Security Audit

Balanced Move (DeFi)

December 2024

info@hashlock.com.au

Hashlock Pty Ltd

Table of Contents

CAUTION

Executive Summary

The ICON Foundation team partnered with Hashlock to conduct a security audit of their Balanced/Xcall smart contracts. Hashlock manually and proactively reviewed the code in order to ensure the project's team and community that the deployed contracts are secure.

Project Context

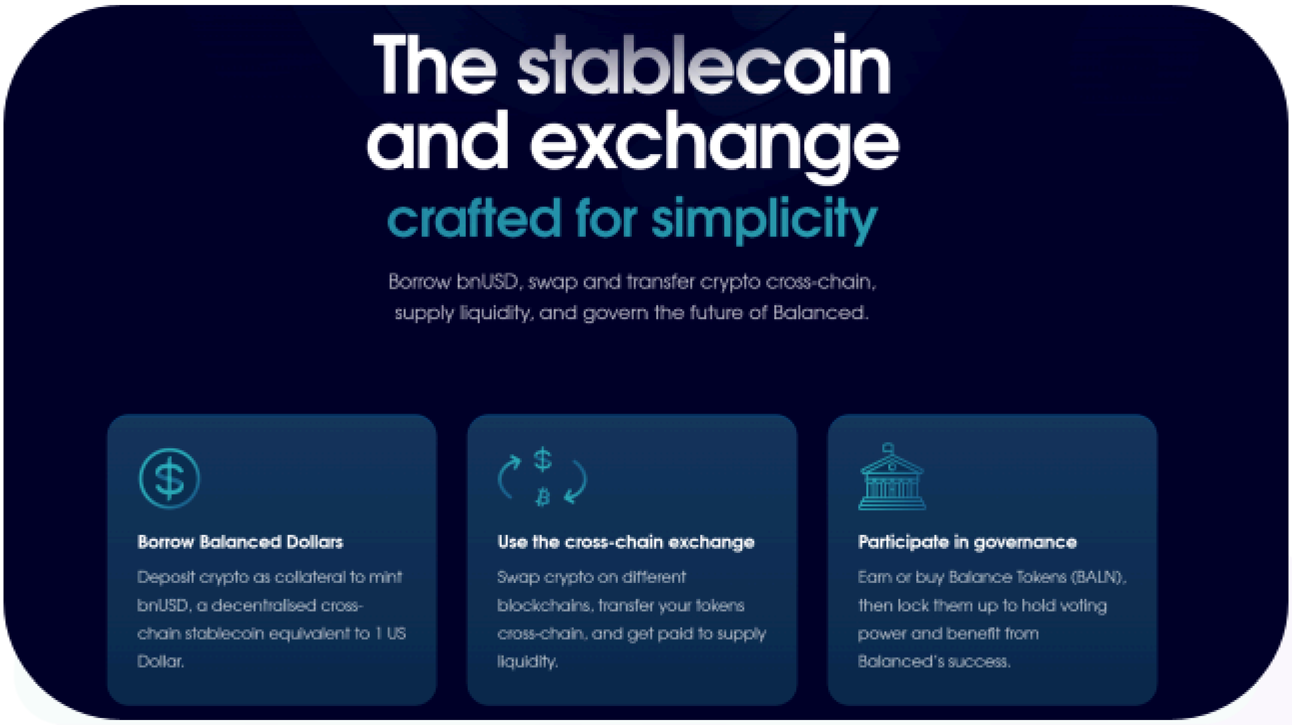
Balanced is a DeFi product crafted for simplicity. It's home to the Balanced Dollar stablecoin (bnUSD), and the largest decentralised exchange on the ICON blockchain.
You can use Balanced to borrow bnUSD, swap assets, supply liquidity, and participate in governance. And now, thanks to xCall, you can also swap and transfer assets cross-chain.

Project Name: Balanced
Compiler Version: N/A
Website: www.balanced.network
Logo:

Visualised Context:

Project Name	Balanced
Launch Date	TBA
Compiler Version	N/A
Language	Move
Network	SUI
Token Ticker	bnUSD

Project Visuals:



The stablecoin and exchange crafted for simplicity

Borrow bnUSD, swap and transfer crypto cross-chain, supply liquidity, and govern the future of Balanced.

Borrow Balanced Dollars

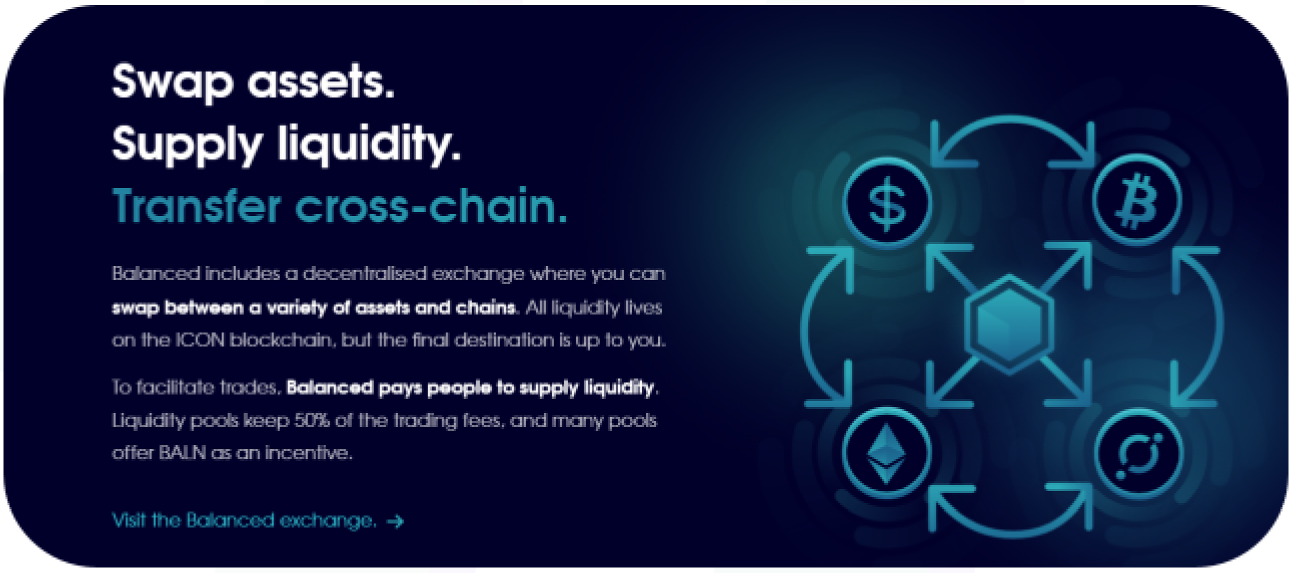
Deposit crypto as collateral to mint bnUSD, a decentralised cross-chain stablecoin equivalent to 1 US Dollar.

Use the cross-chain exchange

Swap crypto on different blockchains, transfer your tokens cross-chain, and get paid to supply liquidity.

Participate in governance

Earn or buy Balance Tokens (BALN), then lock them up to hold voting power and benefit from Balanced's success.



Swap assets.
Supply liquidity.
Transfer cross-chain.

Balanced includes a decentralised exchange where you can **swap between a variety of assets and chains**. All liquidity lives on the ICON blockchain, but the final destination is up to you.

To facilitate trades, **Balanced pays people to supply liquidity**. Liquidity pools keep 50% of the trading fees, and many pools offer BALN as an incentive.

Visit the Balanced exchange. →

Audit scope

We at Hashlock audited the solidity code within the Balanced project, the scope of work included a comprehensive review of the smart contracts listed below. We tested the smart contracts to check for their security and efficiency. These tests were undertaken primarily through manual line-by-line analysis and were supported by software-assisted testing.

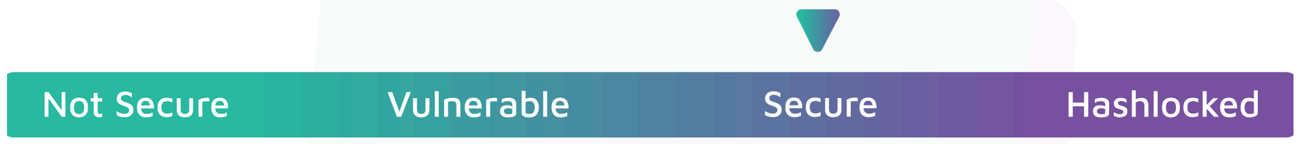
Description	**Balanced Protocol Smart Contracts**
Platform	**Move**
Audit Date	**December, 2024**
Contract 1	balanced/sources/xcall_manager.move
Contract 1 MD5 Sum	896b4009c50fc53f5961cfb74392b597
Contract 2	balanced/sources/utils.move
Contract 2 MD5 Sum	4102e5c079e506b69190d8b43a1d617f
Contract 3	balanced/sources/asset_manager.move
Contract 3 MD5 Sum	2cfdd6571838051ccb665ff8d1479f6b
Contract 4	balanced_crosschain/sources/balanced_crosschain.move
Contract 4 MD5 Sum	48339d042132c9b0d2f711f122c9fe7f
Contract 5	balanced_crosschain/sources/bnusd_crosschain.move
Contract 5 MD5 Sum	d4b7a183881d7077dfc4eb815ee5dfa0
Contract 6	balanced_tokens/balanced_dollar/sources/balanced_dollar.move
Contract 6 MD5 Sum	87218c1ad6a52a3788e36a08b9535895
Contract 7	balanced/sources/messages/cross_transfer.move
Contract 7 MD5 Sum	e46f6ca72175da687011eadc60c59c49
Contract 8	balanced/sources/messages/deposit.move

Contract 8 MD5 Sum	f38149330183fd8af787bdf7947305e3
Contract 9	balanced/sources/messages/cross_transfer_revert.move
Contract 9 MD5 Sum	e70d67bea813cb9d7a8d02dd379c1800
Contract 10	balanced/sources/messages/configure_protocol.move
Contract 10 MD5 Sum	c753baa660c701e3feb79198bdf14e98
Contract 11	balanced/sources/messages/deposit_revert.move
Contract 11 MD5 Sum	572e0d97b6facd8a64a69393ed7fcd4a
Contract 12	balanced/sources/messages/withdraw_to.move
Contract 12 MD5 Sum	35cfc8256809598e39f93a4f1b457990
Contract 13	contracts/sui/xcall/sources/cluster_connection/cluster_connection.move
Contract 13 MD5 Sum	f8cf07baabd86f94cb9cb447459f2e5d
Contract 14	contracts/sui/xcall/sources/cluster_connection/cluster_state.move
Contract 14 MD5 Sum	1fcba09c4a41fb060d9100e907873d74
Contract 15	contracts/sui/xcall/sources/cluster_connection/cluster_entry.move
Contract 15 MD5 Sum	511142bfbe318b0f80ad7e0de09102f8
Contract 16	contracts/sui/xcall/sources/messages/envelope.move
Contract 16 MD5 Sum	acfa375942aa911fe553a10d09e6580f
Contract 17	contracts/sui/xcall/sources/states/xcall_state.move
Contract 17 MD5 Sum	0f4c55c0ed49f05dabb8855881dc8626
Contract 18	contracts/sui/xcall/sources/utils.move
Contract 18 MD5 Sum	abd5a9a9b864e4cb6420b6f26115169a
Contract 19	contracts/sui/xcall/sources/main.move
Contract 19 MD5 Sum	700612ef98a19dfb1b6ce407d6eafa6b
Contract 20	contracts/sui/xcall/sources/connections.move

Contract 20 MD5 Sum	c46b1f6ab921baca1b9091c34dbf8f58
Contract 21	contracts/sui/xcall/sources/messages/call_message.move
Contract 21 MD5 Sum	2b3248c0ce983b435e42a17b307e3774
Contract 22	contracts/sui/xcall/sources/messages/persistent_message.move
Contract 22 MD5 Sum	289cf9b7449d3bafc8ebabc74c09425e
Contract 23	contracts/sui/xcall/sources/messages/call_message_rollback.move
Contract 23 MD5 Sum	070715795c210c432d4bb69ecf7e0de5
Contract 24	contracts/sui/xcall/sources/types/message_request.move
Contract 24 MD5 Sum	066b3207ff30f1613a9cb05d80dd9e95
Contract 25	contracts/sui/xcall/sources/types/network_address.move
Contract 25 MD5 Sum	bac56ff1ead360a05af53a5a74531a59
Contract 26	contracts/sui/xcall/sources/types/cs_message.move
Contract 26 MD5 Sum	f1c7a5f69f485b8d2c68b18e39824500
Contract 27	contracts/sui/xcall/sources/types/rollback_ticket.move
Contract 27 MD5 Sum	859312f765ae87d57f29db351c522dd1
Contract 28	contracts/sui/xcall/sources/types/message_result.move
Contract 28 MD5 Sum	a142aa7f8e850d262392bef8c5aa60f8
Contract 29	contracts/sui/xcall/sources/types/execute_ticket.move
Contract 29 MD5 Sum	8b370c56b8de01bc01a9024eac1e4276
Contract 30	contracts/sui/xcall/sources/types/rollback_data.move
Contract 30 MD5 Sum	a6e6e2395842219e153056ec6bc1028f

Security Rating

After Hashlock’s Audit, we found the smart contracts to be **“Secure”**. The contracts all follow simple logic, with correct and detailed ordering. We initially identified some significant vulnerabilities that have since been addressed.



The ‘Hashlocked’ rating is reserved for projects that ensure ongoing security via bug bounty programs or on chain monitoring technology.

All issues uncovered during automated and manual analysis were meticulously reviewed and applicable vulnerabilities are presented in the Audit Findings section. The general security overview is presented in the Standardised Checks section and the project's contract functionality is presented in the Intended Smart Contract Functions section.

All vulnerabilities initially identified have now been resolved and acknowledged.

Hashlock found:

2 Medium severity vulnerabilities

3 Low severity vulnerabilities

Caution: *Hashlock’s audits do not guarantee a project's success or ethics, and are not liable or responsible for security. Always conduct independent research about any project before interacting.*

Intended Smart Contract Functions

Claimed Behaviour	Actual Behaviour
balanced/sources/xcall_manager.move - This contract implements the execute_call functionality with tracking capabilities, serving as a core component for cross-chain message execution.	Contract achieves this functionality.
balanced/sources/utils.move - View functions and encoding/decoding utilities.	Contract achieves this functionality.
balanced/sources/asset_manager.move - This contract implements multiple functions including both administrative and public interfaces for managing assets within the Balanced ecosystem.	Contract achieves this functionality.
balanced_crosschain/sources/balanced_crosschain.move - This contract implements various cross-chain functionalities without entry points, focusing on internal protocol operations.	Contract achieves this functionality.
balanced_crosschain/sources/bnusd_crosschain.move - This contract implements the ecosystem stablecoin, bnUSD	Contract achieves this functionality.
balanced_tokens/balanced_dollar/sources/balanced_dollar.move - This contract implements core functionality for minting and burning Balanced Dollar tokens.	Contract achieves this functionality.

balanced/sources/messages/cross_transfer.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
balanced/sources/messages/deposit.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
balanced/sources/messages/cross_transfer_revert.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
balanced/sources/messages/configure_protocol.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
balanced/sources/messages/deposit_revert.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
balanced/sources/messages/withdraw_to.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/cluster_connection/cluster_connection.move - This contract implements a focused set of public package functions for cluster connectivity.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/cluster_connection/cluster_state.move - This contract implements validator logic through public package functions for managing cluster state. - Manages core state variables	**Contract achieves this functionality.**
contracts/sui/xcall/sources/cluster_connection/cluster_entry.move - This contract implements numerous functions utilizing connection capabilities for cluster	**Contract achieves this functionality.**

management. - Exposes entry points callable by the external actors	
contracts/sui/xcall/sources/messages/envelope.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/states/xcall_state.move - This contract implements state-related functions and capability management with specific access control mechanisms.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/utils.move - View functions and encoding/decoding utilities. - A sweep-like function allowing to funds withdrawal	**Contract achieves this functionality.**
contracts/sui/xcall/sources/main.move - This contract serves as the main entry point for the xCall system.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/connections.move - This contract implements primarily package-level functions related to connections.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/messages/call_message.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/messages/persistent_message.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/messages/call_message_rollback.move	**Contract achieves this functionality.**

- View functions and encoding/decoding utilities.	
contracts/sui/xcall/sources/types/message_request.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/types/network_address.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/types/cs_message.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/types/rollback_ticket.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/types/message_result.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/types/execute_ticket.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**
contracts/sui/xcall/sources/types/rollback_data.move - View functions and encoding/decoding utilities.	**Contract achieves this functionality.**

Code Quality

This audit scope involves the smart contracts of the Balanced project, as outlined in the Audit Scope section. All contracts, libraries, and interfaces mostly follow standard best practices and to help avoid unnecessary complexity that increases the likelihood of exploitation, however, some refactoring was required.

The code is very well commented on and closely follows best practice nat-spec styling. All comments are correctly aligned with code functionality.
The codebase is fairly extensive, with multiple files dedicated to similar types of functionalities. The codebase could be optimized in terms of size to improve maintainability.

Audit Resources

We were given the Balanced project smart contract code in the form of Github access.

As mentioned above, code parts are well commented. The logic is straightforward, and therefore it is easy to quickly comprehend the programming flow as well as the complex code logic. The comments are helpful in providing an understanding of the protocol's overall architecture.

Dependencies

As per our observation, the libraries used in this smart contracts infrastructure are based on well-known industry standard open source projects.
Apart from libraries, its functions are used in external smart contract calls.

Severity Definitions

Significance	Description
High	High-severity vulnerabilities can result in loss of funds, asset loss, access denial, and other critical issues that will result in the direct loss of funds and control by the owners and community.
Medium	Medium-level difficulties should be solved before deployment, but won't result in loss of funds.
Low	Low-level vulnerabilities are areas that lack best practices that may cause small complications in the future.
Gas	Gas Optimisations, issues, and inefficiencies
QA	Quality Assurance (QA) findings are purely informational and don't impact functionality. These notes help clients improve the clarity, maintainability, or overall structure of the code, ensuring a cleaner and more efficient project. They should be addressed for optimization but are not critical to the system's performance or security.

Audit Findings

Medium

[M-01] Xcall#utils.move - destroy_or_transfer_balance can be called by anyone

Description

The `destroy_or_transfer_balance` function in Xcall's utils module is declared as `public fun`, allowing any module to call this function and transfer balances to arbitrary addresses. While the function appears designed as an internal utility for sweeping dust amounts, its public visibility creates unnecessary risk.

Vulnerability Details

The function is implemented as follows:

```
    public fun destroy_or_transfer_balance<T>(balance: Balance<T>, recipient: address,
ctx: &mut TxContext) {

        if (balance::value(&balance) == 0) {

            balance::destroy_zero(balance);

            return

        };

        transfer::public_transfer(

            coin::from_balance(balance, ctx),

            recipient

        );

    }
```

Impact

Any module could potentially transfer balances of any coin type to any address without restriction.

Recommendation

Remove the `public` modifier if the function is meant for internal use only, or implement capability-based access control by requiring an admin capability token if external access is needed.

Note

The ICON Foundation team informed Hashlock that this function is intentionally public to facilitate its utility in transferring balances for different modules. As each module can only transfer its own balance, the current design does not permit unauthorized transfers and aligns with the intended functionality. Although the function is not currently used, the ICON Foundation may remove it in future updates.

Status

Acknowledged

[M-02] Xcall#xcallmanager - Irreversible protocol removal proposition

Description

In the XcallManager module, `propose_removal` allows an admin to mark a protocol for removal by setting `proposed_protocol_to_remove`. Once set, there is no mechanism to revert this decision, creating a risk of permanent unintended modifications to the protocol list.

Vulnerability Details

The function is implemented as follows:

```
    entry fun propose_removal(config: &mut Config, _: &AdminCap, protocol: String) {
enforce_version(config); config.proposed_protocol_to_remove = protocol;

}
```

The function directly sets the protocol to be removed without any confirmation step or reversion mechanism. This state change is immediately reflected in `get_modified_protocols`. While the function requires AdminCap for access control, the permanent nature of the change creates unnecessary operational risk.

Impact

An admin mistake in protocol removal cannot be corrected, potentially leading to permanent disruption of protocol integrations. The severity is Medium as it requires admin access but could significantly impact protocol operations.

Recommendation

Implement a function to clear or reset the `proposed_protocol_to_remove` value, allowing admins to revert unintended removal propositions. Additionally, consider implementing a time delay or multi-step confirmation process for protocol removals.

Note

The ICON Foundation clarified to Hashlock that the propose_removal function does not directly remove protocols from operation but only proposes them for removal. This design limits operational risk, as the protocol list is not immediately impacted. The team also noted that the proposed_protocol_to_remove field can be reset to an empty string ("") if needed, providing flexibility without additional implementation changes.

Status

Acknowledged

Low

[L-01] Multiple Contracts - Administrative operations lack state reversal possibility

Description

Some administrative functions across the protocol implement one-way state changes without the ability to reverse these operations. This pattern appears in multiple modules, creating operational inflexibility for protocol administrators:

```
// asset_manager.move

entry fun register_token(/* params */) {

    // Irreversible token registration

}

// rollback_data.move

entry fun enable_rollback(/* params */) {

    // One-way rollback enablement

}
```

The functions lack corresponding operations to undo their state changes, such as unregistering tokens or disabling rollbacks. While these operations require admin access, the inability to reverse them could complicate protocol management.

Recommendation

Implement corresponding reverse functions for administrative operations, allowing state changes to be undone when necessary.

Note

The ICON Foundation stated that rollback operations inherently delete the entire rollback object upon execution, ensuring no persistence. Similarly, token registration does not adversely affect the contract's state and does not require reversal functionality. The current design sufficiently handles these scenarios.

Status

Acknowledged

[L-02] Xcall#cluster_entry.move - Missing additional safety checks on validator threshold

Description

In `cluster_entry.move`, the `set_validator_threshold` function updates the validator threshold with minimal safety constraints. While it ensures the threshold doesn't exceed the total validator count, it uses a raw number rather than a dynamic threshold based on the validator set size:

```
entry fun set_validator_threshold(xcall:&mut XCallState, cap:&AdminCap, connection_id:String, threshold:u64, _ctx:&mut TxContext) {

    validate_admin_cap(cap,connection_id);

    let state = get_state_mut(xcall_state::get_connection_states_mut(xcall),connection_id);

    cluster_state::set_validator_threshold(state,threshold);

}
```

The implementation allows setting arbitrary thresholds that could become inappropriate when validators join or leave, requiring manual adjustments. This makes the system more maintenance-intensive and potentially vulnerable to `quorum` manipulation if threshold updates lag behind validator set changes.

Recommendation

Implement a dynamic threshold calculation based on the total validator count. For example:

```
let min_threshold = (total_validators / 2) + 1;

assert!(threshold >= min_threshold, INVALID_THRESHOLD);
```

Consider tracking validators in a registry and automatically adjusting thresholds when the validator set changes.

Note

According to the ICON Foundation, the `set_validator_threshold` function is controlled by admin privileges, reducing potential risk. While a dynamic threshold might enhance flexibility, the team indicated that the current admin-controlled design is effective and manageable for the existing implementation.

Status

Acknowledged

[L-03] Balanced_crosschain#balanced_crosschain.move - Unsafe integer downcast in amount translation

Description

In `balanced_crosschain.move`, the `translate_incoming_amount` function performs an unsafe downcast from `u128` to `u64`:

```
fun translate_incoming_amount(amount: u128): u64 {

    (amount / (std::u64::pow(10, 9) as u128)) as u64

}
```

While the division by 10^9 makes overflow less likely, the direct cast operation `as u64` could potentially truncate values without warning if the result exceeds `u64::MAX`. This silent failure mode could lead to incorrect amount `translations` for very large transactions.

Recommendation

Add explicit bounds checking before the downcast, for example, such as:

```
fun translate_incoming_amount(amount: u128): u64 {

    let result = amount / (std::u64::pow(10, 9) as u128);
    assert!(result <= (std::u64::MAX as u128), AMOUNT_TOO_LARGE);

    (result as u64)

}
```

Status

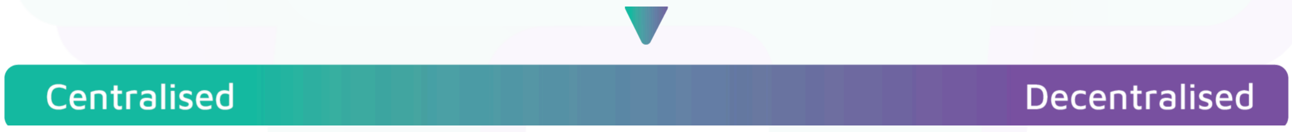
Resolved

Centralisation

The Balanced project prioritizes security and utility over decentralization. All audited contracts are governed by a multisig wallet to ensure secure upgrades.

For any new contracts or changes to take effect, a DAO vote is required, adding an additional layer of security and transparency.

BALN, the governance token, is integral to this process. It is required for voting and has been fairly distributed among Balanced users. This ensures that the protocol remains decentralized, with no single entity having control over decision-making or upgrades.



Conclusion

After Hashlock's analysis, the Balanced project seems to have a sound and well-tested code base, now that our vulnerability findings have been resolved and acknowledged. Overall, most of the code is correctly ordered and follows industry best practices. The code is well commented on as well. To the best of our ability, Hashlock is not able to identify any further vulnerabilities.

Our Methodology

Hashlock strives to maintain a transparent working process and to make our audits a collaborative effort. The objective of our security audits is to improve the quality of systems and upcoming projects we review and to aim for sufficient remediation to help protect users and project leaders. Below is the methodology we use in our security audit process.

Manual Code Review:
In manually analysing all of the code, we seek to find any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behaviour when it is relevant to a particular line of investigation.

Vulnerability Analysis:
Our methodologies include manual code analysis, user interface interaction, and white box penetration testing. We consider the project's website, specifications, and whitepaper (if available) to attain a high-level understanding of what functionality the smart contract under review contains. We then communicate with the developers and founders to gain insight into their vision for the project. We install and deploy the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:
We undergo a robust, transparent process for analysing potential security vulnerabilities and seeing them through to successful remediation. When a potential issue is discovered, we immediately create an issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is vast because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, and then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyse the feasibility of an attack in a live system.

Suggested Solutions:
We search for immediate mitigations that live deployments can take and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the contract details are made public.

Disclaimers

Hashlock's Disclaimer

Hashlock's team has analysed these smart contracts in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in the smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

Due to the fact that the total number of test cases is unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Hashlock is not responsible for the safety of any funds and is not in any way liable for the security of the project.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to attacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.

About Hashlock

Hashlock is an Australian-based company aiming to help facilitate the successful widespread adoption of distributed ledger technology. Our key services all have a focus on security, as well as projects that focus on streamlined adoption in the business sector.

Hashlock is excited to continue to grow its partnerships with developers and other web3-oriented companies to collaborate on secure innovation, helping businesses and decentralised entities alike.

Website: hashlock.com.au
Contact: info@hashlock.com.au