CoinFabrik

Security Audit Report

XLink - Peg-out Endpoints

November 2024

V202410

Executive Summary

CoinFabrik was asked to audit the contracts for **XLink's Peg-out Endpoints** project.

XLink serves as a bi-directional bridge, enabling the transfer of assets between Bitcoin and its Layer 2 networks. It facilitates interaction with L2 smart contracts using native Bitcoin (BTC) or assets issued on Bitcoin's main network.

During this audit we found one critical issue, one high issue, two medium issues and one minor issue. Also, an enhancement was proposed.

The critical issue was fixed. The high issue was mitigated. One medium issue was acknowledged and the other one was mitigated. The minor issue was acknowledged. The enhancement was implemented.

Scope

The audited files are from the git repository located at https://github.com/xlink-network/xlink, in the `./packages/contracts/bridge-stacks/contracts/` directory. The audit is based on the commit `c97ffe567d1113475b63d6d6607215b99403a0a8`. Fixes were reviewed on commit `b785eaf471daaaea7d90271f67af2c4883d2b6fe`.

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

- `./btc-peg-out-endpoint-v2-01.clar`: Peg-out endpoint from Stacks' aBTC to Bitcoin's BTC.
- `./cross-peg-out-endpoint-v2-01.clar`: Peg-out endpoint from any whitelisted Stacks' SIP-010 to other blockchains.
- `./meta-peg-out-endpoint-v2-03.clar`: Peg-out endpoint from any whitelisted Stacks' SIP-010 to Bitcoin's BRC-20.
- `./cross-router-v2-02.clar`: Helper for swapping tokens with a route of at most 5 tokens.
- `./utils/bridge-common-v2-02.clar`: Helper with functions shared by the endpoints.

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.

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	Remote Code Execution Through Unvalidated token-trait Parameter	🟥 Critical	**Resolved**
HI-01	Loss of Claimer Funds Due to Race Condition in Peg-Out Finalization	🟧 High	**Mitigated**
ME-01	Unauthorized Revocation of Peg-Out Requests	🟨 Medium	**Acknowledged**
ME-02	Denial of Service via Repeated Claiming of Peg-Out Requests	🟨 Medium	**Mitigated**
MI-01	Panicking on Possible Error	🟨 Minor	**Acknowledged**

Critical Severity Issues

CR-01 Remote Code Execution Through Unvalidated token-trait Parameter

Location

- `./meta-peg-out-endpoint-v2-03.clar: 167, 205`

Classification

- CWE-20: Improper Input Validation[1]

[1] https://cwe.mitre.org/data/definitions/20.html

Description

The functions `finalize-peg-out` and `revoke-peg-out` in `meta-peg-out-endpoint-v2-03` are vulnerable to remote code execution due to the lack of validation on the `token-trait` parameter. By not validating this parameter, attackers can supply any contract that implements the trait.

Within these functions, methods of the `token-trait` are invoked using `as-contract`, which changes the `tx-sender` to the contract itself. This grants the invoked methods the contract's authority, allowing the supplied contract to execute arbitrary code with elevated privileges. This vulnerability enables remote code execution.

Possible calls enabled by this remote code execution:

- aBTC contract: mint, burn, and transfer tokens.
- Any wrapped BRC-20: mint, burn, and transfer tokens.
- `meta-bridge-registry-v2-03:` modify requests.
- Other contracts where DAO extensions are authorized.

Steps to Exploit

1. The attacker initiates a peg-out request for then being able to call the `revoke-peg-out` function. Alternatively, fulfill a peg-out request on the Bitcoin network in order to be able to call `finalize-peg-out`.
2. Invokes the `finalize-peg-out` or `revoke-peg-out` functions, providing a malicious contract as the `token-trait` parameter.
3. The functions use `as-contract` to call methods on `token-trait`, which executes the malicious code with the contract's authority.

Recommendation

Ensure that `token-trait` corresponds to an authorized and trusted contract. Consider implementing a whitelist and checking parameter addresses against it.

Status

Resolved. Fixed according to the recommendation.

High Severity Issues

HI-01 Loss of Claimer Funds Due to Race Condition in Peg-Out Finalization

Location

- `./btc-peg-out-endpoint-v2-01.clar: 178`
- `./meta-peg-out-endpoint-v2-03.clar: 167-200`

Classification

- CWE-703: Improper Check or Handling of Exceptional Conditions[2]

Description

In the `finalize-peg-out` function, a check that ensures the peg-out request is still within the claimed period has been commented out:

```
;; (asserts! (<= block-height (get claimed request-details)) err-request-claim-expired) ;;
allow fulfilled if not claimed again
```

Without this check, a race condition arises where the initial claimer, who has already transferred funds, may not receive the corresponding wrapped tokens. After the claimed period expires, the requester can revoke the request or another user can re-claim it, effectively invalidating the initial claimer. This means the initial claimer could incur losses by transferring tokens, only to have the request finalized by someone else or revoked.

Steps to Exploit

1. The attacker initiates a peg-out request, becoming the requester.
2. Observes that a claimer has claimed the request but has not fulfilled it within the grace period.
3. Monitors the Bitcoin network to detect the transaction that fulfills the peg-out request.
4. Before the original claimer finalizes the peg-out request, the attacker revokes the peg-out request.
5. The claimer, who transferred funds on Bitcoin, is unable to finalize the peg-out request and does not receive the corresponding wrapped tokens.

This issue also applies to the function of the same name in `meta-peg-out-endpoint-v2-03`, where there is no commented out line.

[2] https://cwe.mitre.org/data/definitions/703.html

Recommendation

Reinstate the commented-out check within the `finalize-peg-out` function to prevent this race condition.

Status

Mitigated. request-revoke-grace-period is set higher than request-claim-grace-period. The first one is set initially to 432, while the second is set to 144. Then, the requester can only revoke requests after 432 tenures (approximately 72 hours) from request initialization.

However, another user can still front-run, claim the request before it is finalized, and finalize it with the previous requester's bitcoin transaction. Only when the peg-out was not fulfilled within the claiming grace period (approximately 24 hours).

Medium Severity Issues

ME-01 Unauthorized Revocation of Peg-Out Requests

Location

- `./btc-peg-out-endpoint-v2-01.clar: 199`
- `./meta-peg-out-endpoint-v2-03.clar: 205`

Classification

- CWE-862: Missing Authorization[3]

Description

The `revoke-peg-out` function in the contract can be called by any user without any authorization checks. This means that any user can revoke any peg-out request, regardless of who created it. This opens the system to griefing attacks where malicious actors can disrupt the normal operation by revoking other users' peg-out requests.

Steps to Exploit

1. An attacker monitors the network for new peg-out requests.
2. Upon detecting a new request, the attacker immediately calls `revoke-peg-out` with the `request-id` of the victim's peg-out request.
3. The victim's request is revoked without their consent, and they must initiate a new request, experiencing delays.

[3] https://cwe.mitre.org/data/definitions/862.html

4. The attacker can repeat this process indefinitely, causing continuous disruption.

Recommendation

Add an authorization check to ensure that only the user who created the peg-out request can revoke it.

Status

Acknowledged. This is intentional. Since there is a grace period before being able to revoke the request, they can be fulfilled without being exposed to this issue. If they are not fulfilled, anyone should be able to revoke them.

ME-02 Denial of Service via Repeated Claiming of Peg-Out Requests

Location

- `./btc-peg-out-endpoint-v2-01.clar: 141-152`
- `./meta-peg-out-endpoint-v2-03.clar: 131-146`

Classification

- CWE-400: Uncontrolled Resource Consumption[4]

Description

The peg-out request system allows any user to claim a peg-out request. This enables malicious actors to repeatedly claim a request, preventing others from fulfilling it. An attacker can monitor the network for new peg-out requests and immediately claim them, setting the `claimed` field to a future block height (current block height + `claim-grace-period`). During this period, the request cannot be claimed by others or revoked by the requester. If the attacker does not finalize the request within the grace period, they can reclaim it again after the grace period expires, effectively blocking legitimate actors from fulfilling the request indefinitely. This results in a denial of service, causing the original requester to experience delays or be unable to have their peg-out request fulfilled within a reasonable time frame.

Steps to Exploit

1. The attacker watches the network for new peg-out requests.
2. Upon detecting a new request, the attacker immediately calls the claiming function with the `request-id`.
3. The attacker intentionally does not fulfill the request within the grace period.

[4] https://cwe.mitre.org/data/definitions/400.html

4. After the grace period expires, the attacker reclaims the request, resetting the claimed field to a new future block height.
5. Steps 4-5 are repeated indefinitely, blocking legitimate actors from fulfilling the request.

Recommendation

Introduce a system that monitors for abnormal claiming patterns, such as multiple unfulfilled claims from various accounts. When such patterns are detected, automatically activate a whitelist of trusted entities allowed to claim and fulfill peg-out requests.

Status

Mitigated. The development team stated this is mitigated by their monitoring of the network activities and resolving such DDOS attacks by executing governance actions.

Minor Severity Issues

MI-01 Panicking on Possible Error

Location

- `./utils/bridge-common-v2-02.clar: 40, 41, 42, 134`

Classification

- CWE-248: Uncaught Exception[5]

Description

Using `unwrap-panic` results in the transaction being finished because of a runtime error when the provided value is an error or a none. The runtime error does not allow the caller to handle that error and act in response. Also, this kind of error does not provide any information about the reason for the reverted transaction to the user.

While that form is a convenient method to unwrap values, it should not be used unless it is impossible to trigger the panic.

Recommendation

Replace `unwrap-panic` for `unwrap!` when there is a flow which might trigger an error or none value.

[5] https://cwe.mitre.org/data/definitions/248.html

Status

Acknowledged. This was acknowledged by the development team.

Enhancements

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

Id	Title	Status
EN-01	Use of block-height for Time Windows	**Implemented**

EN-01 Use of block-height for Time Windows

Location

- `./btc-peg-out-endpoint-v2-01.clar: 146, 202, 203`
- `./meta-peg-out-endpoint-v2-03.clar: 137, 210, 211`

Description

In both endpoints `block-height` is used to calculate time windows. However, this is unreliable since the Nakamoto Release. Now there is a variable amount of Stacks blocks for each Bitcoin block. This makes the time window for grace periods inconsistent.

Recommendation

Use `burn-block-height` instead, as it corresponds to Bitcoin blocks, with an average block time of 10 minutes.

Status

Implemented. Now `tenure-height` is used instead in `meta-peg-out-endpoint-v2-03.clar`. Since `btc-peg-out-endpoint-v2-01.clar` was already deployed on mainnet before epoch 3, therefore `block-height` is equal to `tenure-height`.

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.

Centralization

Both bridges from Bitcoin (`btc-peg-out-endpoint-v2-01` and `meta-peg-out-endpoint-v2-03`) operate without dependence on a privileged role. The contract owner, which will be a DAO, can modify the fees, modify the address where the fees are transferred, and pause the contracts.

In `meta-peg-out-endpoint-v2-03`, the owner can also transfer any SIP-010 token from the endpoint to an arbitrary address.

In `cross-peg-out-endpoint-v2-01`, the DAO can pause the contract, apply a whitelist and whitelist addresses. In this bridge, receiving the tokens on the other network depends on an intermediary who validates the operation.

Upgrades

The contracts do not implement upgradeability mechanisms.

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 two weeks 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.

- Arithmetic errors
- Race conditions
- Misuse of block timestamps
- Denial of service attacks
- Excessive runtime 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
2024-11-19	Initial report based on commit `c97ffe567d1113475b63d6d6607215b99403a0a8`.
2024-11-25	Final report based on commit b785eaf471daaaea7d90271f67af2c4883d2b6fe.