HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: G21 CAPITAL MARKETING
Date: October 12th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for G21 CAPITAL MARKETING
Approved By	Noah Jelich | Senior Solidity SC Auditor at Hacken OU
Type	ERC20 token; Lending
Platform	EVM
Network	Ethereum, GTON Network
Language	Solidity
Methods	Manual Review, Automated Review, Architecture Review
Website	-
Timeline	11.08.2022 - 12.10.2022
Changelog	26.08.2022 - Initial Review 12.10.2022 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Introduction **4**

Scope **4**

Severity Definitions **7**

Executive Summary **8**

Checked Items **9**

System Overview **12**

Findings **17**

Disclaimers **22**

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by G21 Capital Marketing (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/GTON-capital/gcd-protocol
https://github.com/GTON-capital/gcd-oracles
Commit:
4846d084bd8ab53b97abfe65f92436ffafc828f3
d65e130acb0a62838d3795e924ecd0294cf79b8e
Documentation: No

Integration and Unit Tests: Yes
Deployed Contracts Addresses: No
Contracts:

File: ./contracts/auction/LiquidationAuction02.sol
SHA3: 5c0399e60ca885dcd795f36e1f7206a27367464e7170fcd88b7f3e9d300d0fc2

File: ./contracts/CDPRegistry.sol
SHA3: 5305efd5dc5e040f4f46654aa9355d7f69c4610d8608e920866befc7f2cefbfc

File: ./contracts/CollateralRegistry.sol
SHA3: 46985040c3689251ae2015dbf40a8f9537b671ec6320f855aef6f29eb90395e6

File: ./contracts/GCD.sol
SHA3: 5974f1626f277d77571a60240aa5ca2754455e436053c316b9f2f12950945ac0

File: ./contracts/helpers/ReentrancyGuard.sol
SHA3: afad31fbaa9f204dac95f8786bc52dfd2635dd4d663f70045308a0ea31e9c778

File: ./contracts/helpers/TransferHelper.sol
SHA3: 2a73e10319d9f00d57f962f863a396ed5331406b169db52ded231041091c618e

File: ./contracts/interfaces/IAggregator.sol
SHA3: 3c6cdb1afee1019b6b23ee5ebb84f33c5202ba1ccac8b3586fb808cdc3c0c0a0

File: ./contracts/interfaces/ICDPRegistry.sol
SHA3: 9620b8d235994a831d812ad6aa73b18604cb8113d1c381bcce2605b9cda82899

File: ./contracts/interfaces/IForceTransferAssetStore.sol
SHA3: b08b7dfe6dc40b4e5f33f37d25f18600c24c0e51cfca6789b57da1de054adde4

File: ./contracts/interfaces/IFoundation.sol
SHA3: 7003220bf96fef2265d63e9d3cf0044a040a4c158b396f91150d9d759d27a9b5

File: ./contracts/interfaces/IOracleEth.sol
SHA3: 8100397696a654cbef3fed2771152b6acfc1cab6e0b38834b435403f3927d2f4

```
File: ./contracts/interfaces/IOracleRegistry.sol
SHA3: 1a0a7d74cf0e0601776ae963ac5518585281176c41c9167c035d6702d63b9097

File: ./contracts/interfaces/IOracleUsd.sol
SHA3: af1f1e78dbf1e020a4e27af49ceffccdec279d06f981ef6a5e8e4dd06bbcd182

File: ./contracts/interfaces/IToken.sol
SHA3: 806df0506701e04983a16f4edd6752399d1beb44a08f44e92f962fb160101ccb

File: ./contracts/interfaces/IVault.sol
SHA3: 13c1cef89dba047350e1eb0875711cbeea841e3ef91af4e7fb035c358a373966

File: ./contracts/interfaces/IVaultManagerParameters.sol
SHA3: 055b4db613c79c64b47930ab37cd4d682326c310fe45c17603b6b38995df19ff

File: ./contracts/interfaces/IVaultParameters.sol
SHA3: d5fbf3f08afd317ed8573d2392109de6345a8427e20c9564ca0df8ae9d945449

File: ./contracts/interfaces/IWETH.sol
SHA3: 3865222c4ff2eb8d605178339d19987a93cbf6abee6e830d68971fadf5531ef2

File: ./contracts/interfaces/IWrappedToUnderlyingOracle.sol
SHA3: 0df9f79c7313872a44156dc98bb30276a434285f34c29a1f53f825c26310781c

File: ./contracts/oracles/ChainlinkedOracleMainAsset.sol
SHA3: b68074cb85c192fd7aa8a2544fcd2f7b83b2a89c12c242d7b68be30386059f62

File: ./contracts/oracles/OracleRegistry.sol
SHA3: eb9d36df96f86b61b45a4ab51307a22d36c7dfcca295bb0a8cbf8e41471ff9fb

File: ./contracts/oracles/WrappedToUnderlyingOracle.sol
SHA3: 9a3494d258407efda85760d2308dec5954ff4d8f7f99d6b3ca8eb54b474d694b

File: ./contracts/test-helpers/WETH.sol
SHA3: d7ee14a4ab65418637e0d193606ee6b3035936ae9394c468c850f345089dd151

File: ./contracts/vault-managers/CDPManager01.sol
SHA3: efafa6da2164080489390dd100eef22a245abba5b728a27acd2bcff40de14a21

File: ./contracts/vault-managers/VaultManagerParameters.sol
SHA3: c645494a7adaa5d1e7328585e759bba5661cdadaf9f486e8240759383ff88efc

File: ./contracts/Vault.sol
SHA3: 0decbbf022e7776aafe5631892c403a4fe1f82a67e15c6f1b12fcb0274ad2dca

File: ./contracts/VaultParameters.sol
SHA3: d75accd4e9a4b292b9611ea7d4191a108c1ecbf27c17b5bd17553820068d61a1

File: ./contracts/helpers/IOracleRegistry.sol
SHA3: 1b265fad79101bb00d26133c7d8afc443da32b3624d3f7cdf0c78e208fba548d

File: ./contracts/helpers/IOracleUsd.sol
SHA3: 56c2b50696fca1128a2c4a5d7134d4eef0d82be665cb56f4a00d1e2b395f25d5

File: ./contracts/helpers/IVaultParameters.sol
SHA3: bb42fc6d89fd9e96c5187c275fea71bd835f47b38999aab76b82c9b7d7815a27

File: ./contracts/helpers/SafeMath.sol
SHA3: 932f79fc33490368f135c12b46329ac9a13ccd41163b9db5f1f244a66198c93e
```

File: ./contracts/impl/UniswapV3Oracle.sol
SHA3: 9beb6e4d633eb1e0749a20c89ae789a1439d8fcec791314d70c6615d270ad214

File: ./contracts/impl/uniswapV3Oracle/FullMath.sol
SHA3: 7857529e876071e3026c9383b78d5e2d8ce45f832b52c80ea88609457047d2d7

File: ./contracts/impl/uniswapV3Oracle/IUniswapV3Pool.sol
SHA3: f1225a22d827df6ac0de27bf8f7f30dfd32c01c22b62b110345362237c1d3e2e

File: ./contracts/impl/uniswapV3Oracle/IUniswapV3PoolDerivedState.sol
SHA3: 0bc5b04159a7fb067d8827545a982adf1a7da9b732624cfabb496615c9f993a2

File: ./contracts/impl/uniswapV3Oracle/LowGasSafeMath.sol
SHA3: 8efb50f75409d86b20ec10664f200a19149b640ae319b18970c9777fa635ed26

File: ./contracts/impl/uniswapV3Oracle/OracleLibrary.sol
SHA3: 2d1296083f65da13e2c98c1ff88b5e63f93967dc1221404736d709acc82b7aab

File: ./contracts/impl/uniswapV3Oracle/PoolAddress.sol
SHA3: efc47967301e46c120a2c4425212fa6d66d27a7e9203febf3d463fec81f39001

File: ./contracts/impl/uniswapV3Oracle/TickMath.sol
SHA3: c35e2b25d8315f9d943119b58cd5061afb54f8633cc28b46b827b81f84d2f4de

File: ./contracts/interface/IOracleRegistry.sol
SHA3: 18c6f9c801d65b9b1c72115b8d574969be12abef1677220d809120be370f73b1

File: ./contracts/interface/IOracleUsd.sol
SHA3: 3cd52c8f5fa39215243dce639a33c5d9d6448fa5081be591e29c9267453828ec

File: ./contracts/interface/IVaultParameters.sol
SHA3: 8c6e6adb471086132c1743756384f763291dfc5828ff779693881373bafbbfb0

Second review scope

Repository:
https://github.com/GTON-capital/gcd-protocol
https://github.com/GTON-capital/gcd-oracles

Commit:
4846d084bd8ab53b97abfe65f92436ffafc828f3
d65e130acb0a62838d3795e924ecd0294cf79b8e

Documentation:
Technical Description: Link

Integration and Unit Tests: Yes
Deployed Contracts Addresses: No
Contracts:

File: ./contracts/auction/LiquidationAuction02.sol
SHA3: 5c0399e60ca885dcd795f36e1f7206a27367464e7170fcd88b7f3e9d300d0fc2

File: ./contracts/CDPRegistry.sol
SHA3: 5305efd5dc5e040f4f46654aa9355d7f69c4610d8608e920866befc7f2cefbfc

File: ./contracts/CollateralRegistry.sol
SHA3: 46985040c3689251ae2015dbf40a8f9537b671ec6320f855aef6f29eb90395e6

File: ./contracts/GCD.sol
SHA3: 5974f1626f277d77571a60240aa5ca2754455e436053c316b9f2f12950945ac0

```
File: ./contracts/helpers/ReentrancyGuard.sol
SHA3: afad31fbaa9f204dac95f8786bc52dfd2635dd4d663f70045308a0ea31e9c778

File: ./contracts/helpers/TransferHelper.sol
SHA3: 2a73e10319d9f00d57f962f863a396ed5331406b169db52ded231041091c618e

File: ./contracts/interfaces/IAggregator.sol
SHA3: 3c6cdb1afee1019b6b23ee5ebb84f33c5202ba1ccac8b3586fb808cdc3c0c0a0

File: ./contracts/interfaces/ICDPRegistry.sol
SHA3: 9620b8d235994a831d812ad6aa73b18604cb8113d1c381bcce2605b9cda82899

File: ./contracts/interfaces/IForceTransferAssetStore.sol
SHA3: b08b7dfe6dc40b4e5f33f37d25f18600c24c0e51cfca6789b57da1de054adde4

File: ./contracts/interfaces/IFoundation.sol
SHA3: 7003220bf96fef2265d63e9d3cf0044a040a4c158b396f91150d9d759d27a9b5

File: ./contracts/interfaces/IOracleEth.sol
SHA3: 8100397696a654cbef3fed2771152b6acfc1cab6e0b38834b435403f3927d2f4

File: ./contracts/interfaces/IOracleRegistry.sol
SHA3: 1a0a7d74cf0e0601776ae963ac5518585281176c41c9167c035d6702d63b9097

File: ./contracts/interfaces/IOracleUsd.sol
SHA3: af1f1e78dbf1e020a4e27af49ceffccdec279d06f981ef6a5e8e4dd06bbcd182

File: ./contracts/interfaces/IToken.sol
SHA3: 806df0506701e04983a16f4edd6752399d1beb44a08f44e92f962fb160101ccb

File: ./contracts/interfaces/IVault.sol
SHA3: 13c1cef89dba047350e1eb0875711cbeea841e3ef91af4e7fb035c358a373966

File: ./contracts/interfaces/IVaultManagerParameters.sol
SHA3: 055b4db613c79c64b47930ab37cd4d682326c310fe45c17603b6b38995df19ff

File: ./contracts/interfaces/IVaultParameters.sol
SHA3: d5fbf3f08afd317ed8573d2392109de6345a8427e20c9564ca0df8ae9d945449

File: ./contracts/interfaces/IWETH.sol
SHA3: 3865222c4ff2eb8d605178339d19987a93cbf6abee6e830d68971fadf5531ef2

File: ./contracts/interfaces/IWrappedToUnderlyingOracle.sol
SHA3: 0df9f79c7313872a44156dc98bb30276a434285f34c29a1f53f825c26310781c

File: ./contracts/oracles/ChainlinkedOracleMainAsset.sol
SHA3: b68074cb85c192fd7aa8a2544fcd2f7b83b2a89c12c242d7b68be30386059f62

File: ./contracts/oracles/OracleRegistry.sol
SHA3: eb9d36df96f86b61b45a4ab51307a22d36c7dfcca295bb0a8cbf8e41471ff9fb

File: ./contracts/oracles/WrappedToUnderlyingOracle.sol
SHA3: 9a3494d258407efda85760d2308dec5954ff4d8f7f99d6b3ca8eb54b474d694b

File: ./contracts/test-helpers/WETH.sol
SHA3: d7ee14a4ab65418637e0d193606ee6b3035936ae9394c468c850f345089dd151

File: ./contracts/vault-managers/CDPManager01.sol
SHA3: efafa6da2164080489390dd100eef22a245abba5b728a27acd2bcff40de14a21
```

```
File: ./contracts/vault-managers/VaultManagerParameters.sol
SHA3: c645494a7adaa5d1e7328585e759bba5661cdadaf9f486e8240759383ff88efc

File: ./contracts/Vault.sol
SHA3: 0decbbf022e7776aafe5631892c403a4fe1f82a67e15c6f1b12fcb0274ad2dca

File: ./contracts/VaultParameters.sol
SHA3: d75accd4e9a4b292b9611ea7d4191a108c1ecbf27c17b5bd17553820068d61a1

File: ./contracts/helpers/IOracleRegistry.sol
SHA3: 1b265fad79101bb00d26133c7d8afc443da32b3624d3f7cdf0c78e208fba548d

File: ./contracts/helpers/IOracleUsd.sol
SHA3: 56c2b50696fca1128a2c4a5d7134d4eef0d82be665cb56f4a00d1e2b395f25d5

File: ./contracts/helpers/IVaultParameters.sol
SHA3: bb42fc6d89fd9e96c5187c275fea71bd835f47b38999aab76b82c9b7d7815a27

File: ./contracts/helpers/SafeMath.sol
SHA3: 932f79fc33490368f135c12b46329ac9a13ccd41163b9db5f1f244a66198c93e

File: ./contracts/impl/UniswapV3Oracle.sol
SHA3: 9beb6e4d633eb1e0749a20c89ae789a1439d8fcec791314d70c6615d270ad214

File: ./contracts/impl/uniswapV3Oracle/FullMath.sol
SHA3: 7857529e876071e3026c9383b78d5e2d8ce45f832b52c80ea88609457047d2d7

File: ./contracts/impl/uniswapV3Oracle/IUniswapV3Pool.sol
SHA3: f1225a22d827df6ac0de27bf8f7f30dfd32c01c22b62b110345362237c1d3e2e

File: ./contracts/impl/uniswapV3Oracle/IUniswapV3PoolDerivedState.sol
SHA3: 0bc5b04159a7fb067d8827545a982adf1a7da9b732624cfabb496615c9f993a2

File: ./contracts/impl/uniswapV3Oracle/LowGasSafeMath.sol
SHA3: 8efb50f75409d86b20ec10664f200a19149b640ae319b18970c9777fa635ed26

File: ./contracts/impl/uniswapV3Oracle/OracleLibrary.sol
SHA3: 2d1296083f65da13e2c98c1ff88b5e63f93967dc1221404736d709acc82b7aab

File: ./contracts/impl/uniswapV3Oracle/PoolAddress.sol
SHA3: efc47967301e46c120a2c4425212fa6d66d27a7e9203febf3d463fec81f39001

File: ./contracts/impl/uniswapV3Oracle/TickMath.sol
SHA3: c35e2b25d8315f9d943119b58cd5061afb54f8633cc28b46b827b81f84d2f4de

File: ./contracts/interface/IOracleRegistry.sol
SHA3: 18c6f9c801d65b9b1c72115b8d574969be12abef1677220d809120be370f73b1

File: ./contracts/interface/IOracleUsd.sol
SHA3: 3cd52c8f5fa39215243dce639a33c5d9d6448fa5081be591e29c9267453828ec

File: ./contracts/interface/IVaultParameters.sol
SHA3: 8c6e6adb471086132c1743756384f763291dfc5828ff779693881373bafbbfb0
```

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

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

Documentation quality

The total Documentation Quality score is **7** out of **10**. Functional requirements were provided as comment lines in the code and were partially missed. The Customer provided a public technical description.

Code quality

The total CodeQuality score is **2** out of **10**. Several commented code parts were found. A redundant declaration was detected. Most of the code follows the style guide. Some compilation issues were found. Deployment and basic user interactions were partly covered with tests, but some tests were not running. **Test coverage is 9.26%.**

Architecture quality

The architecture quality score is **3** out of **10**. The development environment was Hardhat with implementing tests and deployment, but instructions to run them were missing.

Security score

As a result of the audit, the code contains **16** low severity issues. The security score is **10** 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: **8.2**.

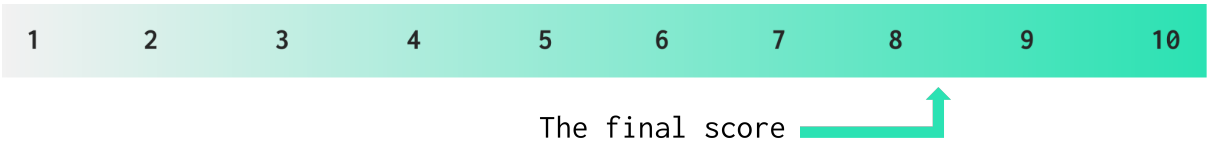

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
26 August 2022	16	1	1	0
11 October 2022	16	0	0	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.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Failed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Failed
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.	Passed
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.	Passed
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.	Passed
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

G21 Capital Marketing is a lending platform that accepts different ERC20 tokens as collateral to borrow liquidity as GCD tokens. System mints GCD tokens when a debt position is created and burns GCD tokens when a debt position is closed.

- *There are three fees for users to borrow GCD tokens:*
  - *Stability fee*
    - *It is a fee that needs to be paid when a user deposits collateral to borrow GCD tokens. The price is stable and does not fluctuate during the borrowing period once set.*
  - *Liquidation fee*
    - *It is calculated as the percentage of the loan which the borrower has to pay if liquidation is triggered. If this happens, the liquidation fee is deducted from the collateral that the user used to open the position.*
  - *Pool fee*
    - *It is a stable 3% fee that is deducted from the conversion of assets and USD values in the oracle contract (UniswapV3Oracle Line#97).*
- *CDPRegistry* – a contract that stores dynamically the data of collateral addresses and their owner addresses. It contains check functions to add or remove the owner addresses from the Collateralized Debt Position(CDP) data according to users' activeness.
- CollateralRegistry – a contract that allows adding or removing a collateral token address to the system.
- GCD- a simple ERC20 token contract. Total supply is not capped, and additional minting is allowed. Vault contract address can mint new tokens or burn users' tokens without allowance.
  It has the following attributes:
  - Name: GCD Stablecoin
  - Symbol: GCD
  - Decimals: 18
- Vault – is the core contract that stores and manages collaterals of all positions and debts. It controls the minting or burning of GCD stable coins according to the borrow or repay transactions.
- VaultParameters – an upgradable contract that allows management of GCD's system operations. It sets the fundamentals of the project that are allowed collaterals, oracle types, stability fee, liquidation fee, vault access and the foundation.
- OracleRegistry – a contract that manages the oracles by allowing setting or unsetting oracles and oracle types for assets.
- CDPManager01 – a contract that has functions to be used by the users to directly deposit collaterals, borrow GCD tokens, or repay the GCD

tokens. It is responsible for calling functions in the *Vault* contract to manage collateralized debt positions.

- LiquidationAuction02 – a contract that allows to burn GCD tokens and take calculated collateral tokens amount from liquidated positions. This buyout function can be called by anyone, but it is for the manager of the system.
- ChainlinkedOracleMainAsset – oracle contract that gets USD prices of given tokens.
- VaultManagerParameters – management contract that allows manager to set borrowing and liquidation fees.
- ReentrancyGuard – a contract module that helps prevent reentrant calls to a function.
- TransferHelper – a simple contract that contains safe transfer and safe approve functions for the given ERC20 token addresses and values.
- ForceTransferAssetStore – a contract that maps assets to the boolean status of forcing 1 token transfer.
- UniswapV3Oracle – a contract to get USD value of GCD tokens. During this conversion, the default pool fee is 3%, which is stable.
- TickMath – a math library for computing sqrt prices for ticks of size 1.0001.
- FullMath – enables division and multiplication with no loss of precision when an intermediate value is overflown.
- OracleLibrary – library contract that provides functions to integrate with V3 pool oracle.
- PoolAddress – a contract that provides functionality for deriving a pool address from the factory, tokens, and fee.
- LowGasSafeMath – library contract that provides optimized overflow and underflow safe math operations.

Privileged roles

- The manager role can:
    - add/remove an asset address to be used as collateral
    - set stability and liquidation fees, oracle type and token debt limit
    - mark asset as `shouldForceTransfer`
    - set/unset oracle addresses and their oracle types
    - set quote params in UniswapV3Oracle
    - set default TWAP period
    - set default quote asset
- The vault role can:
    - mint/burn GCD tokens
    - create new debt positions, deposit collaterals, borrow GCD tokens, withdraw collaterals, repay debt and update users' debt

- The users can:
    - deposit collaterals
    - borrow/repay GCD tokens

Findings

Critical

No critical severity issues were found.

High

1. **Data Consistency**

   After removing an oracle type, “kydonix” array is not getting updated. So, it may include oracles that do not exist anymore and may allow lending transactions with those oracles.

   Path: ./gcd-protocol-502124341/contracts/OracleRegistry.sol: setKeydonixOracleTypes()

   Recommendation: Update the list every time removing an Oracle.

   Status: Mitigated (with Customer notice)

Medium

1. **Data Consistency**

   There is no check for assets if it is registered collateral and has an oracle. In a scenario like *gcdAmount=0* and *assetAmount!=0*, users will be able to deposit any token with `join` function.

   Therefore, this may lead them to pay an unnecessary fee during depositing, and this inconsistent part may mislead the users.

   Path: ./gcd-protocol-502124341/contracts/Vault.sol: join(), depositMain()

   Recommendation: Check oracle type or use `isCollateral` function to validate asset address.

   Status: Mitigated (with Customer notice)

Low

1. **Functions that Can Be Declared External**

   “public” functions that are never called by the contract should be declared “external” to save Gas.

   Paths: ./gcd-protocol-502124341/contracts/VaultParameters.sol : initialize(),
   ./gcd-protocol-502124341/contracts/auction/LiquidationAuction02.sol : buyout(),
   ./gcd-protocol-502124341/contracts/oracles/OracleRegistry.sol: setKeydonixOracleTypes(), unsetOracle(), unsetOracleForAssets(),
   ./gcd-protocol-502124341/contracts/vault-managers/CDPManager01.sol: isLiquidatablePosition(), utilizationRatio(),
   ./gcd-protocol-502124341/contracts/vault-managers/CDPRegistry.sol: getCdpsCountForCollateral(),

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

Status: Reported

2. **Misleading Variable**

   maxOracleType variable does not take a constant value and can be changed by adding new oracle types.

   To get the oracles with getOracles() function, the function iterates over all indexes until the maxOracleType value despite some indexes being empty.

   Path: ./gcd-protocol-502124341/contracts/oracles/OracleRegistry.sol

   Recommendation: Restrict the number of oracle types and fix the unnecessary iterations.

   Status: Reported

3. **Redundant Code Block**

   On lines 112-116, declaring the *r* local variable is redundant since the function can already return the *cdps* local variable.

   Path: ./gcd-protocol-502124341/contracts/CDPRegistry.sol: getCdpsByOwner()

   Recommendation: Remove the redundant code block.

   Status: Reported

4. **State Variables' Default Visibility**

   cdpList and *cdpIndex* mapping's visibilities are not specified. Specifying state variables' visibility helps to catch incorrect assumptions about who can access the variable.

   This makes the contract`s code quality and readability higher.

   Path: ./gcd-protocol-502124341/contracts/CDPRegistry.sol

   Recommendation: Specify variables as public, internal, or private. Explicitly define visibility for all state variables.

   Status: Reported

5. **Commented Code Parts**

   In the contract, 184-185 lines are commented parts of the code.

   This reduces code quality.

   Path: ./gcd-protocol-502124341/contracts/GCD.sol

   Recommendation: Remove commented parts of the code.

   Status: Reported

6. **Commented Code Parts**

   In the contract, line 100 is a commented part of the code.

   This reduces code quality.

   Path:

   ./gcd-oracles-502124076/contracts/impl/UniswapV3Oracle.sol

   Recommendation: Remove commented parts of the code.

   Status: Reported

7. **Commented Code Parts**

   In the TickMath contract, line 26 is a commented part of the code.

   This reduces code quality.

   Path:

   ./gcd-oracles-502124076/contracts/impl/uniswapV3Oracle/TickMath.sol

   Recommendation: Remove commented parts of the code.

   Status: Reported

8. **Missing Requirement**

   In the *GCD* token contract, inside the *decreaseAllowance* function, *(currentallowance >= subtractedValue)* control is not checked. It does not fit the requirements inside the comment "`spender` must have an allowance for the caller of at least `subtractedValue`."

   Path: ./gcd-protocol-502124341/contracts/GCD.sol

   Recommendation: Add the requirement mentioned in the comments.

   Status: Reported

9. **Missing Requirement**

   In the *GCD* token contract, *_burn* function does not have a require statement to check "accountBalance >= amount" status. Therefore, it is impossible to demonstrate to the users the reason of the reverting transaction due to insufficient balance.

   Path: ./gcd-protocol-502124341/contracts/GCD.sol

   Recommendation: Add the requirement mentioned in the comments.

   Status: Reported

10. **Zero Valued Transactions**

    It is possible to deposit 0 Ether with the depositETH function. Users may send the wrong input to this function and pay a fee for nothing.

    Path: ./gcd-protocol-502124341/contracts/Vault.sol

Recommendation: Add this requirement (msg.value > 0).

Status: Reported

11. **Floating Pragma**

Some of the contracts and interfaces have different versions of Solidity. 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.

Path: all

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

Status: Reported

12. **Outdated Solidity Version**

Some contracts have outdated Solidity versions. Using an outdated compiler version can be problematic, especially if publicly disclosed bugs and issues affect the current compiler version.

Path: all

Recommendation: Use a contemporary compiler version.

Status: Reported

13. **Out-of-Gas Possibility**

Iterating over large structures (user-supplied or not), performing external calls in loops may lead to out-of-Gas exceptions. If the list of assets is too large, This can lead to out-of-Gas exceptions.

Path: ./gcd-protocol-502124341/contracts/CDPRegistry.sol: getAllCdps()

Recommendation: Implement size limitations.

Status: Reported

14. **Style Guide Violation**

The provided projects should follow the official guidelines.

Path: all

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

Status: Reported

15. **Unused Import**

Unused libraries/imports/functions/arguments should be removed from the contracts. This will help lower the Gas cost.

Paths: ./gcd-protocol-502124341/contracts/oracles/OracleRegistry.sol: VaultParameters import

./gcd-protocol-502124341/contracts/vault-managers/VaultManagerParameters.sol: VaultParameters import

./gcd-protocol-502124341/contracts/CollateralRegistry.sol: VaultParameters import

./gcd-protocol-502124341/contracts/GCD.sol: VaultParameters import

./gcd-protocol-502124341/contracts/Vault.sol: VaultParameters import

./gcd-protocol-502124341/contracts/auction/ForceTransferAssetStore.sol: VaultParameters import

./gcd-protocol-502124341/contracts/auction/LiquidationAuction02.sol: IOracleRegistry import, IWrappedToUnderlyingOracle import

Recommendation: Remove unused imports.

Status: Reported

16. **Contracts that Cannot Be Compiled**

Due to the usage of the unlocked and different Solidity versions, some contracts in *gcd-oracles* repository create compilation errors.

Path: ./gcd-oracles-502124076/contracts

Recommendation: Use the same and locked compiler version for all contracts.

Status: Reported

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.