Cryptomate Swapper Smart Contracts Audit Report

Contents

1 Introduction

This is a report of an audit of the Cryptomate Swapper Clarity smart contracts by Friedger Müffke from stacks community.

1.1 Summary

The contracts audited are from the 47a3e4a771efb6d813272b916f4c7a0793996307 commit version. The audit review was done by Friedger Müffke. Fixes were made and rechecked based on the commit 53591da8042108325719b46bcb9f29c5ea380aa1.

1.2 Contracts

The following is the list of the Clarity smart contracts reviewed under this audit:

- contracts/cryptomate-dao.clar
- contracts/cryptomate-dao-token-trait.clar
- contracts/ cryptomate-one-step-mint.clar
- contracts/cryptomate-swap.clar
- contracts/cryptomate-token.clar
- contracts/initializable-trait.clar
- contracts/liquidity-token.clar
- contracts/liquidity-token-trait.clar
- contracts/restricted-token-trait.clar
- contracts/wstx-token.clar

1.3 Analyses

The following analyses were performed:

- Misuse of the different call methods
- Functions with excessive gas cost
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters

All analysis were performed under the assumption that the DAO does not whitelist bad contracts.

1.4 Findings and Fixes

Issues	Severity	Status
DAO is fully controlled by the contract deployer	Medium	Acknowledged
The contract *cryptomate-token* does not implement blacklisting as suggested by the code	Medium	Acknowledged
Most permission check was done using *tx-sender*	Minor	Acknowledged
There isn't a way to open the *add-position*, *reduce-position* and *create-pair* to the public in the future	Minor	Acknowledged
The safe-transfer functions in the cryptomate-swap contract give false feeling of security and they are also increasing the gas cost	Enhancement	Not fixed
On *set-contract-address* function in cryptomate-dao contract, the if and is-some check should be changed to match check this would provide less gas	Enhancement	Not fixed
On *set-governance* function in cryptomate-dao contract, name parameter is unnecessary.	Enhancement	Not fixed
In the contract cryptomate-dao, *freeze-governance* would be a better name for the *initialize-governance* function	Enhancement	Not fixed
On *remove-*-token* functions in cryptomate-one-step-mint contract, *remove-*-token-inner* functions should have been used inside the *remove-*-token* to avoid code duplication	Enhancement	Not fixed
In the cryptomate-one-step-mint contract, the *remove-*-token-inner* functions should avoid *unwrap-panic* and *as-max-len* as they are unnecessary and add gas.	Enhancement	Not fixed

In the cryptomate-one-step-mint contract, the remove-all functions should do *(var-set soft-token-list (list))* instead of assigning an empty list which adds the gas cost.	Enhancement	Not fixed
In the cryptomate-swap contract, the *add-postion* and *reduce-postion* functions check unnecessarily for the liquidity trait. This could add to the gas cost.	Enhancement	Not fixed
Use *map-delete* to remove items from a map instead of *map-set*	Enhancement	Not fixed
No need to use begin function if the function body contains just one expression.	Enhancement	Not fixed
Error codes and naming of errors are inconsistent across contracts.	Enhancement	Not fixed
Tests use manual event filtering instead of clarinet assert functions	Enhancement	Not fixed
In cryptomate-swap contract, maps like *blacklist-for-swapping* should use plain types, not tuples.	Enhancement	Not fixed
In the dao contract, payout-address should be initialized with tx-sender for readability.	Enhancement	Not fixed
No tests for create-pair-new-sip10-token-with-stx and create-pair-new-poxl-token-with-stx.	-	Fixed

Table 1: Findings and Fixes

1.5 Severity Classification

Security risks are classified as follows:

- **Critical**: These are issues that we manage to exploit. They compromise the system seriously. **They must be fixed immediately**.

- **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 can 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.

- **Enhancement**: These kinds of findings do not represent a security risk. They are best practices that we suggest to implement.

Severity	Exploitable	Roadblock	To be fixed
Critical	Yes	Yes	Immediately
Medium	in the near future	Yes	As soon as possible
Minor	Unlikely	No	Eventually
Enhancement	No	No	Eventually

Table 2: Severity Classification

2 Issues Found by Severity

2.1 Critical Severity Issues

There weren't any issues found which can be used to exploit the system to stole the funds on the pools or cause a denial of service under the assumption mentioned in the introduction.

2.2 Medium Severity Issues

DAO

The dao being controlled fully by the deployer and no governance as of now means that the deployer has full access to the liquidity pools on the swapper. This can be a serious security issue if the deployer's private key is leaked or if the owner of the key acts badly. So a governance should be implemented as soon as possible.

Cryptomate-token

The *cryptomate-token* has functions to blacklist owners of this token. However, these functions are for information purpose only and do not impose any blacklisting e.g. on the transfer of the token.

2.3 Minor Severity Issues

tx-sender is used instead of *contract-caller* on the following functions of the cryptomate-swap contract. Unless they are meant to be called from another smart contract they all should use contract-caller.

- *add-to-position*
- *reduce-postion*
- *swap-x-for-y*
- *swap-y-for-x*
- *set-fee-to-address*
- *collect-fees*

If there is a plan on opening the liquidity pools for the public to create pairs, add and remove liquidity, it is not possible since all the following function in the cryptomate-swap

smart contract check the address for a white-list. A way to freeze the white-listing should be implemented.

- *create-pair*
- *reduce-position*
- *add-to-position*

2.4 Enhancements

cryptomate-swap contract

The following safe-transfer functions in the cryptomate-swap contract give false feeling of security as the liquidity token can fake the balance. They also are adding to the gas cost.

- *safe-transfer-to-lp*
- *safe-transfer-from-lp*
- *safe-burn*
- *safe-mint*

Unnecessary check for the liquidity token trait on the *add-position* and *reduce-position* functions as the check is done using the data from the liquidity token trait it can easily fake and bypass the check. This expression will also add to the gas cost.

cryptomate-dao contract

On the cryptomate-dao contract the following improvements should be done as they are best practices and some would reduce the gas cost.
- The *set-contract-address* function should use

```
(match current-contract-optional current-contract (map-set
contracts-data { qualified-name: (get qualified-name
current-contract) } { can-mint: false, can-burn: false})
false)
```

instead of

```
(if (is-some current-contract) (map-set contracts-data {
qualified-name: (unwrap-panic (get qualified-name
current-contract))}{ can-mint: false, can-burn: false })}
false)
```

- The *set-governance* function contains unnecessary name parameter as the value for the name parameter would always be the string "governance".
- The best name for the function *initialize-governance* would be *freeze-governance* as it describes its function better.

cryptomate-one-step-mint

The following functions on the cryptomate-one-step-mint contain code duplication and they should use a *contract-call* to an already existing functions.

- *remove-soft-token*
- *remove-poxl-token*
- *remove-liquidity-token*

They all should use the corresponding

- *remove-soft-token-inner*

- *remove-poxl-token-inner*
- *remove-liquidity-token-inner*

after checking their lp-deployer role from the cryptomate-dao contract.
On the following functions in the cryptomate-one-step-mint contract unnecessary complexity is created by using *unwrap-panic* and *as-max-len* calls.

- *remove-soft-token-inner*
- *remove-poxl-token-inner*
- *remove-liquidity-token-inner*
- *remove-soft-token-inner-all*
- *remove-poxl-token-inner-all*
- *remove-liquidity-token-inner-all*

The best practice for each function would as follows
- For *remove-soft-token-inner* function

```
(ok (var-set soft-token-list (filter remove-filter (var-get
soft-token-list))))))
```

instead of

```
(ok (var-set soft-token-list (unwrap-panic (as-max-len? (filter
remove-filter (var-get soft-token-list)) u200))))))
```

-For *remove-poxl-token-inner* function

```
(ok (var-set poxl-token-list (filter remove-filter (var-get
poxl-token-list))))))
```

instead of

```
(ok (var-set poxl-token-list (unwrap-panic (as-max-len? (filter
remove-filter (var-get poxl-token-list)) u200))))))
```

-For *remove-liquidity-token-inner* function

```
(ok (var-set liquidity-token-list (filter remove-filter (var-get
soft-token-list))))))
```

instead of

```
(ok (var-set liquidity-token-list (unwrap-panic (as-max-len? (filter
remove-filter (var-get liquidity-token-list)) u200))))))
```

-For *remove-soft-token-all* function

```
(ok (var-set soft-token-list (list)))
```

instead of

```
(ok (var-set soft-token-list (unwrap-panic (as-max-len?
(var-get empty-token-list) u200))))))
```

-For *remove-poxl-token-all* function

```
(ok (var-set poxl-token-list (list)))
```

instead of

```
(ok (var-set poxl-token-list (unwrap-panic (as-max-len?
(var-get empty-token-list) u200))))))
```

-For *remove-liquidity-token-all* function

```
(ok (var-set liquidity-token-list (list)))
```

instead of

```
(ok (var-set liquidity-token-list (unwrap-panic (as-max-len?
(var-get empty-token-list) u200))))))
```

cryptomate-swap and cryptomate-token contract

In the cryptomate-swap and cryptomate-token contracts use *map-delete* as follows for removing principal from a role instead of *map-set*.

```
(ok (map-delete roles { role: role-to-remove, account: principal-to-remove
})))))
```

In cryptomate-swap contract the following maps should use plain types instead of tuples as follows.

- For the whitelist-for-new-pair map

```
(define-map whitelist-for-new-pair principal bool)
```

instead of

```
(define-map whitelist-for-new-pair {account: principal }
{whitelisted: bool})
```

- For whitelist-for-liquidity-providers map

```
(define-map whitelist-for-liquidity-providers principal bool)
```

instead of

```
(define-map whitelist-for-liquidity-providers {account: principal }
{whitelisted: bool})
```

- For blacklist-for-swapping map

```
(define-map blacklist-for-swapping principal bool)
```

instead of

```
(define-map blacklist-for-swapping {account: principal }
{whitelisted: bool})
```

wstx-token contract

On *get-balance* function of the wstx-token contract unnecessary use of *begin* expression for a function that contains a single expression.

Tests

Tests should use *.expectOk()* or *.expectErr()* for checking errors and the following functions for asserting events on the values returned by the contract calls instead of filtering errors and events manually.

- *.expectSTXTransferEvent()*
- *.expectFungibleTokenTransferEvent()*
- *.expectFungibleTokenMintEvent()*
- *.expectFungibleTokenBurnEvent()*
- *.expectPrintEvent()*

No tests have been written for the *create-pair-new-sip10-token-with-stx* and *create-pair-new-poxl-token-with-stx* function in the cryptomate-one-step-mint contract. These tests were later added to the latest commits and fixed.

2.5 Conclusion

On the audit done by Friedger Müffke, three medium and two minor severity issues were found. Additionally, 14 enhancements and improvements were provided.