Unruggable Memecoin Security Review

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

10th January 2024 - January 24nd 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 ***keep-starknet-strange/unruggable.meme*** repository was done by me, with a focus on the security aspects of the application's smart contracts implementation.

3. About Unruggable Memecoin

Unruggable Memecoin is an initiative by the starkware's exploration team which aims to be a framework that allows memecoins to be created and deployed in such a manner that it is very difficult for the memecoin creators to rug or pull out from a project, while leaving the investors in the project out to dry as this is a somewhat popular trend with memcoins.

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/keep-starknet-strange/unruggable.meme
review commit tag - v-0.1.0-alpha.2
review commit hash - 989f075b9133a2ea5a48ecae1a51f1fa0a5345a1

6. Scope

All smart contracts in **contracts/src** were in scope of the audit

7. Executive Summary

Findings Count

Severity	Amount
Critical	2
High	3

Medium	2
Low	0
Unclassified	5
Total Findings	**12**

Summary of Findings

ID	TITLE	SEVERITY	STATUS
[C-01]	Anyone can prevent a token from being launched on jediswap	Critical	UNRESOLVED
[C-02]	The `ensure_not_multicall` function would handicap aggregators and routers	Critical	UNRESOLVED
[H-01]	There can be an unlimited number of the memecoin holders before launch	High	UNRESOLVED
[H-02]	The initial holders count in `UnruggableMemecoin` may be inaccurate during initialization	High	UNRESOLVED
[H-03]	There is no minimum transfer restriction delay for bot protection	High	UNRESOLVED
[M-01]	The memecoin contract owner address could be the zero address which would make it unable to launch	Medium	UNRESOLVED
[M-02]	It could eventually become expensive to completely withdraw a token from the lock manager	Medium	UNRESOLVED
[U-01]	Unnecessary zero check	Unclassified	UNRESOLVED
[U-02]	The initial supply variable could be renamed to max supply	Unclassified	UNRESOLVED
[U-03]	Multiple `is_launched` checks	Unclassified	UNRESOLVED
[U-04]	The factory contract does not need an owner	Unclassified	UNRESOLVED
[U-05]	Unused Variable	Unclassified	UNRESOLVED

8. Findings

8.1. Critical Findings

[C-01] Anyone can prevent a token from being launched on jediswap

https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/exchanges/jediswap_adapter.cairo#L85

Description

The launch of a token may be prevented, very cheaply, by simply creating a pair on jediswap before the token's actual launch. Whenever the `create_pair` function of `Jediswap Factory` is called, it first checks if the pair already exists (https://github.com/jediswaplabs/JediSwap/blob/15fa9f2c5146bc3968d32a8b48b66208bd77c336/contracts/Factory.cairo#L186-L189) and if it does, the transaction gets reverted.
We also do not have that many tokens on starknet so simply creating a pair for the meme token and `ETH` or `USDC` should do the trick.

Severity

Impact: High, because the token would be unable to launch
Likelihood: High, anyone sufficient motivated could do it cheaply

Recommendations

A check should be made to check if the pool exists first and the quote token amount should be adjusted accordingly if need be

[C-02] The `ensure_not_multicall` function would handicap aggregators and routers

https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L331

Description

The inability to make more than one transfer during the transfer restriction delay period would inevitably hinder aggregators and routers. This is because, to do a swap with an aggregator for example, they usually first transfer the tokens you want to swap to an address which they control, then execute other transactions on your behalf, which might involve transferring the token you want to swap more than once to more than one address.

On the other hand, even when the memecoin token is the token you are supposed to receive after a swap, it also often involves more than one transfer. A simple swap on https://app.ekubo.org/ or https://app.avnu.fi/ would show this.

Severity

Impact: High, because people would find it next to impossible to transfer tokens using existing apps
Likelihood: High, because routers and aggregators are popular

Recommendations

Another mechanism should be come up with to deal with bot protection. I think it would be best if the team considered this themselves.

8.2. High Severity Findings

[H-01] There can be an unlimited number of the memecoin holders before launch

https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L447-L462

Description

The `UnruggableMemecoin` contract specifies that the `MAX_HOLDERS_BEFORE_LAUNCH` should be `10` but anyone can increase or decrease the `pre_launch_holders_count`.

Since transfers of zero transfer values are allowed, anyone can send 0 amount of the token to anyone by calling the `transfer_from(sender, recipient, amount)` function and setting the `sender` of the transferred token to be any address that currently holds the token (other than the factory address) and since the `enforce_prelaunch_holders_limit` function does not check if the transfer value is greater than 0, the holders count would be incremented (https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L454).

```
  # POC: illegal pre launch holders increment

  #[test]
   fn test_illegal_increment_holders() {
       // deploy memecoin contract
let mut memecoin = UnruggableMemecoin::contract_state_for_testing();
       start_prank(CheatTarget::One(snforge_std::test_address()), MEMEFACTORY_ADDRESS());
       UnruggableMemecoin::constructor(ref
memecoin,OWNER(),NAME(),SYMBOL(),DEFAULT_INITIAL_SUPPLY(),INITIAL_HOLDERS(),INITIAL_HOLDERS_AMO
UNTS());

       // ensure that the contract works properly up until this point
       // by asserting that the pre_launch_holders_count is correct
       let initial_holders_count: u8 = INITIAL_HOLDERS().len().try_into().unwrap(); // 2
holders
       assert(
           memecoin.pre_launch_holders_count.read()== initial_holders_count,
               'wrong initial holders count'
       );
       // an unknown person does a zero value transfer so there should still be 2 holder
       let unknown_person = contract_address_const::<'unknown'>();
       start_prank(CheatTarget::One(snforge_std::test_address()), unknown_person);
       memecoin.transfer_from(INITIAL_HOLDER_1(), unknown_person, 0);

       // however there would be pre_launch_holders_count would be 3
       assert(
           memecoin.pre_launch_holders_count.read() == ( 1 + initial_holders_count),
               'wrong current holders count'
       );
   }
```

Similarly, anyone can make the holders count go down by making a zero value transfer from an address that has no memecoin token to an address that holds at least 1 memecoin token. This is because the `enforce_prelaunch_holders_limit` function decrements the pre launch holders count, even if the transfer amount is 0 (https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L447).

Since the memecoin factory will always have some memecoin tokens before launch, this would be exploitable at all times before launch

```
# POC: illegal pre launch holders decrement

#[test]
   fn test_illegal_decrement_holders() {

       // deploy memecoin contract
       let mut memecoin = UnruggableMemecoin::contract_state_for_testing();
       start_prank(CheatTarget::One(snforge_std::test_address()),
MEMEFACTORY_ADDRESS());
       UnruggableMemecoin::constructor(ref
memecoin,OWNER(),NAME(),SYMBOL(),DEFAULT_INITIAL_SUPPLY(),INITIAL_HOLDERS(),INITIAL_
HOLDERS_AMOUNTS());


       // ensure that the contract works properly up until this point
       // by asserting that the pre_launch_holders_count is correct
       let initial_holders_count: u8 = INITIAL_HOLDERS().len().try_into().unwrap();
// 2 holders
       assert(
           memecoin.pre_launch_holders_count.read()== initial_holders_count,
               'wrong initial holders count'
       );


       // an unknown person does a zero value transfer to the memecoin factory
address
       //
       // for this vulnerability to work, its important that
       // 1. the sender (the unknown person in this case) has 0 token balance
       // 2. the recipient has a non zero token balance
       let unknown_person = contract_address_const::<'unknown'>();
       start_prank(CheatTarget::One(snforge_std::test_address()), unknown_person);
       memecoin.transfer(MEMEFACTORY_ADDRESS(), 0);


       // the pre launch holders count reduces by 1
       assert(
           memecoin.pre_launch_holders_count.read() == (initial_holders_count - 1),
               'wrong current holders count'
```

```
        );
    }
```

When this vulnerability is exploited properly, we can either end up having an unlimited number of holders or the max pre launch holders count could be filled up by anyone.

Severity

Impact: Medium, It violates on of the core assumptions of the contract but doesn't seem to cause much damage since in the end, only 10% of the total supply would exist before launch anyway
Likelihood: High, It could easily be done by anyone at any time

Recommendations

The `enforce_prelaunch_holders_limit` function should only be called when the transfer value is non zero.
https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L308C13-L308C37

[H-02] The initial holders count in `UnruggableMemecoin` may be inaccurate during initialization

https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L399

Description

In the `initializer` function in `UnruggableMemecoin` contract, we set the `pre_launch_holders_count` to the number of addresses in the `initial_holders` array but there is nothing stopping anyone from repeating an address in the array. So technically, `pre_launch_holders_count` may read 10 but there is only one holder.

Severity

Impact: Medium, because the information could be relied on
Likelihood: High, because any sufficiently motivated person could do it

Recommendations

The count should reflect the number of unique addresses

[H-03] There is no minimum transfer restriction delay for bot protection

Description

There is no check that prevents the `transfer_restriction_delay` in the `UnruggableMemecoin` contract from being 0. Meanwhile, there is a specified minimum value for `max_percentage_buy_launch.` If it is set to 0, there would be no bot protection after launch and this piece of code would never run https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L310-L333

Severity

Impact: High, because it could severely impact price of tokens if it gets bot sniped. Also, the owner may do a malicious multi call, enabling them to buy up the supply immediately after launch if they think it'll do well.
Likelihood: Medium, it would only happen if the token is popular enough

Recommendations

A minimum transfer restriction delay value should be specified before here https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L166

8.3. Medium Severity Findings

[M-01] The memecoin contract owner address can be the zero address

Description

There is no check that stops the owner of a contract from being the zero address and if it ends up being the zero address, it would become an unlaunchable memecoin.

Severity

Impact: High, because the project would have redeploy contract and existing pre launch holders might have started distributing tokens
Likelihood: Low

Recommendations

A check should be placed here
https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L113

[M-02] It could eventually become expensive to completely withdraw a token from the lock manager

https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/locker/lock_manager.cairo#L190-L212

https://github.com/keep-starknet-strange/unruggable.meme/blob/v0.1.0-alpha.2/contracts/src/locker/lock_manager.cairo#L359-L363

Description

The `user_locks` and `token_locks` arrays in the `LockManager` contracts can be appended to by anyone by simply calling the `lock_tokens` function. When the items in the array start becoming too many, it could take a lot more steps and gas to find an item in an array before removing it, using the `remove_locks_from_list` function. It could thus make it expensive to call the `transfer_lock` function and also the `withdraw`, `partial_withdraw` functions when all liquidity needs to be taken out.

Severity

Impact: Medium, it could make certain function calls expensive in the long run even if the lock manager is only ever called by the factory address

Likelihood:Medium, it could definitely happen but I assume that people would want to withdraw their locked tokens as soon as they can. This means the list might not get too long unless a malicious person wants it to be.

Recommendations

The users could be asked to provide the index of their lock address as a parameter to the required functions above so the process of removing an item from the list becomes a constant time operation. The more expensive option would be to store a map of lock addresses to indices in the contracts then use that when an array item needs to be found.

8.4. Unclassified Impact and Recommended Minor Improvements

[U-01] Unnecessary zero check

https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/token/memecoin.cairo#L447

Description

The `enforce_prelaunch_holders_limit` function checks whether the sender is the zero address but since the token can't be minted to the zero address, can't be burned and can't be transferred to the zero address, the sender address would never be 0.

Recommendations

Remove the check

[U-02] The initial supply variable could be renamed to max supply

Description

For readability, it would be better to rename `intial_supply` in the `UnruggableMemecoun` contract to `max_supply` because since there is no function for minting, other than when the contract's initializer is called, that is the maximum supply of the token that there could ever be

Recommendations

Rename `initial_supply` to `max_supply`

[U-03] Multiple `is_launched` checks

Description

These checks are duplicates of each other
https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/factory/factory.cairo#L141 and
https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/factory/factory.cairo#L138

Recommendations

One Should be removed

[U-04] The factory contract does not need an owner

https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/factory/factory.cairo#L68-L69

Description

The factory contract implements the `Ownable Component` but it isn't used.

Recommendations

It should be removed

[U-05] Unused Variable

https://github.com/keep-starknet-strange/unruggable.meme/blob/989f075b9133a2ea5a48ecae1a51f1fa0a5345a1/contracts/src/exchanges/jediswap_adapter.cairo#L94

Recommendations

It should be removed