/liabtc-mint-endpoint.clar:24-25.

For a general overview on how the `executor-dao` works, see
https://www.alexlab.co/blog/alex-dao

Centralization

xlink-staking

This contract does not have any non-priviledged operation. All the actions except for `add-rewards` need to be made by the dao or an extension.

It must be noted that the dao or any extension may withdraw all the funds for any token by either calling the `withdraw` function after whitelisting the token or by whitelisting an attacking token, changing the validators and calling `add-rewards` with the attacking token, leveraging the contract call in line 172.

```
(and (> delta u0) (as-contract (try! (contract-call? token-trait mint-fixed delta
tx-sender))))
```

This is not a problem by itself, as the dao and extensions are expected to be trustworthy, but shows why the separation of concerns is proposed in ME-01.

liabtc-mint-endpoint

The dao or the extensions can do normal administrative actions that no other accounts can do. See the Priviledged Roles section for details.

Upgrades

There are no provisions to do upgrades in the analyzed contracts.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

xlink-staking

Dao or Extensions

The tracking of extensions is handled in the `executor-dao` contract, which is outside the scope of this audit. The check for this role is implemented in the `is-dao-or-extension` function[12].

[12] See `contracts/aux/xlink-staking.clar:49-50`

An account with this role can:

1. Set the number of required validators via the `set-required-validators` function.
2. Pause or resume operations via the `set-paused` function.
3. Set the older block to accept messages via the `set-block-threshold` function.
4. Change the tracked accrued rewards via the `set-accrued-rewards` function.
5. Whitelist or blacklist a token to be used in the staking via the `set-approved-token` function.
6. Add a new validator via the `add-validator` function.
7. Remove a validator via the `remove-validator` function.
8. Withdraw funds from the contract via the `withdraw` function.
9. Stake tokens via the `stake` function. In order to do so it needs a valid message signed by the validators.
10. Unstake tokens via the `stake` function. In order to do so it needs a valid message signed by the validators.

See ME-01 for problems with this role.

Validators

A set of validators can log the addition of rewards via the `add-rewards` function. This function does not use the native way of validating a message, instead checking on the signature of the message passed. This logic is also triggered as part of the `stake` and `unstake` functions.

The number of validators required can be set by the dao or extensions via the `set-required-validators` function. Validators can be added or removed by the dao or extensions via the `add-validator` and `remove-validator` functions.

liabtc-mint-endpoint

Dao or Extensions

The tracking of extensions is handled in the `executor-dao` contract, which is outside the scope of this audit. The check for this role is implemented in the `is-dao-or-extension` function[13].

An account with this role can:

1. Configure the contract to check the users of the contract against a whitelist or not via the `set-use-whitelist` function.
2. Add or remove an account to/from the whitelist via the `set-whitelisted` function.

[13] See `contracts/liabtc/liabtc-mint-endpoint.clar:24-25`

3. Add and/or remove several accounts to/from the whitelist via the `set-whitelisted` function.
4. Start and stop the operation of the contract via the `set-paused` function[14].
5. Set the minimum number of burn blocks to pass after requesting a burn operation to actually do it via the `set-burn-delay` function.

Minter

If the contract is configured with a whitelist. Only the accounts in the whitelist can mint or revoke burns via the `mint` and `revoke-burn` functions.

xlink.staking Validators

In order to execute the `mint`, `request-burn` and `revoke-burn functions` a valid and up to date message signed by a set of validators needs to be provided.

Funds Flow Analysis

xlink-staking

The only token operated is the one passed as `token-trait` in several functions and whitelisted by the `get-approved-token-or-default` function.

It is minted to the contract itself in the `add-rewards` function, which is a public function that is also called by the `stake` and `unstake` functions.

It is transferred away from the contract in the `withdraw` and `unstake` functions.

It is transferred into the contract in the `stake` function.

liabtc-mint-endpoint

token-liabtc

These tokens are minted to the `tx-sender` in the `mint` public function, that is also called from the `revoke-burn`.

These tokens are burned from `tx-sender` in the `request-burn` function.

[14] Contract pausing has been split into mint-pausing and burn-pausing using the `set-mint-paused` and `set-burn-paused` functions for the fixes commit.

SP2XD7417HGPRTREMKF748VNEQPDRR0RMANB7X1NK.token-abtc

These tokens are:

- transferred to `xlink-staking` in the `mint` public function.
- transferred from `xlink-staking` to `liabtc-mint-registry` in the `request-burn` function.
- transferred from `liabtc-mint-registry` to `xlink-staking` in the revoke-burn function.
- transferred .liabtc-mint-registry to the contract in the `finalize-burn` function.[15]

These tokens can also be minted to `xlink-staking` according to the validators' information when the `mint`, `request-burn` and `revoke-burn` functions are executed.

About CoinFabrik

CoinFabrik is a research and development company specialized in Web3, with a strong background in cybersecurity. Founded in 2014, we have worked on over 500 decentralization projects, including EVM-based and other platforms like Solana, Algorand, and Polkadot. Beyond development, we offer security audits through a dedicated in-house team of senior cybersecurity professionals, working on code in languages such as Substrate, Solidity, Clarity, Rust, TEAL, and Stellar Soroban.

Our team has an academic background in computer science, software engineering, and mathematics, with accomplishments including academic publications, patents turned into products, and conference presentations. We actively research in collaboration with universities worldwide, such as Cornell, UCLA, and École Polytechnique in Paris, and maintain an ongoing collaboration on knowledge transfer and open-source projects with the University of Buenos Aires, Argentina. Our management and people experience team has extensive expertise in the field.

Methodology

CoinFabrik was provided with the source code. Our auditors spent one week auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses.

[15] See EN-04.

- Arithmetic errors
- Race conditions
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

After delivering a report with our findings, the development team had the opportunity to comment on every finding and fix the issues they considered convenient. Once fixed and/or commented, our team ran a second review process to verify that the changes to the code effectively solve the issues found and do not unintentionally add new ones. This report includes the final status after the second review.

Severity Classification

Security risks are classified as follows:

🟥 Critical	These are issues that we manage to exploit. They compromise the system seriously. Blocking bugs are also included in this category. They must be fixed **immediately**.
🟧 High	These refer to a vulnerability that, if exploited, could have a substantial impact, but requires a more extensive setup or effort compared to critical issues. These pose a significant risk and **demand immediate attention**.
🟨 Medium	These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
🟨 Minor	These issues represent problems that are relatively small or difficult to take advantage of, but might be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**

Issue Status

An issue detected by this audit has one of the following statuses:

- **Unresolved:** The issue has not been resolved.
- **Resolved:** Adjusted program implementation to eliminate the risk.
- **Partially Resolved:** Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Acknowledged:** The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **Mitigated:** Implemented actions to minimize the impact or likelihood of the risk.

Disclaimer

This audit report has been conducted on a **best-effort basis within a tight deadline defined by time and budget constraints**. We reviewed only the specific smart contract code provided by the client at the time of the audit, detailed in the Scope section. We do not review other components that are part of the solution: neither implementation, nor general design, nor business ideas that motivate them.

While we have employed the latest tools, techniques, and methodologies to identify potential vulnerabilities, **this report does not guarantee the absolute security of the contracts, as undiscovered vulnerabilities may still exist.** Our findings and recommendations are suggestions to enhance security and functionality and are not obligations for the client to implement.

The results of this audit are valid solely for the code and configurations reviewed, and any modifications made after the audit are outside the scope of our responsibility. CoinFabrik disclaims all liability for any damages, losses, or legal consequences resulting from the use or misuse of the smart contracts, including those arising from undiscovered vulnerabilities or changes made to the codebase after the audit.

This report is intended exclusively for the **XLink** team and should not be relied upon by any third party without the explicit consent of CoinFabrik. Blockchain technology and smart contracts are inherently experimental and involve significant risk; users and investors should fully understand these risks before deploying or interacting with the audited contracts.

Changelog

Date	Description
2024-11-14	Initial report based on commit `47cbf8db47579c8afc59d4f5a73ef53caa507d58`.
2024-11-20	Check fixes on commit `69c4974db88a87930a183cb53396b1a7f4b55cf5`.
2024-11-25	Additional finding on commit `db5500de46f17e93e25e4941e5ccf4d5b3544987`.
2024-11-26	Check additional finding fix on commit `67848619b86be302a18d8f9b4dfe5526e46864cb`.