Starknet Token Bound Account Security Review

Conducted by: credence0x
Twitter, Telegram, Discord: @credence0x
E-mail: hello@zerocredence.com

15th March 2024 - 1st April 2024

1. Disclaimer

A smart contract security review does not guarantee the absence of vulnerabilities but I have put in my best effort to find as many vulnerabilities as possible. That being said, subsequent security reviews, bug bounty programs and on-chain monitoring are strongly still recommended.

2. Introduction

A time-constrained security review of the ***Starknet-Africa-Edu/tba*** repository was done by me, with a focus on the security aspects of the application's smart contracts.

3. About Starknet Token Bound Accounts

Starknet Token Bound Accounts is an initiative by the starknet africa exploration team which aims to implement eip-6551 on starknet while incorporating starknet's unique features to make a better standard.

4. Risk Classification

SEVERITY	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

4.1 Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.

- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

4.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

4.3. Action required for severity levels

- Critical - Must fix as soon as possible
- High - Must fix (before deployment)
- Medium - Should fix
- Low - Could fix
- Unclassified - Minor improvements the project should consider making

5. Security Assessment Summary

repository: https://github.com/Starknet-Africa-Edu/tba
review commit hash - c41c29970208fe385d2af9f2939d3e292310323c

6. Scope

The following smart contracts were in scope of this audit

- src/account/*
- src/interfaces/*
- src/presets/*
- src/registry/*
- src/upgradeable/*

7. Executive Summary

Findings Count

Severity	Amount
Critical	1

High	2
Medium	0
Low	1
Total Findings	**4**

Summary of Findings

ID	TITLE	SEVERITY	STATUS
[C-01]	Anyone can create and use token bound accounts before token exists	Critical	RESOLVED
[H-01]	Sequencer can't call the __validate__ or __execute__ function so transactions can't originate from account	High	UNRESOLVED
[H-02]	The token bound account can't validate signatures	High	RESOLVED
[L-01]	`owner` function should not take parameters	Low	RESOLVED

8. Findings

8.1. Critical Findings

[C-01] Anyone can create and use token bound accounts before token exists

https://github.com/Starknet-Africa-Edu/TBA/blob/c41c29970208fe385d2af9f2939d3e292310323c/src/account/account.cairo#L251-L265

Description

Whenever a transaction is sent to starknet, the sequencer calls the `__validate__` function then calls the `__execute__` function. Both these functions call the `_is_valid_transaction` function and the call passes when `caller == owner`. When the sequencer calls `__execute__`, the caller address is 0 and when the token id has no owner yet, the returned address is 0. This means that when these special conditions are met, anyone can deploy and have full access to an account before the rightful owner ever gets control and they may change the class hash if they wish.

The registry prevents this by disallowing deployment if you don't own the token but SNIP72 does not force people to use the singleton registry. It says you SHOULD use it. Not MUST.

Also it may be assumed that since the deployment salt can be specified, people should just specify another salt and get another account. However, token bound accounts specifically enable apps to send rewards to addresses that may not even exist yet and when they are computing these addresses, it is certain that they use a predetermined salt.

Recommendations

Check that the NFT owner is not the zero address in the `initializer` function before the token bound account is created.

Resolved in
https://github.com/Starknet-Africa-Edu/TBA/commit/ec89deb1eab00072e840aa2c11bfd261d25d1f4b

8.2. High Severity Findings

[H-01] Sequencer can't call the __validate__ or __execute__ function so transactions can't originate from account

https://github.com/Starknet-Africa-Edu/TBA/blob/c41c29970208fe385d2af9f2939d3e292310323c/src/account/account.cairo#L109-L124

Description

It is expected that the `__execute__` function will only be called by the sequencer after it has called the `__validate__` function but with the way the code is currently written, that is the only method the token owner can call to interact with the token bound account. By implementing it this way, the sequencer will never be able to call the `__execute__` function successfully as the caller address from the sequencer will be the zero address and the function expects the caller to be the token owner. This means that no transaction can truly originate from that account (except as described in C-01).

Recommendations

It is strongly recommended that the __execute__ function is updated to follow the openzeppelin standard and that the main content of the function (as is) is added to a new function called `execute`, also following the erc6551 standard.

[H-02] The token bound account cant validate signatures

https://github.com/Starknet-Africa-Edu/TBA/blob/c41c29970208fe385d2af9f2939d3e292310323c/src/account/account.cairo#L251-L265

Description

The is_valid_signature function does not validate signatures but tries to ensure that the caller of the function is the NFT owner. That function is meant to be called externally so it is expected that the caller will almost never be the owner. It's just a way for anyone to verify that a hash was signed by an account.

Recommendations

Create a `is_valid_signer` function to verify that the caller of the execute function is the token owner. You can also amend the internal `_is_valid_signature function` to call the is_valid_signature function of the NFT owner's account. This way the signature of the token bound account is always consistent with the signature of whoever owns the token bound account. Here is a reference implementation from realms. https://github.com/BibliothecaDAO/token-abstract-accounts/blob/e15439c02ea8b5e54138d4bb676f26a4566dc532/src/components.cairo#L281.

Resolved in https://github.com/Starknet-Africa-Edu/TBA/commit/ec89deb1eab00072e840aa2c11bfd261d25d1f4b

8.3. Low Severity Findings

[L-01] `owner` function should not take parameters

https://github.com/Starknet-Africa-Edu/TBA/blob/c41c29970208fe385d2af9f2939d3e292310323c/src/account/account.cairo#L129

Description

That function should not need to take in parameters. It is supposed to return the current owner of the token attached to the account

Recommendations

Remove all parameters and use the stored token contract and id to get owner

Resolved in
https://github.com/Starknet-Africa-Edu/TBA/commit/ec89deb1eab00072e840aa2c11bfd261d25d1f4b