HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: TiTi PROTOCOL FOUNDATION LTD
Date: Nov 15th, 2022

Document

Name	Smart Contract Code Review and Security Analysis Report for TiTi PROTOCOL FOUNDATION LTD
Approved By	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU Noah Jelich | Lead Solidity SC Auditor at Hacken OU
Type	ERC20 token; Staking; Swapping; Market Maker; Oracle; Distributor
Platform	EVM
Network	Ethereum, BSC
Language	Solidity
Methods	Manual Review, Automated Review, Architecture Review
Website	https://titi.finance/
Timeline	19.08.2022 - 15.11.2022
Changelog	08.09.2022 - Initial Review 29.09.2022 - Second Review 15.11.2022 - Third Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by TiTi PROTOCOL FOUNDATION LTD (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is smart contracts in the repository:

Initial review scope

Repository:
https://github.com/TiTi-Finance/TiTi-Core-Protocol

Commit:
e59154e9ac97202d72053b8000db6eea10947741

Documentation:
Whitepaper

Technical description

Functional requirements

Integration and Unit Tests: Yes

Contracts:

```
File: ./contracts/controller/ReOrdersController.sol
SHA3: 22601662bd4f278472d35bf3e1486458565ed0fb3e02d6c6e7c2d9536d2af52c

File: ./contracts/gov/TiTIGovernor.sol
SHA3: 7484516df1c706882ead322049d00ba54376e6587371e21fe86729df9c5c9b56

File: ./contracts/gov/TiTiTimelockController.sol
SHA3: c4d91bf701e2154e6ffb7296587789d0f0e5bf10a152431ee3be24aa2a5ede92

File: ./contracts/interfaces/IMAMMSwapPair.sol
SHA3: 3efe7b406f8ce5872ac9ffa86d8cc7236d726041e4e21f8ffd4e1b53aac19307

File: ./contracts/interfaces/IMarketMakerFund.sol
SHA3: 85316d35a3ecf765500325a8a861132ddd2bedf6d66a0ef58b4b1a1a8fd553ae

File: ./contracts/interfaces/IMerkleDistributor.sol
SHA3: 3a2999b8d04f0274e9d0aa2e376ecad5ff6cca4d11c639bb91cdb9e6550bbfe8

File: ./contracts/interfaces/IMMFLPStakingPool.sol
SHA3: 9b7438507abb98c1f106794b7f73ce24f428d89eb1beb779832130af9d3c54c8

File: ./contracts/interfaces/IPausable.sol
SHA3: 202fe19bccdc92f8c7ba82839426e1639748efa654c7056c02cae279fd363ea6

File: ./contracts/interfaces/IReOrdersController.sol
SHA3: 121f0fa0da9f53fabf9951c20243b1d795140ea4d0e4295f09a0e9694bd1c627

File: ./contracts/interfaces/ITiTiStaking.sol
SHA3: 9d0b82d49513c8b71baad9ac8106b9256e2c8b9fe19f4a7a07186da2c02ae053

File: ./contracts/interfaces/ITiUSDToken.sol
```

```
SHA3: 2f5947ed8915bf75a1e62b96b4e01b28c391af68083c43861af65f9856cf1c64

File: ./contracts/libraries/Babylonian.sol
SHA3: 38617315cf7bfef8e925b1b40e39a66fae75492e1040a48c183a1986ac3608a5

File: ./contracts/libraries/FixedPoint.sol
SHA3: 28f7f4d63077c6c2fd4d7a66965bfea5a27fc4903f3970264861ec3fcc57e5b5

File: ./contracts/libraries/UQ112x112.sol
SHA3: b958f933896e9340cdfcc63e3ee5b99e7594772f12688b1286b8d9ff364148a2

File: ./contracts/mamm/MAMMSwapPair.sol
SHA3: 6e313c91cf5c3b8002c804f625069f3b96011307984f05c69b52e0da7b7d6d43

File: ./contracts/Migrations.sol
SHA3: f38ad4185f0fa410f3427a0bae9195f29bf1c8806f1a019cc727d7c39b53811d

File: ./contracts/mmf/MarketMakerFund.sol
SHA3: 799f4c4ea20cb5ec38dd1bb9297c09362fe63132700faa0736d7eeb0e03a263f

File: ./contracts/mmf/MMFLPStakingPool.sol
SHA3: 9da7cb80b481c8b32933014209828adee1aab14aba6d8e4b063feb2b33754ecd

File: ./contracts/mock/MockTiTiStakingV2.sol
SHA3: 793d49bd0683657bc7e9bdb26586a2fd4e862e00fb4bddfa66f751f8f95076f6

File: ./contracts/mock/MockUSDC.sol
SHA3: f88d15af39a29cbe373fdd61c8f6eacc7b51b28611874fa857cdd5891bffc931

File: ./contracts/oracles/TiTiOracles.sol
SHA3: 379fc22bf7c8071bc31b310e7a23193701342bada57158ca8571a65c7c52f01b

File: ./contracts/staking/TiTiStakingV1.sol
SHA3: 20958fc9c2a79f70d4d0e162c7b3359cfcf9645bff1be9d99b442162792eb449

File: ./contracts/TiTiToken.sol
SHA3: 1298e44d723af449853580d7a56e3740ca83db3c3056b0683e74bb4b68f239c9

File: ./contracts/TiUSDToken.sol
SHA3: 08806f3ddd6e40390d09068b31de49dac6b854ee1230f9fe9304442b843d478f

File: ./contracts/use2earn/TiTiMerkleDistributor.sol
SHA3: 906bd860b91f82c2f48c9a2c8b123716f4d25c9c924ae1bcadc5137bd7bd9c24

File: ./contracts/vault/BaseVault.sol
SHA3: 3853d136c6506f68a1baa59237a596e7fa02c3dd0b22861f42aeef7415b3b624

File: ./contracts/vault/ProtocolFeeVault.sol
SHA3: 465fe7fc51d3853565773fb8d5bda3b839e2ac0d7f7fc1e68e7ea285255bff5b

File: ./contracts/vault/RainyDayFundVault.sol
SHA3: 009f52383c8681a6f3f7ce1fb1989e753c1319746a5ef3064737c8997d1ed6b4
```

Second review scope

Repository:

https://github.com/TiTi-Finance/TiTi-Core-Protocol

Commit:

cb75faf4f179893c333348e4562dc3bd01dd7178

Documentation:

Whitepaper

Technical description

Functional requirements

Integration and Unit Tests: Yes
Contracts:

```
File: ./contracts/controller/ReOrdersController.sol
SHA3: 3b27225e8c09935ce01227a702bef1cd94c091df6b7dc574dd3aca8b11044f1f

File: ./contracts/gov/TiTIGovernor.sol
SHA3: 404054f0962e229a1afe5db7cb9f6cf33c3a1737ca1147b65fa31f0316988754

File: ./contracts/gov/TiTiTimelockController.sol
SHA3: 938f8a9820eb9f699f308f457cdd0f86475eadcdfc240f918670d508dc8802e4

File: ./contracts/interfaces/IMAMMSwapPair.sol
SHA3: ee528989d9147672f77775cf28d753a7d59c393d6766cf7c6bdeba0e9eab3bab

File: ./contracts/interfaces/IMarketMakerFund.sol
SHA3: cc3b8ebe64c18228e4451d971d4ad0bc788cf80935de39d9a1bb49345b69595a

File: ./contracts/interfaces/IMerkleDistributor.sol
SHA3: c846f2af146dcf68e5127ede8ee577eaa2fbc9926b7c9395ab104329c4c09db4

File: ./contracts/interfaces/IMMFLPStakingPool.sol
SHA3: 51768ac3c949f9787c596dc80a6e93bf6b4f2e1de7648c16b794113c87b1082c

File: ./contracts/interfaces/IPausable.sol
SHA3: 3c8bd1a68e4bd716583aa90d7354764593fb262f9d676f15accd4d9e2c3141c7

File: ./contracts/interfaces/IReOrdersController.sol
SHA3: 62305a536cdfdbec80ce77b61b8e25115c58c4bf5189da15faaddce8f5be902d

File: ./contracts/interfaces/ITiTiStaking.sol
SHA3: 38f819feec079c17ef571358cb0fda0e4db0087dc845f4bca1e30a38b4dd6400

File: ./contracts/interfaces/ITiUSDToken.sol
SHA3: fb9d17fd0a27352474bc51a05463cfb35b981928811836a740eec8fc7e52cab0

File: ./contracts/libraries/Babylonian.sol
SHA3: 12cf2ac7e16ef8e766e8ffa7a83cbdf85a0f6012a263309a262daf2463901292

File: ./contracts/libraries/FixedPoint.sol
SHA3: 9612a1aee0ab3f50931a4501cb8636dec73254f9f8b92dc2c2793f0e75adaf1c

File: ./contracts/libraries/UQ112x112.sol
SHA3: fcc177ac2ad978b1b5255987600efecb59ff6ce71fa6c1c2cd2a7ff0b189c792

File: ./contracts/mamm/MAMMSwapPair.sol
SHA3: 1837b4a16950ed03a26984e076695e4364a2c6d73ab9f8ad7ad521aeb6d09ee9

File: ./contracts/Migrations.sol
SHA3: f38ad4185f0fa410f3427a0bae9195f29bf1c8806f1a019cc727d7c39b53811d

File: ./contracts/mmf/MarketMakerFund.sol
SHA3: 4aa81649191b96da58775011f8cf3e5a92b8fbef2fdeb64df1f0b32db45f0d49

File: ./contracts/mmf/MMFLPStakingPool.sol
SHA3: 2cacb93a25199a9148a52dbf671f456c4ef429249c2d139b3f45692c0526dcbc
```

```
File: ./contracts/mock/MockTiTiStakingV2.sol
SHA3: b4f99fcf416eb600f97ef78e4b98888ba21c6ffe28e045b6bf99a0bd29717416

File: ./contracts/mock/MockUSDC.sol
SHA3: dde25c1416e14227b3a43886e1e43407f46f198b247bcad38e11b524906839a1

File: ./contracts/oracles/TiTiOracles.sol
SHA3: 560d0a4a19b5fb6928789da79fea9f6a7941177733d391ed38110910b012bfc9

File: ./contracts/staking/TiTiStakingV1.sol
SHA3: 446e16e653cb51a72a5b4d409a360265b976c96f52f5ce3b2b821e96d9023054

File: ./contracts/TiTiToken.sol
SHA3: b9119810ac2bdbe298179c5b40dd4651102aaa6fc05a11b108b6df252df564a7

File: ./contracts/TiUSDToken.sol
SHA3: bdcece5593ecc87f2dd48def7181c4b6216403a43b5fadf087f7cdf7772f28b4

File: ./contracts/use2earn/TiTiMerkleDistributor.sol
SHA3: 6557de6dee48b452bf9f074da3f4848a932376fd945a533cc0e20ebf4bc20092

File: ./contracts/vault/BaseVault.sol
SHA3: 71cd7c0f69a52f41d9b419aa49bd7386bf08fe18de8ee68946a104a017dfbe2c

File: ./contracts/vault/ProtocolFeeVault.sol
SHA3: 4aca612d9cd3a41c006e896271ebd8840dce94a609262ba6651d32a65ffed311

File: ./contracts/vault/RainyDayFundVault.sol
SHA3: 60587391923a41b20e7da79c004830a014b7883cbbea4aa1bb09e3404ea3d221
```

Second review scope

Repository:

https://github.com/TiTi-Finance/TiTi-Core-Protocol

Commit:

190dd0d2417a6b382cccf97de199faca6efd11bc

Documentation:

Whitepaper

Technical description

Functional requirements

Integration and Unit Tests: Yes

Contracts:

```
File: ./contracts/controller/ReOrdersController.sol
SHA3: 2076b8b442d3c00fcfd098fa5906ec068235f8c1876935a52b45f9617452311e

File: ./contracts/gov/TiTIGovernor.sol
SHA3: 404054f0962e229a1afe5db7cb9f6cf33c3a1737ca1147b65fa31f0316988754

File: ./contracts/gov/TiTiTimelockController.sol
SHA3: e40eba18b4f1527f8ec170419e0ab786aad066d10b5e00ce734175e1619c305f

File: ./contracts/interfaces/IMAMMSwapPair.sol
SHA3: ee528989d9147672f77775cf28d753a7d59c393d6766cf7c6bdeba0e9eab3bab

File: ./contracts/interfaces/IMarketMakerFund.sol
SHA3: cc3b8ebe64c18228e4451d971d4ad0bc788cf80935de39d9a1bb49345b69595a
```

```
File: ./contracts/interfaces/IMerkleDistributor.sol
SHA3: c846f2af146dcf68e5127ede8ee577eaa2fbc9926b7c9395ab104329c4c09db4

File: ./contracts/interfaces/IMMFLPStakingPool.sol
SHA3: 51768ac3c949f9787c596dc80a6e93bf6b4f2e1de7648c16b794113c87b1082c

File: ./contracts/interfaces/IPausable.sol
SHA3: 3c8bd1a68e4bd716583aa90d7354764593fb262f9d676f15accd4d9e2c3141c7

File: ./contracts/interfaces/IReOrdersController.sol
SHA3: b7ce42efe77185f2b5a07883283f03d7f85a999f9796eb2b27dfe09cab324ac2

File: ./contracts/interfaces/ITiTiStaking.sol
SHA3: 38f819feec079c17ef571358cb0fda0e4db0087dc845f4bca1e30a38b4dd6400

File: ./contracts/interfaces/ITiUSDToken.sol
SHA3: fb9d17fd0a27352474bc51a05463cfb35b981928811836a740eec8fc7e52cab0

File: ./contracts/libraries/Babylonian.sol
SHA3: 12cf2ac7e16ef8e766e8ffa7a83cbdf85a0f6012a263309a262daf2463901292

File: ./contracts/libraries/FixedPoint.sol
SHA3: 9612a1aee0ab3f50931a4501cb8636dec73254f9f8b92dc2c2793f0e75adaf1c

File: ./contracts/libraries/UQ112x112.sol
SHA3: fcc177ac2ad978b1b5255987600efecb59ff6ce71fa6c1c2cd2a7ff0b189c792

File: ./contracts/mamm/MAMMSwapPair.sol
SHA3: cc40a4cdcae2ae8c6d81946c8571203d470068ba3ff8b6dbdc0066861f262386

File: ./contracts/Migrations.sol
SHA3: f38ad4185f0fa410f3427a0bae9195f29bf1c8806f1a019cc727d7c39b53811d

File: ./contracts/mmf/MarketMakerFund.sol
SHA3: 4fb0d0ab51a692c9c1aa6ced9dc000c02f8ed94f8272a1d2d65012169601169e

File: ./contracts/mmf/MMFLPStakingPool.sol
SHA3: 2cacb93a25199a9148a52dbf671f456c4ef429249c2d139b3f45692c0526dcbc

File: ./contracts/mock/MockTiTiStakingV2.sol
SHA3: b4f99fcf416eb600f97ef78e4b98888ba21c6ffe28e045b6bf99a0bd29717416

File: ./contracts/mock/MockUSDC.sol
SHA3: dde25c1416e14227b3a43886e1e43407f46f198b247bcad38e11b524906839a1

File: ./contracts/oracles/TiTiOracles.sol
SHA3: 490b77e72c9ea30787ad523fc0068c51823232c22a551104dacd33fe76ce1099

File: ./contracts/staking/TiTiStakingV1.sol
SHA3: 277c04d3c0a5b009b11fd27b30569754d123454bc7ad6cf47e764a44870bdf28

File: ./contracts/TiTiToken.sol
SHA3: aaa87df0ad1cb7717a8b19b9f1290a12f764cc9f95a13db81d494ef377f6cb00

File: ./contracts/TiUSDToken.sol
SHA3: bdcece5593ecc87f2dd48def7181c4b6216403a43b5fadf087f7cdf7772f28b4

File: ./contracts/use2earn/TiTiMerkleDistributor.sol
SHA3: 8d9a9f802b86981a47e95373ad99a14065d601022e42f94b9d370ca4ef39df05
```

```
      File: ./contracts/vault/BaseVault.sol
      SHA3: 71cd7c0f69a52f41d9b419aa49bd7386bf08fe18de8ee68946a104a017dfbe2c

      File: ./contracts/vault/ProtocolFeeVault.sol
      SHA3: 44a5ed1d32b84754f3298cdd549fa26df18f43f1b944ee362c5780c386fc0fde

      File: ./contracts/vault/RainyDayFundVault.sol
      SHA3: b5ea11eaf08c7f2f58a2a9bf82cc279b8c5d87da3961cffe41d90c4903b61176
```

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**. Functional and technical requirements are provided.

Code quality

The total Code Quality score is **5** out of **10**. Style guide violation was found. The development environment was not well built and did not have instructions for tests and deployment.

Test coverage

Deployment and basic user interactions are partly covered with tests. Negative cases coverage is missed, and most test scenarios are not built.
Test coverage of the project is 79.34%.

Security score

As a result of the audit, the code contains **1** high, **2** low severity issues. The security score is **5** out of **10**.

All found issues are displayed in the "Findings" section.

Summary

According to the assessment, the Customer's smart contract has the following score: **6**.

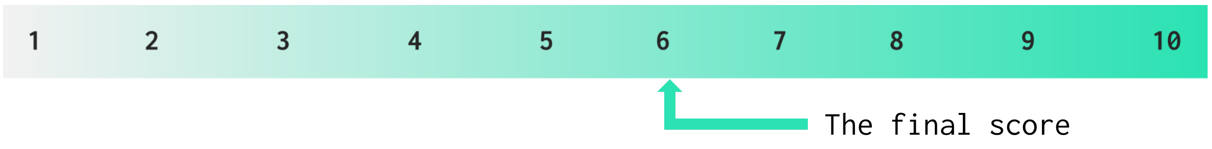

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
30 August 2022	9	5	5	3
26 September 2022	2	0	1	0
15 November2022	2	0	1	0

Checked Items

We have audited provided smart contracts for commonly known and more specific vulnerabilities. Here are some of the items that are considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Passed
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless it is required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization	SWC-115	tx.origin should not be used for	Passed

through tx.origin		authorization.	
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifier should always be used. All parameters from the signature should be used in signer recovery	Not Relevant
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of unused variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP standards violation	EIP	EIP standards should not be violated.	Passed
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of	Passed

		data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	
Style guide violation	**Custom**	Style guides and best practices should be followed.	Failed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Failed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Passed
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed
Stable Imports	**Custom**	The code should not reference draft contracts, that may be changed in the future.	Passed

System Overview

TiTi Protocol is a decentralized stable token system with the following contracts:

- *TiUSDToken* – is a decentralized stable coin with mint, burn, snapshot, and permit mechanisms.
  It has the following attributes:
  - Name: TiUSDToken
  - Symbol: TiUSD
  - Decimals: 18
  - Total supply: Not fixed
- TiTiToken - is a governance token for TiTi Protocol with mint, burn, snapshot, and vote mechanisms.
  - Name: TiTiToken
  - Symbol: TiTi
  - Decimals 18
  - Total supply: Not fixed
- BaseVault - is a contract designed to deposit and withdraw ERC20 tokens.
- ProtocolFeeVault - is a contract designed to deposit and withdraw ERC20 tokens.
- RainyDayFundVault - is a contract designed to deposit and withdraw ERC20 tokens.
- TiTiMerkleDistributor - is the Use-To-Earn reward distributor of TiTi Protocol.
- TiTiStakingV1 - is the staking module of the TiTi Protocol.
- TiTiOracles - is a price oracle.
- MMFLPStakingPool - is a pool to stake lp tokens earned from USDC-TiUSD pair.
- MarketMakerFund - is the market maker fund contract of the TiTi Protocol.
- MAMMSwapPair - is the Monopoly Market Maker contract of the TiTi Protocol.
- TiTiGovernor - the Governance Module of TiTi Protocol. Implements TiTi Protocol DAO.
- TiTiTimeLockController - the TimelockController module of TiTi Protocol. This module is used to manage the timelock logic in the protocol.
- ReOrdersController - the ReOrders control module of TiTi Protocol. This module implements and manages the ReOrders function.

Privileged roles

- SNAPSHOT_ROLE of TiUSDToken can
  - take a snapshot
- MINTER_ROLE of the TiUSDToken can

- mint TiUSDToken
- reorders TiUSDToken

- DEFAULT_ADMIN_ROLE of the TiUSDToken can
  - set new admin
- SNAPSHOT_ROLE of TiTiToken can
  - take a snapshot
- MINTER_ROLE of the TiTiToken can
  - mint TiUSDToken
- DEFAULT_ADMIN_ROLE of the TiTiToken can
  - set new admin
  - set new minters
- Owner of BaseVault can
  - withdraw tokens from contract
- Owner of ProtocolFeeVault can
  - withdraw tokens from contract
- Owner of RainyDayFundVault can
  - withdraw tokens from contract
- Owner of TiTiMerkleDistributor can
  - update MerkleRoot
  - update staking contract address
  - deposit tokens to contract
  - extract tokens from contract
- Governor role of TiTiStakingV1 can
  - set workers
  - set pending governor
  - set merkle root
  - skim
- Worker role of TiTiStakingV1 can
  - add reward to the contract
- MMF role of MMFLPStakinPool can
  - stake
  - withdraw
  - get reward
- Owner of MMFLPStakingPool can
  - recover unsupported tokens
  - set reward distribution contract
- Owner of MarketMakerFund can
  - set new MAMM contract
  - set new reorders controller contract
  - set new LP staking pool contract
  - pause contract
  - unpause contract
- Owner of MAMMSwapPair can
  - set new reorders controller contract

    - set fee to address
    - set new MMF contract
    - set period time
    - set is allowed contracts call
    - pause
    - unpause
- Owner of ReOrdersController can
    - set new MAMM contract address
    - set new MMF contract address
    - set new price delta
    - set new duration
    - set new allocations
    - pause
    - unpause

Findings

■■■■ Critical

1. Access Control Violation

onlyGov privileged users can withdraw as many previously staked TiTi tokens as 1 token left in the contract.

This can lead to loss of staked funds of users.

Path: ./contracts/staking/TiTiStakingV1.sol: extract()

Recommendation: Remove the function or mention it in the documentation.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

2. Invalid Calculations

During adding liquidity, TiUSD token is minted to MAMMSwapPair contract, and USDC is sent from the user to MMSwapPair. However, while removing the liquidity, MarketMakerFund's all balance is burned, and all USDC tokens that are in the contract are transferred from the user to MarketMakerFund.

This can break the whole system, and users lose their funds.

Path:./contracts/mmf/MarketMakerFund.sol : removeLiquidity(), withdrawAll()

Recommendation: When removing the liquidity, burn and transfer the required amount instead of all contract balance.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

■■■ High

1. Requirements Violation

Users cannot claim a reward if the requested amount is less than the previously claimed amount.

Path: ./contracts/use2earn/TiTiMerkleDistributor.sol:claimAndStake(), claim()

Recommendation: Instead of storing the last claimed reward amount, use claimed bit map for each root to store claimed rewards.

Status: Mitigated (with Customer notice) (Revised commit: 190dd0d2417a6b382cccf97de199faca6efd11bc)

2. Requirements Violation

UNBONDING_DURATION of TiTiStakingV1 declared as 30 days in the contract. However, it is stated as 7 days in the documentation.

Path: ./contracts/staking/TiTiStakingV1.sol

Recommendation: Change UNBONDING_DURATION to 7 days in TiTiStakingV1 Contract.

Status: Mitigated (with Customer notice) (Revised commit: 190dd0d2417a6b382cccf97de199faca6efd11bc)

3. **Invalid Calculations**

pavAmount is calculated according to the following formula; ∆PAV_{n} = (X_{n} - X_{n-1}) - (Y_{n-1} - Y_{n}) * PegPrice_{n}. So, in the _reorders function, the result of (∆B - ∆A) is assigned to the pavAmount variable.

This calculation can result in a negative value, and on line 172, the conversion of value to uint256 with toUint256 function is aimed. However, the function accepts only the positive variables.

This may lead failing of transaction and may disrupt the entire system operation.

Path: ./contracts/controller/ReOrdersController.sol: _reorders()

Recommendation: Consider negative result in calculation.

Status: Mitigated (with Customer notice) (Revised commit: 190dd0d2417a6b382cccf97de199faca6efd11bc)

4. **Access Control Violation**

mmf address can be set, and the owner has that ability. The owner can call the *removeLiquidity* function after changing the *mmf* address and deprive the *MAMMSwapPair* contract of funds.

Path: ./contracts/controller/ReOrdersController.sol

Recommendation: Do not allow setting the *mmf* address.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

5. **Token Supply Manipulation**

According to documentation, the token supply should be fixed. However, the contracts allow unlimited minting.

Path: ./contracts/TiTiToken.sol: mint()

Recommendation: Limit token supply or update the documentation.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

6. **Missing Functionality**

TiTiOracles contract is pausable, but it has no implemented function to pause/unpause it.

Path: ./contracts/controller/TiTiOracles.sol

Recommendation: Write functions that call _pause and _unpause functions from OpenZeppelin's Pausable contract.

Status: Fixed (Revised commit: f6abdfa19f0f6d7a3c77bd75a5559effe57cd2c6)

7. **Secure Oracles Usage**

TiTiOracles contract does not have a pausing mechanism.

Path: ./contracts/controller/TiTiOracles.sol

Recommendation: Build a pausing system for the case of oracles corruption.

Status: New

■■ Medium

1. **Redundant Variable**

isToken0 variable in *ReOrdersController* contract is declared as immutable. The variable with the same name in *MAMMSwapPair*, *MarketMakerFund*, *TiTiOracles* contracts is declared as public, but there is no implementation to set/adjust it later. So, this hardcoded redundant value causes complexity in the code.

Paths: ./contracts/controller/ReOrdersController.sol, ./contracts/interfaces/IMAMMSwapPair.sol, ./contracts/mamm/MAMMSwapPair.sol, ./contracts/mmf/MarketMakerFund.sol, ./contracts/oracles/TiTiOracles.sol

Recommendation: Remove the redundant variable and edit the if-else statements according to that change.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

2. **Checks-Effects-Interactions Pattern Violation**

As a best practice, always allow making internal state changes after completing the transferring tokens from the user's address.

Path: ./contracts/mmf/MMFLPStakingPool.sol LPTokenWrapper.stake(), LPTokenWrapper.withdraw()

Recommendation: Firstly, execute the *safeTransferFrom* function, then make internal state changes.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

3. **Checks-Effects-Interactions Pattern Violation**

In the function, firstly, the *totalTiTi* should have been checked right after the totalTiTi is calculated.

Following the Checks-Effects-Interactions pattern is always best practice.

Path: ./contracts/staking/TiTiStakingV1.sol: withdraw()

Recommendation: Execute require statement right after the *totalTiTi* calculation.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

4. **Checks-Effects-Interactions Pattern Violation**

In the function, firstly, the contract balance should have been checked, and then the transfer should be executed.

Following the Checks-Effects-Interactions pattern is always best practice.

Path: ./contracts/staking/TiTiStakingV1.sol: skim()

Recommendation: Change the order of two lines.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

5. **Checks-Effects-Interactions Pattern Violation**

In the function, firstly, the *totalTiTi* should have been checked, and then the transfer should be executed.

Following the Checks-Effects-Interactions pattern is always best practice.

Path: ./contracts/staking/TiTiStakingV1.sol: extract()

Recommendation: Change the order of two lines 257 and 258.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

■ Low

1. **Floating Pragma**

Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Paths: all

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

2. Functions that Can Be Declared External

In order to save Gas, public functions that are never called in the contract should be declared as external.

Path: ./contracts/mamm/MAMMSwapPair.sol: getMMFFunds()

Recommendation: Use the external attribute for functions never called from the contract.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

3. Redundant Import

The use of unnecessary imports will increase the Gas consumption of the code. Thus they should be removed from the code.

IERC20 is redundant for the BaseVault, TiTiMerkleDistributor, MMFLPStakingPool and MAMMSwapPair.

Initializable is redundant for the TiTiMerkleDistributor.

IERC20Upgradeable is redundant for the TiTiStaking

IMAMMSwapPair is redundant for the MAMMSwapPair

SafeERC20 is redundant for the ReOrdersController

Paths: ./contracts/vault/BaseVault.sol

./contracts/use2earn/TiTiMerkleDistributor.sol

./contracts/staking/TiTiStakingV1.sol

./contracts/mmf/MMFLPStakingPool.sol

./contracts/mamm/MAMMSwapPair.sol

./contracts/controller/ReOrdersController.sol

Recommendation: Remove the redundant import.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

4. Unfinalized Functionality

On line 117, reorders function explanation demonstrates that the functionality is not finalized. This makes the project look uncompleted.

The whole functionality must be settled, and there must not be doubts written in the repository’s comment lines.

Paths: all

Recommendation: Finalize the code and remove the parts/comments that are not concluded.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

5. **Redundant Code Part**

updateReward modifier declares lastUpdateTime variable redundantly because it is never used in the function.

Redundant declarations cause more Gas consumption and increase the code complexity.

Path: ./contracts/mmf/MMFLPStakingPool.sol

Recommendation: Remove the redundant line (Line#112).

Status: Mitigated (with Customer notice) (Revised commit: 190dd0d2417a6b382cccf97de199faca6efd11bc)

6. **Missing Event Emitting**

setRewardDistribution() should emit an event for updating the variable.

Path: ./contracts/mmf/MMFLPStakingPool.sol : IRewardDistributionRecipient.setRewardDistribution()

Recommendation: Emit the necessary event in the function.

Status: Fixed (cb75faf4f179893c333348e4562dc3bd01dd7178)

7. **Style Guide Violation**

The provided contracts should follow the official guidelines to provide consistency.

Paths: all

Recommendation: Follow the official style guide: https://docs.soliditylang.org/en/v0.8.13/style-guide.html

Status: Reported (Revised commit: 190dd0d2417a6b382cccf97de199faca6efd11bc)

8. **Missing Event Emitting**

acceptGovernor function should emit an event for assigning the new governor address.

Path: ./contracts/staking/TiTiStakingV1.sol

Recommendation: Write an event and emit it in the function.

Status: Fixed (Revised commit: cb75faf4f179893c333348e4562dc3bd01dd7178)

9. **Missing Zero Address Validation**

Address parameters are being used without checking against the possibility of 0x0.

This can lead to unwanted external calls to 0x0.

Paths: ./contracts/use2earn/TiTiMerkleDistributor.sol: _updateStaking()

./contracts/staking/TiTiStakingV1.sol: initialize()

./contracts/controller/ReOrdersController.sol: constructor()

./contracts/mamm/MAMMSwapPair.sol: constructor()

./contracts/TiUSDToken.sol: setNewAdmin()

Recommendation: Implement zero address checks.

Status: Reported (Revised commit: 190dd0d2417a6b382cccf97de199faca6efd11bc)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in 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).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted to and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

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 recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

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 vulnerabilities that can lead to hacks. Thus, Consultant cannot guarantee the explicit security of the audited smart contracts.