HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: GovWorld
Date: July 21, 2023

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 GovWorld
Approved By	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU
Type	Lending Platform
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://www.govworld.io/
Changelog	22.02.2023 - Initial Review 11.04.2023 - Second Review 12.06.2023 - Third Review 21.07.2023 - Fourth Review

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

Table of contents

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

Introduction

Hacken OÜ (Consultant) was contracted by GovWorld (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/GovWorld/protocol-contracts/blob/contracts-v2
Commit	c9057e0fb5c09d596ae7ad8e86d282bff71fc83e
Whitepaper	Whitepaper
Functional Requirements	Internal functional documentation (SHA3: f9a7b72af284ced43323c10acb2c907fcb9c37fc65549d2f9c1a5f360877bf22)
Technical Requirements	Internal technical documentation (SHA3: f9a7b72af284ced43323c10acb2c907fcb9c37fc65549d2f9c1a5f360877bf22)

Contracts

File: ./contracts/facets/addressprovider/AddressProviderFacet.sol
SHA3: e163abd7a244f6f5b039aba2fd22fd9c19ddca98ef2180bd9ac008734a616b41

File: ./contracts/facets/admin/AdminRegistryFacet.sol
SHA3: 513e3cac070d5f5f5df61db9524698bd13e53b71bcbd6025c97d206396d3c748

File: ./contracts/facets/admin/LibAdmin.sol
SHA3: 080e0465a49459bf4949cfe39d7204e99ea126c6e2463a1ed4ea86a8e473c7b1

File: ./contracts/facets/admin/LibAdminStorage.sol
SHA3: 317ef785ddc0ffdc191378b25a5f271392d88935dcfdfcf674a3bf033df8758a

File: ./contracts/facets/claimtoken/ClaimTokenFacet.sol
SHA3: a34d61bcac1fadb4714edf22c89bcf78f012b1e9691b11e18681bf8d7d4b722d

File: ./contracts/facets/claimtoken/LibClaimTokenStorage.sol
SHA3: c12bb3f87b17e45e80aa642007240d230176d33910fe4a15fff5e617edb08aaf

File: ./contracts/facets/govTier/GovTierFacet.sol
SHA3: 978f04d299cc4511fc01a3b5abdca68b5d69356ad446d93ba7d4db12534c1baa

File: ./contracts/facets/govTier/LibGovTier.sol
SHA3: 599c2cff7d0c50ae9d6c5ee7db2cb9ba7d7d452a05603e80ea61d1209f6e11f1

File: ./contracts/facets/govTier/LibGovTierStorage.sol
SHA3: d8986be146db5f80efea02b592a7ca32ccf6994fe5c166b9daf12a9769a145b3

File: ./contracts/facets/gTokenFactory/GTokenFactoryFacet.sol
SHA3: 4135596fe4681b7f1c5b6ea1b1c351d473d57e37670bd41df6d949501498d7f0

File: ./contracts/facets/liquidator/LibLiquidator.sol
SHA3: c03b3da6b113e37e76ebc4f4ef0e189f5e1f4057326e32fd07ecfa05bdd6904b

File: ./contracts/facets/liquidator/LibLiquidatorStorage.sol
SHA3: 002af955e8962ca73c64366fd94ce7f80cec6edf9e4b0c82e053e703a5912a9b

File: ./contracts/facets/liquidator/LiquidatorFacet.sol
SHA3: 2fd687324b81555f9e60c6dd06c8d9bc865926e8240df5fe7dc299b295be1a23

File: ./contracts/facets/market/libraries/LibMarketProvider.sol
SHA3: 479ff406c7e08cebce68a729258af2ef5963e0e93e30638f42cfae312aafbfba

File: ./contracts/facets/market/libraries/LibMarketStorage.sol
SHA3: d9eb1dd4a8fba342aaeb000fd6edcb4f5eb7597f7159fb43b87836beb08b95e4

File: ./contracts/facets/market/libraries/LibNetworkMarket.sol
SHA3: 719d2dcecb1fb7b8af797bf8092591240eab3e1e1373f37ab7b23b35cf01cf32

File: ./contracts/facets/market/libraries/LibNFTMarket.sol
SHA3: cbf49590182552cc1138f916ba45c014175da9fbd298a241bf5b0dab5aaf6160

File: ./contracts/facets/market/libraries/LibTokenMarket.sol
SHA3: 1cf3240e67a19a03c77b1541a2ee75d78877a8d58541526d2f1bad7f3275c15a

File: ./contracts/facets/market/network/NetworkMarketFacet.sol
SHA3: 1bb3804389fcf17b8cff52956acce0bcefa1addc035bb10ff3a37f5482c2a4d3

File: ./contracts/facets/market/nft/NFTMarketFacet.sol
SHA3: d3da16e9155e2da9b96501e1b65dc4afce8292c739fd09f1b1c9405033ea4179

File: ./contracts/facets/market/token/TokenMarketFacet.sol
SHA3: b6b8ed151a8b097bbd6ea2ea152bdf684ae58c314364ed25747b604df644cb15

File: ./contracts/facets/marketRegistry/LibMarketRegistryStorage.sol
SHA3: 6557ab80b52c1d099bb1f3b7480893429a8beaaa92a68458b7699572a844e9f8

File: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol
SHA3: 4a43c7b4fe3bc43d59e81c3980a6ae4ad65126dc0fad61e5e42a7f4b02161fea

File: ./contracts/facets/nftTier/GovNFTTierFacet.sol
SHA3: ee84424d6b425f4eee6ae8a746866af08adc10c04cfee2ec92b7882aeabc8d88

File: ./contracts/facets/nftTier/LibGovNFTTier.sol
SHA3: 7fe4018764d1f650f12b5a5a7f7218993f38011592ee6d5cbb9be8b7d3408c02

File: ./contracts/facets/nftTier/LibGovNFTTierStorage.sol
SHA3: cb59981f17931bbbccf52dacdfa1c43d7c70b32d7d44337a9c804fe36e56aaac

File: ./contracts/facets/oracle/LibPriceConsumer.sol
SHA3: 6a422bcf31d3b0b3b28bd34dcd120809c48f5e96a2baf0d370372a7a59e08766

File: ./contracts/facets/oracle/LibPriceConsumerStorage.sol
SHA3: 139cbd2f5ce339bac24faecba0876fa9344908530697631793f179ba8b50688

File: ./contracts/facets/oracle/PriceConsumerFacet.sol
SHA3: 617aca5897093ab872ae4edf60fdac75a07e08b28d49fb3b8b838c439068638a

File: ./contracts/facets/pausable/PausableFacet.sol
SHA3: 761db4e871e95b1644232d8f3d91700014ca3829d62df5a856a0b86f12e79fc0

File: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol
SHA3: bc52d65507ea63abc3f942af5910b127137bd5a1110ef4cbbc17330064306468

File: ./contracts/facets/protocolRegistry/LibProtocolStorage.sol

SHA3:
11591ff612cef839b81abd77dac2032be1cb46ac0be77a54d5be25f5036a5ee3

File: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol
SHA3: add2a62761c5913dfcd5018139388f43bfac8ca1bc11ff24f3a591d57f8bf141

File: ./contracts/facets/token/GToken.sol
SHA3: c9f32cdb64011b4065a958d8c470fba84769dd2caef818b341a7e28c061114d4

File: ./contracts/facets/userTier/LibUserTier.sol
SHA3: 556783a8a9d92ba17a753af386aba4a2e098d289e6434f298c5e4bcf1b237ba3

File: ./contracts/facets/userTier/UserTierFacet.sol
SHA3: 4cae99b0a73a6cdcc0e787d1b9bc7d0cb45e39d85e2a3dbf6c25bef968b468a5

File: ./contracts/facets/vcTier/LibVCTierStorage.sol
SHA3: 93f91ff9afbac667b980b2d1193c8589b4b69966d5d089943f4e77c4399a6852

File: ./contracts/facets/vcTier/VCTierFacet.sol
SHA3: 7368f15c3f07545640b44693592201da1fb4c5c31730989ad451a3a33134950d

File: ./contracts/interfaces/IAdminRegistry.sol
SHA3: a69022317a56886e9d4913106b622e87ad00a8d6d0f56ceef897cc3343446f41

File: ./contracts/interfaces/IClaimToken.sol
SHA3: 032ea8b8455665a56287cdb8aa8681d14ef3193a256ef281f40c202a8cfdef8d

File: ./contracts/interfaces/IDexFactory.sol
SHA3: 58adf6b9b305d5baa20018b95a076a6fe37c3d562627b88c5c2fb36af6802371

File: ./contracts/interfaces/IDexPair.sol
SHA3: 4014302fef1dc17627e4e64646e624dfd4433b296e4807643280c434cdb5473b

File: ./contracts/interfaces/IDiamondCut.sol
SHA3: 4d648ccac12675848263d1f5568bf036d4ef3fe5919410ab243cbd047a1b07ff

File: ./contracts/interfaces/IDiamondLoupe.sol
SHA3: 22ffe42c8f67069b2124ef93937a8fe6faa9d912f8fc7892fc69c8f03be94664

File: ./contracts/interfaces/IERC20Extras.sol
SHA3: d9cd3ce53748b56d8e6c83d0f88908854b7fe31692e7873c435416ab0333227c

File: ./contracts/interfaces/IGovNFTTier.sol
SHA3: 430d94c3e9c892eb4474a03980b2a943ad671a4349b6e7a5d9b3b46c0c6b2811

File: ./contracts/interfaces/IGovTier.sol
SHA3: 504d97a162a5a0fd6ca74a099170b61d063beafc261519e6afa36f0791034a0f

File: ./contracts/interfaces/IGToken.sol
SHA3: b1c47075b23edc29bfd42128b245277d64f8907ffe34be4dc4b8c8e7f730257c

File: ./contracts/interfaces/IGTokenFactory.sol
SHA3: d4de0e392459ff810ab7789d217d80c1419d603ed36932cbbdfc10307e061b37

File: ./contracts/interfaces/IMarketRegistry.sol
SHA3: a9e4e2f09fcf300ef3328a56038531a73ee9da82bd955896ba32380b3f6b8a56

File: ./contracts/interfaces/IPriceConsumer.sol
SHA3: 6cccf892f4e4a827e528d6eac87917ab784f55028e2070b194994b9e2a3b8194

File: ./contracts/interfaces/IProtocolRegistry.sol
SHA3: a962296938282ba86a5d5ceb3723110240910d78168067c6586e3fa279ddd622

File: ./contracts/interfaces/IUniswapSwapInterface.sol
SHA3: c805cb12258f66ecebeb159bf412e1df042018f5b046e824af54076e80bb8aa4

File: ./contracts/interfaces/IUniswapV2Router01.sol
SHA3: e3ce9868d3833269bc63500bcb43a5ef25e4ae2fe876b2fe51fe2bed06eb9050

File: ./contracts/interfaces/IUniswapV2Router02.sol
SHA3: 62855cfb573a6af3a57d6ca56e9373514b025cc84dd1526610ab9d8fc19b37fb

File: ./contracts/interfaces/IUserTier.sol
SHA3: 2c3a76e6f0c2bce1238b9a6319486935d8323135e03aa5a72bda335576ba30b5

File: ./contracts/interfaces/IVCTier.sol
SHA3: 1389765cb2a9cdc54f28967a5103a9adc00fe517bc7fea9cd84a038dc99ad660

File: ./contracts/shared/Diamond.sol
SHA3: d7b1e88fa1a0efd68bcf507308935df709b4ae071f41aef35229e6d519ca2dee

File: ./contracts/shared/facets/DiamondCutFacet.sol
SHA3: b54f141ac489fdc7919297c9162e7dfcfe51e0eea6163096350dece5a8dc4fc1

File: ./contracts/shared/facets/DiamondLoupeFacet.sol
SHA3: 41fd84caba9eacceb1ee327ec865e9ad2c0d761dd2cc7c600d81eb090a90ffe3

File: ./contracts/shared/facets/OwnershipFacet.sol
SHA3: 093101daf7c0c3a8daf56576c79e90ee173aae7652a5cc55a5bf1e5c504feecb

File: ./contracts/shared/interfaces/IDiamondCut.sol
SHA3: 4d648ccac12675848263d1f5568bf036d4ef3fe5919410ab243cbd047a1b07ff

File: ./contracts/shared/interfaces/IDiamondLoupe.sol
SHA3: 22ffe42c8f67069b2124ef93937a8fe6faa9d912f8fc7892fc69c8f03be94664

File: ./contracts/shared/interfaces/IERC165.sol
SHA3: d2fc32ffccc9a3ce51f7a79bf14430706981ed374d2a3619fd8bb5c2c74128ba

File: ./contracts/shared/interfaces/IERC173.sol
SHA3: df587d7945371179091fc9649cd05690283cd1dcc50c445afe243fc6e63252e7

File: ./contracts/shared/libraries/LibAppStorage.sol
SHA3: de75d7e865068685d834d9278f70bf2b364dd0c45f9cdb625946301bd7f18ed3

File: ./contracts/shared/libraries/LibDiamond.sol
SHA3: 2df5cb0da4bcb01dc97a62a56504a21dd0d4e1432d33d4c0a8359887494ef6f1

File: ./contracts/shared/libraries/LibMeta.sol
SHA3: 8fb1223dd7acdab261cb57cbb9baeba499f2a26fc7d0e6aac83e9c7fc4ba6639

File: ./contracts/shared/libraries/LibPausable.sol
SHA3: c66f93afc794c0667ae72050952fdab1f64b4f5da9c28d9b24f7595c1f37c512

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/AddressStringUtil.sol
SHA3: df0fcd49fc2364a050d0ee48ce7bad77ff34718ab4d92786f7f74778a0380388

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/Babylonian.sol
SHA3: abdf57aa6c0acb54245f8322e99454a7977dbfde1af0c808e5f8818f68b304d4

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/BitMath.sol
SHA3: d77af914ac7454b5e6fe36fb9c15cb76f013d33bed66c855b7c1fcea97458f9c

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FixedPoint.sol
SHA3: b0d3da542a8dbf65f993fe968e397d457353a6124c3349369b84e7f8a640eea2

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FullMath.sol
SHA3: 297c94c68a0045d3757bc4037347ed413df9f1790e6abe2610780c52df43c9be

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/SafeERC20Namer.sol
SHA3: a4b01b142a1f18e418724e84e6bd0436a263610c99572a4ad66008bc3bdf9251

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/TransferHelper.sol
SHA3: 5e0ba812461ef72f9ca6f8033e5962cdbaaee39702e96c2fbab4bef91a1025f5

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IERC20.sol
SHA3: 24cc29ac72194902dd1dad4532ed4191bed8c5fa19624b751cd224675d4e545c

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2Callee.sol
SHA3: eedb43f6d1cd39070bc18a6217399ed38fefb3280a6acc6c4124f1e9423fc784

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2ERC20.sol
SHA3: c97b85bf5faf5024bc3dd1be8aafcd1b876692e4a81c2daeecf2b2ad3688b425

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2Factory.sol
SHA3: b0627746d906ffd79fb54e4a850fc6f7212bf99f3ce3999082ee6f5652ede02d

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2Pair.sol
SHA3: f458fcb4e4e1df7b1e646440c68b61d5de5303e10335b980d606f424e0cb7866

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/libraries/Math.sol
SHA3: 04f5d054167653e3d34cc1bcd90afd18e3408a59ef480f9a6fb2563b6f7ffbfb

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/libraries/SafeMath.sol
SHA3: 567cf3815667d04d1664dc04bc54ca6326e86845c4ff97985c79603c9aaf0fd1

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/libraries/UQ112x112.sol
SHA3: 82e8014c70a4e2f204804a9e4b31eeb065fbb96d18c9da1a419a214e2d2ad12e

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV2ERC20.sol
SHA3: 8be556fe4efc8c099e11360cc84557e3bc2b2ceb49772cba90bf89478db10fc0

```
File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV
2Factory.sol
SHA3: d3de08f1a3afd5545d8860733b9088f38cf35231d1bdf9311c4fd6c2522baef0

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV
2Pair.sol
SHA3: 56788239b960c6cd7395c97c08ea4fd16ab5c5e4aa5636fa5389d2ec2a631994

File: ./contracts/uniswap/v2-periphery-master/interfaces/IERC20.sol
SHA3: 10b180dd59c7ed1ffb033e7404be7190547e935e39632f618bc0a93f6e239c51

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Migrator.sol
SHA3: 52557b69d41955a1f7ba0a512b4fc2cb71f274c4f472948d48981edf37a7d7c7

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Router01.sol
SHA3: f48501c3bec8d7c11cb4aacf4552eb2d39eb82ac91ec12aec98294ebead8ee51

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Router02.sol
SHA3: 8d1c371a297b04b4d18c7cdd3b3be34f5aa47143f47cafba41e0bf1b16d6dd8e

File: ./contracts/uniswap/v2-periphery-master/interfaces/IWETH.sol
SHA3: f15ae6abf9e1e7dbf4d4ae696d793453d8c137ccd1681af40778b47219a2d339

File: ./contracts/uniswap/v2-periphery-master/interfaces/V1/IUniswapV1Exchange.sol
SHA3: a5b2ced2aae302f26ad0a8d28cb686e9225c4862bf09e1df3d3f869055adbfb0

File: ./contracts/uniswap/v2-periphery-master/interfaces/V1/IUniswapV1Factory.sol
SHA3: 8481bb4d99a81d27c4f3b310ed9f0d1cf1c63c376643d93a270549488c7a17fa

File: ./contracts/uniswap/v2-periphery-master/libraries/SafeMath.sol
SHA3: 7aa01d2d0c692ee14c9e011b0d5ded5f57920cafaf76bb5a766ca1fb98f8ff3d

File: ./contracts/uniswap/v2-periphery-master/libraries/UniswapV2Library.sol
SHA3: f49b157ba97bca53b585000cf513fe7198294ede2277ea02ee07cb420ed46ea0

File: ./contracts/uniswap/v2-periphery-master/libraries/UniswapV2OracleLibrary.sol
SHA3: 07a0255a6e053db1c7189d48e5cbfef51075c9c89fac53e5892f46ae59446ded

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Migrator.sol
SHA3: d972dfcbd4f9a914a6b30097aace840b0adc7942c05b861f9dc5d5f65ff2788f

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Router01.sol
SHA3: f05e30b58f26dc2bc4af01f26ee113544514c8a95452d390e76f8a369e82d2ab

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Router02.sol
SHA3: caac4e4a74c608c1b14b7a3f62f3ffdd10374c1afd8cc4454a97645d28d85301
```

Second review scope

Repository	https://github.com/GovWorld/protocol-contracts/blob/contracts-v2
Commit	ed54c7c9dab35b28c9b7fea0edfbd97fd5e7cc2d
Whitepaper	Not provided
Functional Requirements	https://govworld.gitbook.io/govworld-docs/
Technical Requirements	https://govworld.gitbook.io/govworld-docs/

Contracts

File: ./contracts/facets/addressprovider/AddressProviderFacet.sol
SHA3: c0ce6e7fdf282998156bf963cfaa22121bc219f943205a5a70402009fa3fd4cd

File: ./contracts/facets/admin/AdminRegistryFacet.sol
SHA3: e1a4a8b4465f90f495816c35f4cd42400e0d166b888c68453fcce85091923f44

File: ./contracts/facets/admin/LibAdmin.sol
SHA3: cab578818a1fc92987b6ccae14791ef9d75163139a6a44bff64fb6e78882437a

File: ./contracts/facets/admin/LibAdminStorage.sol
SHA3: 317ef785ddc0ffdc191378b25a5f271392d88935dcfdfcf674a3bf033df8758a

File: ./contracts/facets/claimtoken/ClaimTokenFacet.sol
SHA3: 1a342c2de41ffc223b3e28a24333702dba5bc52da98cea46b2556c99139890d3

File: ./contracts/facets/claimtoken/LibClaimTokenStorage.sol
SHA3: ceed8eb984218c64766f04631261de063f2f038f215b0f1d2a8e469ab0be3680

File: ./contracts/facets/govTier/GovTierFacet.sol
SHA3: 43ca2936b6c900fe8db0adcdaa67ad3af93f4abfdc43f613dc91bccd39fe2504

File: ./contracts/facets/govTier/LibGovTier.sol
SHA3: afa1beb1a383090fb7d3cf29256efe5f892b4912ea6d010fb47cbef6a77dc892

File: ./contracts/facets/govTier/LibGovTierStorage.sol
SHA3: 7761211e336ff5635cb70a65298ee3b0639f55057ab5435c5d607753d5b2a025

File: ./contracts/facets/gTokenFactory/GTokenFactoryFacet.sol
SHA3: 5e5679c351419ab8b57604c6458a5409729afdd266e881d7be25c06b852a5d5d

File: ./contracts/facets/liquidator/LibLiquidator.sol
SHA3: 80f57e1b348602d81830ebba62cec4aee644447e3bce3440f16901da6e43c4d1

File: ./contracts/facets/liquidator/LibLiquidatorStorage.sol
SHA3: b63e0d85a14e011f23b464d1fa61d0da19f2c732fc1e72faa97e3f5f12e53ed3

File: ./contracts/facets/liquidator/LiquidatorFacet.sol
SHA3: f693ad468a156538afab45cd72670c9823261c46066dede3ee21e0b7e0e00440

File: ./contracts/facets/market/libraries/LibMarketProvider.sol
SHA3: d859447f2c7b270c88d8d4d209f830add77fada948ad66dd82399f7ed989cd45

File: ./contracts/facets/market/libraries/LibMarketStorage.sol
SHA3: 35e7e41ca6a894b73a6f70bbe0f332ce94b26cf113b17446a2bf85364a1678b8

File: ./contracts/facets/market/libraries/LibNetworkMarket.sol
SHA3: f7abc1202367d093fe17b3c7061af69795c4542aa05cf9299771d8582324580b

File: ./contracts/facets/market/libraries/LibNFTMarket.sol
SHA3: c8f3a22701130eed33533b8e06ff5fbe709819ef19869d1c9272a720b5d46665

File: ./contracts/facets/market/libraries/LibTokenMarket.sol
SHA3: 906baddf52f1fd08f07c7a601f00022d4d42b574e897d82f250c275f9e2cbde5

File: ./contracts/facets/market/network/NetworkMarketFacet.sol
SHA3: dc738dd01cfd2deb73bf925212c3479d87af232d81ac3ff1e55a07fff0933e3e

File: ./contracts/facets/market/nft/NFTMarketFacet.sol
SHA3: aeafb8a3f3d5a110d45f32df693ef803c6fb7b797f9ff32c7d7f9e2d53e4ee21

File: ./contracts/facets/market/token/TokenMarketFacet.sol
SHA3: ab473d9c18cd7d4d2dac0eec21a16b2342868bf4cf5ba293ee813ea8cabe3c5a

```
File:
./contracts/facets/marketRegistry/LibMarketRegistryStorage.sol
SHA3: 069461d88d71482d6b39a4758e96264d5e45bbb5550b3d0be037eac2a5362b09

File: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol
SHA3: 0b05af9d441660167504800ef368c3bda5856718e0680f7fff4bc39585a7ceeb

File: ./contracts/facets/nftTier/GovNFTTierFacet.sol
SHA3: 111991433e4c8912388ef3ce2e22a1e92189af0c16ac57019a57f1b9dc242409

File: ./contracts/facets/nftTier/LibGovNFTTier.sol
SHA3: cb1c9e4362844972ce8221ddc30e59c425ca0589b38e080b873e154f92e53a4a

File: ./contracts/facets/nftTier/LibGovNFTTierStorage.sol
SHA3: 024dad4afb9eed2eecaa8360835576d50ee3811cad309be7c210c56288a27e66

File: ./contracts/facets/oracle/LibPriceConsumer.sol
SHA3: 3a96a047daafa1f51518128b68670b2225f75502eacbbe8fb6460ee652f0929f

File: ./contracts/facets/oracle/LibPriceConsumerStorage.sol
SHA3: f0e0650d3bfa9f8ed62963fe3f2d7888fd639260b89614368b9f286665a2ac11

File: ./contracts/facets/oracle/PriceConsumerFacet.sol
SHA3: c53f90282f53cd5e25ef6bd0300ee012fc2051d24b101e9a48d65c1f482ca6c6

File: ./contracts/facets/pausable/PausableFacet.sol
SHA3: 54b9be28c9db0406ffe10080de00d3f5a6fbfba6b1921b9ab702bd8f0200f1d5

File: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol
SHA3: 5cc3933b9bfb05044420698e6c7461c40c9b225f9fbe3ecd45d31b3887c04ba8

File: ./contracts/facets/protocolRegistry/LibProtocolStorage.sol
SHA3: 9751fb089963239415092222d4afb1a789d74b7476e6367e8c9202459c82bf698

File: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol
SHA3: b998ad97000775de671579ec705e7fd0db54ae3ed19a6d4aa416eb1b17f2cae6

File: ./contracts/facets/token/GToken.sol
SHA3: 59ec26bdf0e1a416df090b8c558b1901a899fbcd057a6b5376dae19427afcb91

File: ./contracts/facets/userTier/LibUserTier.sol
SHA3: 73619ec8f32ccf39e45420d97d4976ba56bbbae729fdd6320a39bff27056350d

File: ./contracts/facets/userTier/UserTierFacet.sol
SHA3: 5fae272cc8691f357c66abd9c5844a950364d3d4de4ad2a8f803c011a391841d

File: ./contracts/facets/vcTier/LibVCTierStorage.sol
SHA3: 4451b8e8e1ad0c8d8bc5c3caead6f5eb1543866c098d56b6d4289bbbb3aecd22

File: ./contracts/facets/vcTier/VCTierFacet.sol
SHA3: 537a2c67876cc5cc662dee71b1e0ffaa99f9194a8fb1fe0f65087b1c863bd201

File: ./contracts/interfaces/IAdminRegistry.sol
SHA3: a69022317a56886e9d4913106b622e87ad00a8d6d0f56ceef897cc3343446f41

File: ./contracts/interfaces/IClaimToken.sol
SHA3: 56d1ac9c37253f6d03cf9c06148aaad1be06e8931cce63028e2ae821af579486

File: ./contracts/interfaces/IDexFactory.sol
SHA3: 6bc45374c37092ca74cf6e88fdc3f1e5bc5f22d64eb25405a11e3c1567ccbfa3

File: ./contracts/interfaces/IDexPair.sol
SHA3: 6c06d64a4e84cecf4ed9e1aee1ab825f2e86e3391e0615db25deccffd5730b48

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

SHA3:
4d648ccac12675848263d1f5568bf036d4ef3fe5919410ab243cbd047a1b07ff

File: ./contracts/interfaces/IDiamondLoupe.sol
SHA3: 22ffe42c8f67069b2124ef93937a8fe6faa9d912f8fc7892fc69c8f03be94664

File: ./contracts/interfaces/IERC20Extras.sol
SHA3: e3b2fb98c8ba819670cf6f3eb5ae52630785f95876de1315971be7c240ba8be2

File: ./contracts/interfaces/IERC721Extras.sol
SHA3: 26e11317cccc3b6bddbded144be75300c50a2a2e54d28812f41f58b05ce43034

File: ./contracts/interfaces/IGovNFTTier.sol
SHA3: 430d94c3e9c892eb4474a03980b2a943ad671a4349b6e7a5d9b3b46c0c6b2811

File: ./contracts/interfaces/IGovTier.sol
SHA3: 504d97a162a5a0fd6ca74a099170b61d063beafc261519e6afa36f0791034a0f

File: ./contracts/interfaces/IGToken.sol
SHA3: cbaa60ed2fa70d80b0bdb6ea3a56083e53c12356f3ea060a2bf52784e5b76809

File: ./contracts/interfaces/IGTokenFactory.sol
SHA3: 8f62ef0d6be2e681e1d4c713a53012ec49ec17ccfafd8b9e93bdf5dad607e679

File: ./contracts/interfaces/IMarketRegistry.sol
SHA3: a9e4e2f09fcf300ef3328a56038531a73ee9da82bd955896ba32380b3f6b8a56

File: ./contracts/interfaces/IPriceConsumer.sol
SHA3: 55dfec013e7b16bb6949b3dcbd19943a74aa610d81007d31db84d905ff6513bf

File: ./contracts/interfaces/IProtocolRegistry.sol
SHA3: 6c9107caa5b17fa9bfb3be15495da271a91d224f1fc40f04706240681c6c3bc8

File: ./contracts/interfaces/IUniswapSwapInterface.sol
SHA3: c805cb12258f66ecebeb159bf412e1df042018f5b046e824af54076e80bb8aa4

File: ./contracts/interfaces/IUniswapV2Router01.sol
SHA3: e3ce9868d3833269bc63500bcb43a5ef25e4ae2fe876b2fe51fe2bed06eb9050

File: ./contracts/interfaces/IUniswapV2Router02.sol
SHA3: 62855cfb573a6af3a57d6ca56e9373514b025cc84dd1526610ab9d8fc19b37fb

File: ./contracts/interfaces/IUserTier.sol
SHA3: 2c3a76e6f0c2bce1238b9a6319486935d8323135e03aa5a72bda335576ba30b5

File: ./contracts/interfaces/IVCTier.sol
SHA3: 1389765cb2a9cdc54f28967a5103a9adc00fe517bc7fea9cd84a038dc99ad660

File: ./contracts/shared/Diamond.sol
SHA3: d7b1e88fa1a0efd68bcf507308935df709b4ae071f41aef35229e6d519ca2dee

File: ./contracts/shared/facets/DiamondCutFacet.sol
SHA3: b54f141ac489fdc7919297c9162e7dfcfe51e0eea6163096350dece5a8dc4fc1

File: ./contracts/shared/facets/DiamondLoupeFacet.sol
SHA3: 41fd84caba9eacceb1ee327ec865e9ad2c0d761dd2cc7c600d81eb090a90ffe3

File: ./contracts/shared/facets/OwnershipFacet.sol
SHA3: 1a248f734988437e53dfd663cfbcd34abf67c72d03a38b79a84f2f5cb254e3aa

File: ./contracts/shared/interfaces/IDiamondCut.sol
SHA3: 4d648ccac12675848263d1f5568bf036d4ef3fe5919410ab243cbd047a1b07ff

File: ./contracts/shared/interfaces/IDiamondLoupe.sol
SHA3: 22ffe42c8f67069b2124ef93937a8fe6faa9d912f8fc7892fc69c8f03be94664

File: ./contracts/shared/interfaces/IERC165.sol
SHA3: d2fc32ffccc9a3ce51f7a79bf14430706981ed374d2a3619fd8bb5c2c74128ba

File: ./contracts/shared/interfaces/IERC173.sol
SHA3: df587d7945371179091fc9649cd05690283cd1dcc50c445afe243fc6e63252e7

File: ./contracts/shared/libraries/LibAppStorage.sol
SHA3: 4ada2aa15a1f1fc36d05bcef0349e1da5615e3706a769675f87cb605be0971a3

File: ./contracts/shared/libraries/LibDiamond.sol
SHA3: 1bbb11f46e30fb7b0a931ef348aeb460ba49acc762542c93f1e9ecd931aedee5

File: ./contracts/shared/libraries/LibMeta.sol
SHA3: 843971b0ad9e1a9ef1784a6b72507eae7fac1fd8eeeacd3b114c89eda665135a

File: ./contracts/shared/libraries/LibPausable.sol
SHA3: df534bfd71db25ead4893d7751422cdad9cbbd34baaea35964774e61ca32a7e4

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/AddressStringUtil.sol
SHA3: df0fcd49fc2364a050d0ee48ce7bad77ff34718ab4d92786f7f74778a0380388

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/Babylonian.sol
SHA3: abdf57aa6c0acb54245f8322e99454a7977dbfde1af0c808e5f8818f68b304d4

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/BitMath.sol
SHA3: d77af914ac7454b5e6fe36fb9c15cb76f013d33bed66c855b7c1fcea97458f9c

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FixedPoint.sol
SHA3: b0d3da542a8dbf65f993fe968e397d457353a6124c3349369b84e7f8a640eea2

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FullMath.sol
SHA3: 297c94c68a0045d3757bc4037347ed413df9f1790e6abe2610780c52df43c9be

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/SafeERC20Namer.sol
SHA3: a4b01b142a1f18e418724e84e6bd0436a263610c99572a4ad66008bc3bdf9251

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/TransferHelper.sol
SHA3: 5e0ba812461ef72f9ca6f8033e5962cdbaaee39702e96c2fbab4bef91a1025f5

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IERC20.sol
SHA3: 24cc29ac72194902dd1dad4532ed4191bed8c5fa19624b751cd224675d4e545c

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2Callee.sol
SHA3: eedb43f6d1cd39070bc18a6217399ed38fefb3280a6acc6c4124f1e9423fc784

```
File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfac
es/IUniswapV2ERC20.sol
SHA3: c97b85bf5faf5024bc3dd1be8aafcd1b876692e4a81c2daeecf2b2ad3688b425

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfac
es/IUniswapV2Factory.sol
SHA3: b0627746d906ffd79fb54e4a850fc6f7212bf99f3ce3999082ee6f5652ede02d

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfac
es/IUniswapV2Pair.sol
SHA3: f458fcb4e4e1df7b1e646440c68b61d5de5303e10335b980d606f424e0cb7866

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/librarie
s/Math.sol
SHA3: 04f5d054167653e3d34cc1bcd90afd18e3408a59ef480f9a6fb2563b6f7ffbfb

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/librarie
s/SafeMath.sol
SHA3: 567cf3815667d04d1664dc04bc54ca6326e86845c4ff97985c79603c9aaf0fd1

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/librarie
s/UQ112x112.sol
SHA3: 82e8014c70a4e2f204804a9e4b31eeb065fbb96d18c9da1a419a214e2d2ad12e

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV
2ERC20.sol
SHA3: 8be556fe4efc8c099e11360cc84557e3bc2b2ceb49772cba90bf89478db10fc0

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV
2Factory.sol
SHA3: b34bc571fc7e5fe11baa81acf3592d549ff4dfaa54245844eaf40e754be21df4

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV
2Pair.sol
SHA3: 56788239b960c6cd7395c97c08ea4fd16ab5c5e4aa5636fa5389d2ec2a631994

File: ./contracts/uniswap/v2-periphery-master/interfaces/IERC20.sol
SHA3: 10b180dd59c7ed1ffb033e7404be7190547e935e39632f618bc0a93f6e239c51

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Migrator.sol
SHA3: 52557b69d41955a1f7ba0a512b4fc2cb71f274c4f472948d48981edf37a7d7c7

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Router01.sol
SHA3: f48501c3bec8d7c11cb4aacf4552eb2d39eb82ac91ec12aec98294ebead8ee51

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Router02.sol
SHA3: 8d1c371a297b04b4d18c7cdd3b3be34f5aa47143f47cafba41e0bf1b16d6dd8e

File: ./contracts/uniswap/v2-periphery-master/interfaces/IWETH.sol
SHA3: f15ae6abf9e1e7dbf4d4ae696d793453d8c137ccd1681af40778b47219a2d339

File: ./contracts/uniswap/v2-periphery-master/interfaces/V1/IUniswapV1Exchange.sol
SHA3: a5b2ced2aae302f26ad0a8d28cb686e9225c4862bf09e1df3d3f869055adbfb0

File: ./contracts/uniswap/v2-periphery-master/interfaces/V1/IUniswapV1Factory.sol
SHA3: 8481bb4d99a81d27c4f3b310ed9f0d1cf1c63c376643d93a270549488c7a17fa
```

File: ./contracts/uniswap/v2-periphery-master/libraries/SafeMath.sol
SHA3: 7aa01d2d0c692ee14c9e011b0d5ded5f57920cafaf76bb5a766ca1fb98f8ff3d

File: ./contracts/uniswap/v2-periphery-master/libraries/UniswapV2Library.sol
SHA3: f49b157ba97bca53b585000cf513fe7198294ede2277ea02ee07cb420ed46ea0

File: ./contracts/uniswap/v2-periphery-master/libraries/UniswapV2OracleLibrary.sol
SHA3: 07a0255a6e053db1c7189d48e5cbfef51075c9c89fac53e5892f46ae59446ded

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Migrator.sol
SHA3: d972dfcbd4f9a914a6b30097aace840b0adc7942c05b861f9dc5d5f65ff2788f

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Router01.sol
SHA3: f05e30b58f26dc2bc4af01f26ee113544514c8a95452d390e76f8a369e82d2ab

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Router02.sol
SHA3: 7da7ca7f6459ca9b448f90b1e7f3ec3fb9fed4bb17208e6c26b7eb4e60d7da59

Third review scope

Repository	https://github.com/GovWorld/protocol-contracts/blob/contracts-v2
Commit	3c0d52c047425a95075ecda2458c60612acf7c20
Whitepaper	Not provided
Functional Requirements	https://govworld.gitbook.io/govworld-docs/
Technical Requirements	https://govworld.gitbook.io/govworld-docs/

Contracts

File: ./contracts/facets/addressprovider/AddressProviderFacet.sol
SHA3: c0ce6e7fdf282998156bf963cfaa22121bc219f943205a5a70402009fa3fd4cd

File: ./contracts/facets/admin/AdminRegistryFacet.sol
SHA3: 34886b9aacb3222e489cf1deb8524b89afd9acd2c42075034c71aae49f69ea13

File: ./contracts/facets/admin/LibAdmin.sol
SHA3: 8bf4e9794a1c82c44df61e245c14f80b5133a1a84d5b30839583da128f83e06a

File: ./contracts/facets/admin/LibAdminStorage.sol
SHA3: 863565c73136041ef502f549a0bbb722216a8bffd28e38bb25c01b8d470f8d60

File: ./contracts/facets/claimtoken/ClaimTokenFacet.sol
SHA3: c04cd193ee17f3ce6932c0af2f280998e3acf4be133c74cbd08fddd1c771bf1a

File: ./contracts/facets/claimtoken/LibClaimTokenStorage.sol
SHA3: ceed8eb984218c64766f04631261de063f2f038f215b0f1d2a8e469ab0be3680

File: ./contracts/facets/govTier/GovTierFacet.sol
SHA3: abe22669b8ae347eed48d5dc0cbad9d5f41e8670e9f3cad14eed62c89850a6bb

File: ./contracts/facets/govTier/LibGovTier.sol
SHA3: e485004776ad47e3a63b716bf30cb9524fc16037ec740019c54e3d71cc4a6620

File: ./contracts/facets/govTier/LibGovTierStorage.sol
SHA3: 42c149dcd3275c313f51516c1be0d4e2b17e55e3cfe71541d4afdf17e3abbfa4

File: ./contracts/facets/gTokenFactory/GTokenFactoryFacet.sol
SHA3: d6be7e8665a23099cf52ac11660e8ca6b22fa4614de4231029df08a4ff6b2766

File: ./contracts/facets/liquidator/LibLiquidator.sol
SHA3: b9246e4691a092bc832a64486fb037aee7534a0a76fb87ad5f0aae5b12828bf9

File: ./contracts/facets/liquidator/LibLiquidatorStorage.sol
SHA3: b63e0d85a14e011f23b464d1fa61d0da19f2c732fc1e72faa97e3f5f12e53ed3

File: ./contracts/facets/liquidator/LiquidatorFacet.sol
SHA3: aca4a4ce89e30e37ffa4bdd47ad9f42b2518b45e96e489733ad495dd5afdb285

File: ./contracts/facets/market/libraries/LibMarketProvider.sol
SHA3: a4594a1392861d877a703b316628575e3326860ba533033e192f364338187ae4

File: ./contracts/facets/market/libraries/LibMarketStorage.sol
SHA3: 82d481b39b7e37de30c7bbbde80134acd5261081cb7e4f5d93b62b3c87e0516a

File: ./contracts/facets/market/libraries/LibNetworkMarket.sol
SHA3: 5f146de640f66eb44e690acaca0c77d4594320ca06ea6fc9a0244900add97979

File: ./contracts/facets/market/libraries/LibNFTMarket.sol
SHA3: 540ad0fb2b0fd049c2400476bf08518cce8c4c91523e2e9d69bc523d52bedeed

File: ./contracts/facets/market/libraries/LibTokenMarket.sol
SHA3: f54c432aa24f9e12177e29c41789c5315c2230a8d48e8da87d48fc6c1aed1982

File: ./contracts/facets/market/network/NetworkMarketFacet.sol
SHA3: 2c9951a4ebc3222bf0e450251c5bed0d662f63df470004bddf0560befdde1a73

File: ./contracts/facets/market/nft/NFTMarketFacet.sol
SHA3: bfdb1727cb892dbd36ab79f3e7c546bdcc3ff26a7a220cdd1056ef8f1cd1d67a

File: ./contracts/facets/market/token/TokenMarketFacet.sol
SHA3: a566c077e1435f1597d216486daab43e2e845e09807c73a22ca6fa11eb1474fc

File: ./contracts/facets/marketRegistry/LibMarketRegistryStorage.sol
SHA3: fc6d019928fd5e403f6a03932c9ce9862ba2bbd126c35af714ffbb53f7cf389b

File: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol
SHA3: 4f15cf7f8bd7d817ba5a48900510330ed986e1c88777975e2426d5d748626b8b

File: ./contracts/facets/nftTier/GovNFTTierFacet.sol
SHA3: 765b13afa38eccd224af4f197f6784ef138ede4ff7c03a173ca792c8596df01a

File: ./contracts/facets/nftTier/LibGovNFTTier.sol
SHA3: 0107c83b90fbe505920e9cfb5d01ea5c0b2c7d7950a467d689c66fb04f2f3cec

File: ./contracts/facets/nftTier/LibGovNFTTierStorage.sol
SHA3: 57fbc5448938b2c9d9bd92c19ba6d0f90dd3c14e0b2ffe7432d475cad36385c9

File: ./contracts/facets/oracle/LibPriceConsumer.sol
SHA3: 977e7259335162389611294ffebc3439c04ca8a43a1a9344713ef84f028109cf

File: ./contracts/facets/oracle/LibPriceConsumerStorage.sol
SHA3: 59053a1027b4cf2136d0ae58f18e536680fd3f83710b07d462b5eb057e29d9c3

File: ./contracts/facets/oracle/PriceConsumerFacet.sol
SHA3: c2817af9a0ce0dfa48455c222925099422c47a309ad697d847e958b91a95bc49

File: ./contracts/facets/oracle/UniswapOracleV3.sol
SHA3: 2e3a3f7dd07248055438723bfe550cc42f7d70fcd12e67f0399c5989df4c68d4

File: ./contracts/facets/pausable/PausableFacet.sol

SHA3:
54b9be28c9db0406ffe10080de00d3f5a6fbfba6b1921b9ab702bd8f0200f1d5

File: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol
SHA3: a9c5d199c78c8bb43a41ac83789f4ad0de8a5f7148241385a47721c8ea1f4869

File: ./contracts/facets/protocolRegistry/LibProtocolStorage.sol
SHA3: 88f6ab9e001c48d6b158fe8d67d9380ef274bad70c5e12f13d1c6d19ff1ef8c8

File: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol
SHA3: 30c07c93240071dc62f025493b1fe925bbb0b2eb662a8ccfdf7404fe41b447e3

File: ./contracts/facets/token/GToken.sol
SHA3: 6af29e423b9d873269d23248cf539a8cedddd703c96b2e84b7a9086a2d947774

File: ./contracts/facets/userTier/LibUserTier.sol
SHA3: 73619ec8f32ccf39e45420d97d4976ba56bbbae729fdd6320a39bff27056350d

File: ./contracts/facets/userTier/UserTierFacet.sol
SHA3: fd76f5c470f3c490927226a921aeb2019e270f203bfcc2a45c2e241fc3f8adb4

File: ./contracts/facets/vcTier/LibVCTierStorage.sol
SHA3: e408b029d3e48b48c131a2a077c68674842efc65c6f81f82ce1e7d918d09e17d

File: ./contracts/facets/vcTier/VCTierFacet.sol
SHA3: 6a5e70d13031eb037c0f5be8ae117ee4e00f48c2c40ae2e64e489970fb5d7eab

File: ./contracts/interfaces/IAdminRegistry.sol
SHA3: 4c210c0f2e36ac9ff059849f397dcb8dfd391b08355e96b67cc33ddb0c8dea30

File: ./contracts/interfaces/IClaimToken.sol
SHA3: 570e891ed2481ba65ca3f863d8fbc45debcb2834025c3946cf4d7e661b08dc94

File: ./contracts/interfaces/IDexFactory.sol
SHA3: 870fa283df6e1303c710433dd72026138178de78d585b716cdba06e602f25ed6

File: ./contracts/interfaces/IDexPair.sol
SHA3: 9cb7d3bfc87617d59f5013950261d3444dff6f5b1cafa3372ee1f4ac75edeaac

File: ./contracts/interfaces/IDiamondCut.sol
SHA3: 097ab56d21ca87e1260ed732d35b1928724f207337eaca41d89141bfb8349b99

File: ./contracts/interfaces/IDiamondLoupe.sol
SHA3: 90cdc2f3d119001a023cf3d1d600761551625f42d582ffad112b33fae29a161e

File: ./contracts/interfaces/IERC20Extras.sol
SHA3: e3b2fb98c8ba819670cf6f3eb5ae52630785f95876de1315971be7c240ba8be2

File: ./contracts/interfaces/IERC721Extras.sol
SHA3: 26e11317cccc3b6bddbded144be75300c50a2a2e54d28812f41f58b05ce43034

File: ./contracts/interfaces/IGovNFTTier.sol
SHA3: 67994d64fe466ea7041dfa014126242716e4cefa68cce69fd490a8d678f2a732

File: ./contracts/interfaces/IGovTier.sol
SHA3: 08ff9c36d7b95804ac9ded8a4daa49a74775c3d4a4b84961dab4748aa2bb2b91

File: ./contracts/interfaces/IGToken.sol
SHA3: cbaa60ed2fa70d80b0bdb6ea3a56083e53c12356f3ea060a2bf52784e5b76809

File: ./contracts/interfaces/IGTokenFactory.sol
SHA3: dd9eb048be64d11c424681672c0919fba35f4a81dd52d2724e75d9a5b87b5485

File: ./contracts/interfaces/IMarketRegistry.sol
SHA3: dffb84364fa3877f60f62bc1d41e89f6b18c2e998d9c4036540e85db84ae6aac

File: ./contracts/interfaces/IPriceConsumer.sol
SHA3: 3a783d69c184c4d2fb660d88c7d9087b91e569007c6f762f78194c43ec852623

File: ./contracts/interfaces/IProtocolRegistry.sol
SHA3: cfadc6d6399e1fe3acf759bb08cc3608457784882af5654b50b2e915e7e760d5

File: ./contracts/interfaces/IUniswapOracleV3.sol
SHA3: ca5f4019f65135e3fafca21710dce9708e25ff6198499cc72ad3778d2cb6398d

File: ./contracts/interfaces/IUniswapSwapInterface.sol
SHA3: efd32d7d7dbb5dfb0c9ba6074a0367782af1ef9b2d5bc9c331c616cc43941a08

File: ./contracts/interfaces/IUniswapV2Router01.sol
SHA3: efc965b76796230d3891d4b1aad253d2f8505c2f93ceac848b80842a9e74c518

File: ./contracts/interfaces/IUniswapV2Router02.sol
SHA3: 62855cfb573a6af3a57d6ca56e9373514b025cc84dd1526610ab9d8fc19b37fb

File: ./contracts/interfaces/IUniswapV3Factory.sol
SHA3: 94e9658d22a17443e9239084c69b5564b05ea8b5035c6b53496701d234ae08d5

File: ./contracts/interfaces/IUniswapV3Router.sol
SHA3: ee81e5d470eaf442e4246dad1d675abd8321cff23dcea9c50478fa593ce5801c

File: ./contracts/interfaces/IUserTier.sol
SHA3: 4df71e8f1a1ee5d63bc1cd5760be4cdecfb9911a61801380fcd2b6df94436339

File: ./contracts/interfaces/IVCTier.sol
SHA3: 98f60723ac0c5b923c6e5a551823de956e3604fd3166cec3d181cefb15390eb6

File: ./contracts/shared/Diamond.sol
SHA3: 87992ecfcf9b825aa00c46e89652a3362c0a6b16ea271db81a0f0a55c64f468b

File: ./contracts/shared/facets/DiamondCutFacet.sol
SHA3: 05a8390cefbff7131d5b91798ab96f4641cecbf48905cf33cbea02b13245bd82

File: ./contracts/shared/facets/DiamondLoupeFacet.sol
SHA3: 8ad8c8c614cd363c1ab45b98878988eb65df965bc75d10c5d0b80460df43a469

File: ./contracts/shared/facets/OwnershipFacet.sol
SHA3: b248daedc25610b9353257c90ca8b51eb6802e2ae943ddbf8410c897e6f7d80e

File: ./contracts/shared/interfaces/IDiamondCut.sol
SHA3: 097ab56d21ca87e1260ed732d35b1928724f207337eaca41d89141bfb8349b99

File: ./contracts/shared/interfaces/IDiamondLoupe.sol
SHA3: 90cdc2f3d119001a023cf3d1d600761551625f42d582ffad112b33fae29a161e

File: ./contracts/shared/interfaces/IERC165.sol
SHA3: 2db7e91cf306335bccf3e9d84ece8a71d376f930daf61122b2571bd324d66a58

File: ./contracts/shared/interfaces/IERC173.sol
SHA3: 4f32c30a4b8c3894cd9276e0d9436910caa4b9ea09ca0ef292b77108dc55cf39

File: ./contracts/shared/libraries/LibAppStorage.sol
SHA3: 4ada2aa15a1f1fc36d05bcef0349e1da5615e3706a769675f87cb605be0971a3

File: ./contracts/shared/libraries/LibDiamond.sol
SHA3: e6e0a5cde2fabf2d742837a415bfc68730ba4edc102c0c10252a56efde6fc168

File: ./contracts/shared/libraries/LibMeta.sol
SHA3: 0a312736a8f5e05d2550bf689d4b1dc4d51b2ea70fe66afee2e708f8334140e7

File: ./contracts/shared/libraries/LibPausable.sol

SHA3:
df534bfd71db25ead4893d7751422cdad9cbbd34baaea35964774e61ca32a7e4

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Migrator.sol
SHA3: 4a626e4ec0864c6eb6475dc93a0d18045f4883d028006ad90019dce0cdb374eb

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Router01.sol
SHA3: 0e8867b9a87d9df07eae068f2c342f090c77b4b4fed5e552f79022c8eae2e2a8

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Router02.sol
SHA3: 02ef0ec1bf1d2057e66a4f9d56a62fc0b48f0d269229792d72321fead3cca9dd

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/AddressStringUtil.sol
SHA3: b9c4e63fcbf4a879c477d8c4fd347837d2668a4a2a6a1005d17ca9d1cd96a861

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/Babylonian.sol
SHA3: 6ea2bfd8b989ec7d616d22b725d5ff874f29be2a0533427d011725a48bf52e6b

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/BitMath.sol
SHA3: 34b174459a8734ddc07284797ba967ec5e91f6776e8db225943d0c395c79266a

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FixedPoint.sol
SHA3: 66966ce56c3063843982a3cde51bd0761afd1aa9a8e23dd98103e84afba729e6

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FullMath.sol
SHA3: 78a3e49791c5c474d49e88a681c1b9be90471c9f0218802f39b8701bdeff312c

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/SafeERC20Namer.sol
SHA3: 38b9bbadb9fce6ca34ccc1fe4e1190724b5f5c2a7d4b077c1ae2ce75985754ad

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/TransferHelper.sol
SHA3: c8b7cc0e76242f60dff0e224b7b27c4d8fae5477661789104d47cf59885f910e

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV2ERC20.sol
SHA3: 4ddb0763c4561a9ff3a52a7837aeb7dc88cbe79bcbe1d54659b97da8962959fa

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV2Factory.sol
SHA3: 0e2db8efa398c32bec215d16f11014140fe38e34e5577e2527cc2d7d68b06734

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV2Pair.sol
SHA3: b669b00dba5bdbfa97da9e5b6607f0ee2c31ad3642dedff549c88404715e8300

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IERC20.sol

SHA3:
42e4b863a781d4cea0b2cb626d5bb58684d397a544fbf160dffc3a0148afc1d2

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2Callee.sol
SHA3: 56e42f76e0b6298b3b6d440efc13f2a670c2c85e013511277b74465f69a79f18

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2ERC20.sol
SHA3: 04d4cc12d43a113c7b1e59fb15f18ae1c277563f79fa3d135ddf5f85659dc8b9

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2Factory.sol
SHA3: 1029024b2479a10eaf1a7c45507b63f1cc1212e5b1674892120f0410641271f3

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2Pair.sol
SHA3: 1a67b3a6d7369f6936b5976379edd61d476e4295abca0c15405de1276a1e531b

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/libraries/Math.sol
SHA3: 931c3cb4dee32a740a8c2ac7f845de5ac733703652cfcf6a2d43d1faeb25e72b

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/libraries/SafeMath.sol
SHA3: 5a2ee0cd9bde270a65a7bbf231cab9bb4dc04f8f4d6faa4984f9951ee10d2200

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/libraries/UQ112x112.sol
SHA3: d2eccc238c08eb60a9068c02a5c01bca3566b878324684d93bb3570f02c67512

File: ./contracts/uniswap/v2-periphery-master/interfaces/IERC20.sol
SHA3: dcb2e043bde8a32f25fa7faa805041130e67d31bc8fad2ab5e568f4363290a5c

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Migrator.sol
SHA3: 715808ea0f1dde62f29190cf8a4b26c52c35fc21ba95799d8daf66634404f944

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Router01.sol
SHA3: ede34403bc4c23c55dbf1dd7bebdbec1a839d8983b8803d4c91a410e015d280f

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Router02.sol
SHA3: 521ea3060bf020b52e6af0bbe77d50dd8f4faea788b1d4d15414d5ba48b3285a

File: ./contracts/uniswap/v2-periphery-master/interfaces/IWETH.sol
SHA3: 633ed9ff88e7d7d06b9341a98aa08717d936279af05aa35666f7d19b04b4d0a7

File: ./contracts/uniswap/v2-periphery-master/interfaces/V1/IUniswapV1Exchange.sol
SHA3: c81cfb4a356f2d0298a9bde96219d8b58c98b10161e60eb0003f0a6d62eda9bb

File: ./contracts/uniswap/v2-periphery-master/interfaces/V1/IUniswapV1Factory.sol
SHA3: b5e8b8c643e7ed5858e12fdf40c7be71da0325e12a44ecc29fdd889ab3aa368a

File: ./contracts/uniswap/v2-periphery-master/libraries/SafeMath.sol
SHA3: 8303fdecb7e19e705dffcd1716f127f8f970ad7b1046f049f36db66fa63fff15

File: ./contracts/uniswap/v2-periphery-master/libraries/UniswapV2Library.sol
SHA3: 311529dd998ee58138d52e6eaa7a0d8c1f2d4c3220d2c62bfc627e44804d61dc

File:
./contracts/uniswap/v2-periphery-master/libraries/UniswapV2OracleLibrary.sol
SHA3: fb79749da5b63abf3ca6fc47d0d70b1d73506e990a2f32dc93516b4dda02ba16

Fourth review scope

Repository	https://github.com/GovWorld/protocol-contracts/blob/contracts-v2
Commit	5ff534b40115cb68cf86c66af7960fb90b97c7e8
Whitepaper	Not provided
Functional Requirements	https://govworld.gitbook.io/govworld-docs/ SHA3:08dc38476a04b8c9052a9e628255741044261987b8caddd6aaec56d30ed7ac3e
Technical Requirements	https://govworld.gitbook.io/govworld-docs/ SHA3:08dc38476a04b8c9052a9e628255741044261987b8caddd6aaec56d30ed7ac3e

Contracts

File: ./contracts/facets/addressprovider/AddressProviderFacet.sol
SHA3: c0ce6e7fdf282998156bf963cfaa22121bc219f943205a5a70402009fa3fd4cd

File: ./contracts/facets/admin/AdminRegistryFacet.sol
SHA3: 6c146475b42f4513ef22aa2904056fbf12302a259e4f058e776a1adad83d7366

File: ./contracts/facets/admin/LibAdmin.sol
SHA3: 8bf4e9794a1c82c44df61e245c14f80b5133a1a84d5b30839583da128f83e06a

File: ./contracts/facets/admin/LibAdminStorage.sol
SHA3: 863565c73136041ef502f549a0bbb722216a8bffd28e38bb25c01b8d470f8d60

File: ./contracts/facets/claimtoken/ClaimTokenFacet.sol
SHA3: 6d88caf7b4a1e000ba32b294ab12be9bc4317a40fc0159d187192c8f9625b8b8

File: ./contracts/facets/claimtoken/LibClaimTokenStorage.sol
SHA3: ceed8eb984218c64766f04631261de063f2f038f215b0f1d2a8e469ab0be3680

File: ./contracts/facets/govTier/GovTierFacet.sol
SHA3: abe22669b8ae347eed48d5dc0cbad9d5f41e8670e9f3cad14eed62c89850a6bb

File: ./contracts/facets/govTier/LibGovTier.sol
SHA3: e485004776ad47e3a63b716bf30cb9524fc16037ec740019c54e3d71cc4a6620

File: ./contracts/facets/govTier/LibGovTierStorage.sol
SHA3: 42c149dcd3275c313f51516c1be0d4e2b17e55e3cfe71541d4afdf17e3abbfa4

File: ./contracts/facets/gTokenFactory/GTokenFactoryFacet.sol
SHA3: d6be7e8665a23099cf52ac11660e8ca6b22fa4614de4231029df08a4ff6b2766

File: ./contracts/facets/liquidator/LibLiquidator.sol
SHA3: b0327a7c73d8efa57b3de81e408c0a1ec08cedb6e5f50a6a7abb4037fbdb6f82

File: ./contracts/facets/liquidator/LibLiquidatorStorage.sol
SHA3: 1802184bee63168e2decd45bf518d6387f3360ffdea5a641a0c59cedd2aed14a

File: ./contracts/facets/liquidator/LiquidatorFacet.sol
SHA3: a4ebc431d94a4e8f0fd8bc73c3feee15a8c888c59161571add96dcffe49bc5d2

File: ./contracts/facets/market/libraries/LibMarketProvider.sol
SHA3: a4594a1392861d877a703b316628575e3326860ba533033e192f364338187ae4

File: ./contracts/facets/market/libraries/LibMarketStorage.sol
SHA3: 4e4609e41d69cfe287ea3fb8acb5199ba277d2a21a596d7e781d3038e591f684

File: ./contracts/facets/market/libraries/LibNetworkMarket.sol
SHA3: 6f68334864320b31af2d44ed9237737c9381fc10f3dba4a531addce230d05e73

File: ./contracts/facets/market/libraries/LibNFTMarket.sol
SHA3: 540ad0fb2b0fd049c2400476bf08518cce8c4c91523e2e9d69bc523d52bedeed

File: ./contracts/facets/market/libraries/LibTokenMarket.sol
SHA3: ceb1915b98b43dafecfe2c312127274d5565bf47eb4bc17c62ef6f538b0f6533

File: ./contracts/facets/market/network/NetworkMarketFacet.sol
SHA3: 485a87b39f2d656f6caa370616f7483dfaa71090202ee9f8e4ffe2600ec01c7d

File: ./contracts/facets/market/nft/NFTMarketFacet.sol
SHA3: bfdb1727cb892dbd36ab79f3e7c546bdcc3ff26a7a220cdd1056ef8f1cd1d67a

File: ./contracts/facets/market/token/TokenMarketFacet.sol
SHA3: c1369f652b6a080636fd6e958ad4d64cf3feb73cfcc0120b9d32d7c505932583

File: ./contracts/facets/marketRegistry/LibMarketRegistryStorage.sol
SHA3: fc6d019928fd5e403f6a03932c9ce9862ba2bbd126c35af714ffbb53f7cf389b

File: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol
SHA3: 0e72dc6922cb83a265fa849487905a90176ae59b55ff98fafd8718fc44192597

File: ./contracts/facets/nftTier/GovNFTTierFacet.sol
SHA3: 765b13afa38eccd224af4f197f6784ef138ede4ff7c03a173ca792c8596df01a

File: ./contracts/facets/nftTier/LibGovNFTTier.sol
SHA3: 0107c83b90fbe505920e9cfb5d01ea5c0b2c7d7950a467d689c66fb04f2f3cec

File: ./contracts/facets/nftTier/LibGovNFTTierStorage.sol
SHA3: 57fbc5448938b2c9d9bd92c19ba6d0f90dd3c14e0b2ffe7432d475cad36385c9

File: ./contracts/facets/oracle/LibPriceConsumer.sol
SHA3: 983c1ebd1be303920c0c6538fc0a96cd0dc64bb694477adcb2735e84d985a0e5

File: ./contracts/facets/oracle/LibPriceConsumerStorage.sol
SHA3: 59053a1027b4cf2136d0ae58f18e536680fd3f83710b07d462b5eb057e29d9c3

File: ./contracts/facets/oracle/PriceConsumerFacet.sol
SHA3: d51d56fa0bc1829ef2f9eb044949766b0db6f74d0e707ca6ba2a4653e5f089bb

File: ./contracts/facets/oracle/UniswapOracleV3.sol
SHA3: 4e8693026e79eacef097e64e5e8658828938624df72fbed97ecc6fd3b0bf3d3f

File: ./contracts/facets/pausable/PausableFacet.sol
SHA3: 54b9be28c9db0406ffe10080de00d3f5a6fbfba6b1921b9ab702bd8f0200f1d5

File: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol
SHA3: a49c5f975ccb2e55199377a8962ab0961acc4dddde781b458a2320eca248c976

File: ./contracts/facets/protocolRegistry/LibProtocolStorage.sol
SHA3: 88f6ab9e001c48d6b158fe8d67d9380ef274bad70c5e12f13d1c6d19ff1ef8c8

File: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol
SHA3: 30c07c93240071dc62f025493b1fe925bbb0b2eb662a8ccfdf7404fe41b447e3

File: ./contracts/facets/token/GToken.sol

SHA3:
6af29e423b9d873269d23248cf539a8cedddd703c96b2e84b7a9086a2d947774

File: ./contracts/facets/userTier/LibUserTier.sol
SHA3: c3b2d4d11f01a549363a9324baaad033923f93c0be25341d569ee12a76a30478

File: ./contracts/facets/userTier/UserTierFacet.sol
SHA3: fd76f5c470f3c490927226a921aeb2019e270f203bfcc2a45c2e241fc3f8adb4

File: ./contracts/facets/vcTier/LibVCTierStorage.sol
SHA3: e408b029d3e48b48c131a2a077c68674842efc65c6f81f82ce1e7d918d09e17d

File: ./contracts/facets/vcTier/VCTierFacet.sol
SHA3: 6a5e70d13031eb037c0f5be8ae117ee4e00f48c2c40ae2e64e489970fb5d7eab

File: ./contracts/interfaces/IAdminRegistry.sol
SHA3: 4c210c0f2e36ac9ff059849f397dcb8dfd391b08355e96b67cc33ddb0c8dea30

File: ./contracts/interfaces/IAggregationExecutor.sol
SHA3: 7e4452040f3ed384046ff0afe2039a8a9f85add0fdf1d3076a10c289563c40e7

File: ./contracts/interfaces/IAggregationRouterV5.sol
SHA3: 2dee60fc39da9d57be79b07ac184138211c481787d11adb89d2c3bc83635ba35

File: ./contracts/interfaces/IClaimToken.sol
SHA3: 570e891ed2481ba65ca3f863d8fbc45debcb2834025c3946cf4d7e661b08dc94

File: ./contracts/interfaces/IDexPair.sol
SHA3: 9cb7d3bfc87617d59f5013950261d3444dff6f5b1cafa3372ee1f4ac75edeaac

File: ./contracts/interfaces/IDiamondCut.sol
SHA3: 097ab56d21ca87e1260ed732d35b1928724f207337eaca41d89141bfb8349b99

File: ./contracts/interfaces/IDiamondLoupe.sol
SHA3: 90cdc2f3d119001a023cf3d1d600761551625f42d582ffad112b33fae29a161e

File: ./contracts/interfaces/IERC20Extras.sol
SHA3: e3b2fb98c8ba819670cf6f3eb5ae52630785f95876de1315971be7c240ba8be2

File: ./contracts/interfaces/IGovNFTTier.sol
SHA3: 67994d64fe466ea7041dfa014126242716e4cefa68cce69fd490a8d678f2a732

File: ./contracts/interfaces/IGovTier.sol
SHA3: 08ff9c36d7b95804ac9ded8a4daa49a74775c3d4a4b84961dab4748aa2bb2b91

File: ./contracts/interfaces/IGToken.sol
SHA3: cbaa60ed2fa70d80b0bdb6ea3a56083e53c12356f3ea060a2bf52784e5b76809

File: ./contracts/interfaces/IGTokenFactory.sol
SHA3: dd9eb048be64d11c424681672c0919fba35f4a81dd52d2724e75d9a5b87b5485

File: ./contracts/interfaces/IMarketRegistry.sol
SHA3: dffb84364fa3877f60f62bc1d41e89f6b18c2e998d9c4036540e85db84ae6aac

File: ./contracts/interfaces/IPriceConsumer.sol
SHA3: 3a783d69c184c4d2fb660d88c7d9087b91e569007c6f762f78194c43ec852623

File: ./contracts/interfaces/IProtocolRegistry.sol
SHA3: cfadc6d6399e1fe3acf759bb08cc3608457784882af5654b50b2e915e7e760d5

File: ./contracts/interfaces/IUniswapOracleV3.sol
SHA3: a7c687b9a13282189de94e4997fcc3856d7ad877bf150be03560c5f6f1f92922

File: ./contracts/interfaces/IUniswapSwapInterface.sol
SHA3: efd32d7d7dbb5dfb0c9ba6074a0367782af1ef9b2d5bc9c331c616cc43941a08

File: ./contracts/interfaces/IUniswapV2Router01.sol
SHA3: efc965b76796230d3891d4b1aad253d2f8505c2f93ceac848b80842a9e74c518

File: ./contracts/interfaces/IUniswapV2Router02.sol
SHA3: 62855cfb573a6af3a57d6ca56e9373514b025cc84dd1526610ab9d8fc19b37fb

File: ./contracts/interfaces/IUniswapV3Factory.sol
SHA3: 94e9658d22a17443e9239084c69b5564b05ea8b5035c6b53496701d234ae08d5

File: ./contracts/interfaces/IUniswapV3Router.sol
SHA3: ee81e5d470eaf442e4246dad1d675abd8321cff23dcea9c50478fa593ce5801c

File: ./contracts/interfaces/IUserTier.sol
SHA3: 4df71e8f1a1ee5d63bc1cd5760be4cdecfb9911a61801380fcd2b6df94436339

File: ./contracts/interfaces/IVCTier.sol
SHA3: 98f60723ac0c5b923c6e5a551823de956e3604fd3166cec3d181cefb15390eb6

File: ./contracts/shared/Diamond.sol
SHA3: 87992ecfcf9b825aa00c46e89652a3362c0a6b16ea271db81a0f0a55c64f468b

File: ./contracts/shared/facets/DiamondCutFacet.sol
SHA3: 05a8390cefbff7131d5b91798ab96f4641cecbf48905cf33cbea02b13245bd82

File: ./contracts/shared/facets/DiamondLoupeFacet.sol
SHA3: 8ad8c8c614cd363c1ab45b98878988eb65df965bc75d10c5d0b80460df43a469

File: ./contracts/shared/facets/OwnershipFacet.sol
SHA3: b248daedc25610b9353257c90ca8b51eb6802e2ae943ddbf8410c897e6f7d80e

File: ./contracts/shared/interfaces/IDiamondCut.sol
SHA3: 097ab56d21ca87e1260ed732d35b1928724f207337eaca41d89141bfb8349b99

File: ./contracts/shared/interfaces/IDiamondLoupe.sol
SHA3: 90cdc2f3d119001a023cf3d1d600761551625f42d582ffad112b33fae29a161e

File: ./contracts/shared/interfaces/IERC165.sol
SHA3: 2db7e91cf306335bccf3e9d84ece8a71d376f930daf61122b2571bd324d66a58

File: ./contracts/shared/interfaces/IERC173.sol
SHA3: 4f32c30a4b8c3894cd9276e0d9436910caa4b9ea09ca0ef292b77108dc55cf39

File: ./contracts/shared/libraries/LibAppStorage.sol
SHA3: 4ada2aa15a1f1fc36d05bcef0349e1da5615e3706a769675f87cb605be0971a3

File: ./contracts/shared/libraries/LibDiamond.sol
SHA3: e6e0a5cde2fabf2d742837a415bfc68730ba4edc102c0c10252a56efde6fc168

File: ./contracts/shared/libraries/LibMeta.sol
SHA3: 0a312736a8f5e05d2550bf689d4b1dc4d51b2ea70fe66afee2e708f8334140e7

File: ./contracts/shared/libraries/LibPausable.sol
SHA3: df534bfd71db25ead4893d7751422cdad9cbbd34baaea35964774e61ca32a7e4

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Migrator.sol
SHA3: 4a626e4ec0864c6eb6475dc93a0d18045f4883d028006ad90019dce0cdb374eb

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Router01.sol
SHA3: 0e8867b9a87d9df07eae068f2c342f090c77b4b4fed5e552f79022c8eae2e2a8

File: ./contracts/uniswap/v2-periphery-master/UniswapV2Router02.sol
SHA3: 02ef0ec1bf1d2057e66a4f9d56a62fc0b48f0d269229792d72321fead3cca9dd

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/AddressStringUtil.sol
SHA3: b9c4e63fcbf4a879c477d8c4fd347837d2668a4a2a6a1005d17ca9d1cd96a861

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/Babylonian.sol
SHA3: 6ea2bfd8b989ec7d616d22b725d5ff874f29be2a0533427d011725a48bf52e6b

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/BitMath.sol
SHA3: 34b174459a8734ddc07284797ba967ec5e91f6776e8db225943d0c395c79266a

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FixedPoint.sol
SHA3: 66966ce56c3063843982a3cde51bd0761afd1aa9a8e23dd98103e84afba729e6

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FullMath.sol
SHA3: 78a3e49791c5c474d49e88a681c1b9be90471c9f0218802f39b8701bdeff312c

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/SafeERC20Namer.sol
SHA3: 38b9bbadb9fce6ca34ccc1fe4e1190724b5f5c2a7d4b077c1ae2ce75985754ad

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/TransferHelper.sol
SHA3: c8b7cc0e76242f60dff0e224b7b27c4d8fae5477661789104d47cf59885f910e

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV2ERC20.sol
SHA3: 4ddb0763c4561a9ff3a52a7837aeb7dc88cbe79bcbe1d54659b97da8962959fa

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV2Factory.sol
SHA3: 0e2db8efa398c32bec215d16f11014140fe38e34e5577e2527cc2d7d68b06734

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/UniswapV2Pair.sol
SHA3: b669b00dba5bdbfa97da9e5b6607f0ee2c31ad3642dedff549c88404715e8300

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IERC20.sol
SHA3: 42e4b863a781d4cea0b2cb626d5bb58684d397a544fbf160dffc3a0148afc1d2

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2Callee.sol
SHA3: 56e42f76e0b6298b3b6d440efc13f2a670c2c85e013511277b74465f69a79f18

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/IUniswapV2ERC20.sol
SHA3: 04d4cc12d43a113c7b1e59fb15f18ae1c277563f79fa3d135ddf5f85659dc8b9

```
File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfac
es/IUniswapV2Factory.sol
SHA3: 1029024b2479a10eaf1a7c45507b63f1cc1212e5b1674892120f0410641271f3

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfac
es/IUniswapV2Pair.sol
SHA3: 1a67b3a6d7369f6936b5976379edd61d476e4295abca0c15405de1276a1e531b

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/librarie
s/Math.sol
SHA3: 931c3cb4dee32a740a8c2ac7f845de5ac733703652cfcf6a2d43d1faeb25e72b

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/librarie
s/SafeMath.sol
SHA3: 5a2ee0cd9bde270a65a7bbf231cab9bb4dc04f8f4d6faa4984f9951ee10d2200

File:
./contracts/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/librarie
s/UQ112x112.sol
SHA3: d2eccc238c08eb60a9068c02a5c01bca3566b878324684d93bb3570f02c67512

File: ./contracts/uniswap/v2-periphery-master/interfaces/IERC20.sol
SHA3: dcb2e043bde8a32f25fa7faa805041130e67d31bc8fad2ab5e568f4363290a5c

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Migrator.sol
SHA3: 715808ea0f1dde62f29190cf8a4b26c52c35fc21ba95799d8daf66634404f944

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Router01.sol
SHA3: ede34403bc4c23c55dbf1dd7bebdbec1a839d8983b8803d4c91a410e015d280f

File: ./contracts/uniswap/v2-periphery-master/interfaces/IUniswapV2Router02.sol
SHA3: 521ea3060bf020b52e6af0bbe77d50dd8f4faea788b1d4d15414d5ba48b3285a

File: ./contracts/uniswap/v2-periphery-master/interfaces/IWETH.sol
SHA3: 633ed9ff88e7d7d06b9341a98aa08717d936279af05aa35666f7d19b04b4d0a7

File: ./contracts/uniswap/v2-periphery-master/interfaces/V1/IUniswapV1Exchange.sol
SHA3: c81cfb4a356f2d0298a9bde96219d8b58c98b10161e60eb0003f0a6d62eda9bb

File: ./contracts/uniswap/v2-periphery-master/interfaces/V1/IUniswapV1Factory.sol
SHA3: b5e8b8c643e7ed5858e12fdf40c7be71da0325e12a44ecc29fdd889ab3aa368a

File: ./contracts/uniswap/v2-periphery-master/libraries/SafeMath.sol
SHA3: 8303fdecb7e19e705dffcd1716f127f8f970ad7b1046f049f36db66fa63fff15

File: ./contracts/uniswap/v2-periphery-master/libraries/UniswapV2Library.sol
SHA3: 311529dd998ee58138d52e6eaa7a0d8c1f2d4c3220d2c62bfc627e44804d61dc

File: ./contracts/uniswap/v2-periphery-master/libraries/UniswapV2OracleLibrary.sol
SHA3: fb79749da5b63abf3ca6fc47d0d70b1d73506e990a2f32dc93516b4dda02ba16
```

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect code quality

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 requirements are detailed.
- Technical description is robust.

Code quality

The total Code Quality score is **10** out of **10**.

Test coverage

Code coverage of the project is **92.31%** (branch coverage).

- Three tests are failing during the testing on the local hardhat environment.

Security score

As a result of the audit, the code contains **no** 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: **9.72**.

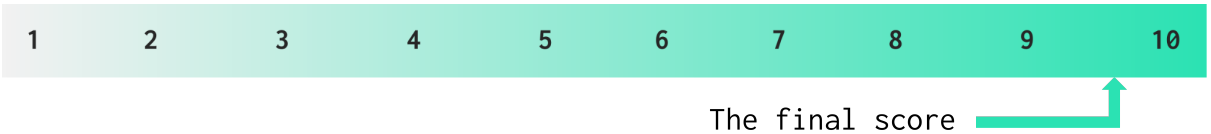

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
17 February 2023	27	37	30	12
11 April 2023	17	17	14	4
09 June 2023	4	1	5	0
21 July 2023	0	0	0	0

Risks

- The smart contracts in the system are upgradeable. This poses a risk that as it empowers the admin to modify the code, which can introduce vulnerabilities or alter the contract's functionality without consensus from other participants.
- There is no guarantee that the loans will be repaid by the borrowers. The liquidation may not fully cover the loan.
- The liquidation can not be guaranteed as it is manually triggered by the liquidators.
- The GTokens are automatically approved to the protocol when they are minted. The GToken can not be transferred, approved to the addresses except the whitelisted receivers.
- The GToken whitelisting functionality allows setting the whitelisted receivers. In case the address except the govDiamond is set, this may lead to the critical behavior described in the C05 issue: the locking of the loan repayment or liquidation.
- The user manually sets the price for each NFT collateral token without any validation by passing the corresponding `stakedNFTPrice` field value in the `LoanDetailsNFT` structure. The `stakedNFTPrice` is used to verify if the user has sufficient tier for the loan creation with the specified `loanAmount`. Users may create a loan and specify a high enough NFT price value to pass tier validation and set a high loan amount.

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items 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.	Not Relevant
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 required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
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 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	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 or be locked on the contract.	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 data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Passed
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 sufficient, 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, which may be changed in the future.	Passed

System Overview

GovWorld is a loan platform built on the EIP-2535 standard with the following contracts:
Base contracts:

- *Diamond* — is a contract that acts as a proxy in the system, delegates the call to the facets in its fallback function.
- *LibMeta* — is a library that contains the function for the msg.sender definition.
- *DiamondCutFacet* — is a facet that allows the system owner to perform diamond cuts (add, delete, replace function selectors).
- *DiamondLoupeFacet* — is a facet that allows to get all facets, each facet address, selectors, to get the facet by the selector that belongs to it.
- *OwnershipFacet* — is a facet that allows the system owner to transfer to ownership.
- *LibDiamond* — is a library that provides the definition of diamond storage: facets, selectors, system owner. The library provides functions that perform diamond cuts and set the owner address.
- *LibAppStorage* — is a library that provides the definition of diamond storage: Gov and Gov Gov token addresses.
- *Modifiers* — is a library that provides access check modifiers for the admins accesses, owner; non-paused system status.

Address provider functionality:

- *AddressProviderFacet* — is a facet that allows the owner to set Gov and Gov Gov token addresses.

Market registry functionality:

- *MarketRegistryFacet* — is a facet that allows super admin to set minimum loan amount allowed to create loan, loan limit amount for the activation, the LTV percentages limit, allowed multi collateral limit (10 maximum), 1inch aggregator address, set and update whitelisted for activation lenders addresses. The facet allows any user to obtain all the mentioned data (except 1inch aggregator address), to check if each address belongs to the all the whitelisted addresses and if it is whitelisted for the activation.
- *LibMarketRegistryStorage* — is a library responsible for the market registry storage. It stores the defined store position and the struct for it: allowed loan activation limit, minimal loan allowed amount, LTC percentage, multi collateral limit, all the whitelisted lenders addresses, whitelisted addresses statuses.

Protocol registry functionality:

- *LibProtocolStorage* — is a library responsible for the protocol registry storage. It stores Gov platform fee, Gov autosell fee, Gov threshold fee, approved tokens data (dex router address, gToken, is token to mint, token type (dex, elite, vip), is token enabled as

collateral), sp wallets for approved tokens, statuses of stable coins (enabled or not).

- *ProtocolRegistryFacet* – is a facet that allows administrators to manage the protocol registry. An owner can initialize the contract; when initializing the Gov platform fee is set to 2%, Gov autosell fee is set to 7%, Gov threshold fee is set to 2%.

  Superadmin can change the limits through this contract within the defined limits: Gov platform fee <= 20%, Gov threshold fee <= 50%, Gov autosell fee <= 20%.

  The facet allows to check if token approved, enabled as collateral, if sp wallet added to token, to obtain all the approved tokens addresses, each approved token data (dex router address, gToken, is token to mint, token type (dex, elite, vip), is token enabled as collateral), to check if stable coin approved, to get Gov platform fee, Gov autosell fee, Gov threshold fee, Sps wallet added to each token, to check if synthetic mint option is on for the approved collateral token.

  The facet allows an admin with editToken permission to set statuses of stable coins, to allow and to disallow tokens as collateral, to update data of approved tokens that have VIP type and not set g Token; an admin with the addToken permission to allow tokens and add their data: if token type is vip, its *gToken* is deployed and is token to mint indicator is set to the defined by an admin value; an admin with the addSp permission to add Sps wallets to the token if its type is vip (one or bulk); an admin with the editSp permission to remove or update Sps wallets to the token if its type is vip (one or bulk);
- *LibProtocolRegistry* – is a library that provides helper functionality for the *ProtocolRegistryFacet*. It provides functions for adding, updating tokens, adding, updating and removing Sps.
  It contains events emitted in the *ProtocolRegistryFacet.*

Gov tier functionality:

- *LibGovTierStorage* – is a library responsible for the Gov tier storage. It defines the storing of tier levels (information for each tier level: Gov holdings, ltv percentage of the Gov holdings, Gov intel indicator, if tier is for single token, if tier is for multiply tokens, if tier is for NFT token, if tier is for NFT tokens, if tier is for reverse loan), tiel lever of each address, all addresses that have tiers, address provider, indicator if the Gov tier is initialized.
- *GovTierFacet* – is a facet that allows the management of the Gov tiers data.
  The owner can initialize the Gov tier: the if initialized Gov tier is set to true and the following tiers are added:
  - bronze:
    - Gov holdings: 15000e18
    - Ltv percentage of the Gov holdings: 30

        - Gov intel indicator: false
        - If tier is for single token: true
        - If tier is for multiply tokens: false
        - If tier is for NFT token: true
        - If tier is for NFT tokens: false
        - If tier is for reverse loan: false
    - silver:
        - Gov holdings: 30000e18
        - Ltv percentage of the Gov holdings: 40
        - Gov intel indicator: false
        - If tier is for single token: true
        - If tier is for multiply tokens: true
        - If tier is for NFT token: true
        - If tier is for NFT tokens: true
        - If tier is for reverse loan: false
    - gold:
        - Gov holdings: 75000e18
        - Ltv percentage of the Gov holdings: 50
        - Gov intel indicator: true
        - If tier is for single token: true
        - If tier is for multiply tokens: true
        - If tier is for NFT token: true
        - If tier is for NFT tokens: true
        - If tier is for reverse loan: true
    - platinum:
        - Gov holdings: 150000e18
        - ltv percentage of the Gov holdings: 70
        - Gov intel indicator: true
        - If tier is for single token: true
        - If tier is for multiply tokens: true
        - If tier is for NFT token: true
        - If tier is for NFT tokens: true
        - If tier is for reverse loan: true

The facet functionality allows admin with edit tier level role to add tier, its Gov holdings value should be less than the govToken total supply and bigger than all the other tiers` Gov holdings values; and to update the tier, the Gov holdings value should be between previous and next tiers levels` values. The functionality allows to save tier levels: update if tier exists and add if it does not.

The super admin can set and update the tier level for each user.

It allows to get all the tiers levels keys, each tier level data (Gov holdings, ltv percentage of the Gov holdings, Gov intel indicator, if tier is for single token, if tier is for multiply tokens, if tier is for NFT token, if tier is for NFT tokens, is for reverse loan), each

wallet tier, the wallet tiers of all the addresses, check if address has a tier level.

The maximal amount of tiers is 30.

- *LibGovTier* – is a library that provides functionality for managing *LibGovTierStorage*, used in the *GovTierFacet*: to add, update, save and remove tiers; to check if tier is added and get tier index.

NFT tier functionality:

- *LibGovNFTTierStorage* – is a library responsible for the Gov NFT tier storage. It defines the storing of NFT and Sp tier levels (information for each Sp tier level: Gov ltv percentage, if tier is for single token, if tier is for multiple tokens, if tier is for single NFT token, if tier is for multiple NFT tokens; information for each NFT tier level: NFT contract address, if tier is traditional, Sp token address, traditional (Gov) tier id, allowed NFTs for tier addresses, allowed Sun tokens for tier addresses), all the NFT tier levels keys, all the Sp tier levels keys.
- *GovNFTTierFacet* – is a facet that allows the management of the NFT and Sun tiers data.
  The facet functionality allows admin with edit tier level role to add and update Sp tier, its ltv value should be greater than 0; to remove Sp and Nft tiers.

  The super admin is allowed to add NFT tier level, if its traditional indicator is set to true, it can be traditional (its traditional tier level is checked through the *LibGovTier* for existence) or SP (defined SP`s tier ltv percentage should be bigger than 0); traditional and SP can not be combined. Other tier data added: allowed NFTs and Sun tokens addresses. Sp tier data: Gov ltv percentage, if tier is for single token, if tier is for multiple tokens, if tier is for single NFT token, if tier is for multiple NFT tokens. The super admin can add NFTs and Sun tokens to Nft tiers.

  The facet allows to get all the Sp tiers keys, all the NFT tiers keys, the amount of the NFT tiers, each Nft address` tier level, and each Sp tier data.

  The facet functionality allows to get the user`s tier: all the NFTs tiers are looped and that is owned by the user and has the bigger LTV (Gov tier ltv for traditional tier, and Sp ltv for Sp tier) is chosen.

  The tiers and allowed tokens of each type are limited to 30 per each.
- *LibGovNFTTier* – is a library that provides helper functionality for the *LibGovNFTTierStorage*, used in the *GovNFTTierFacet*: to remove Sp and Nft tier keys, to get Sp and NFT tier keys indexes, to check if NFT tier created.

VC tier functionality:

- *LibVCTierStorage* – is a library responsible for the VC tier storage. It defines the storing of VC tier levels (information for each VC

tier level: NFT contract address, traditional (Gov) tier id, allowed NFTs for tier addresses, allowed Sun tokens for tier addresses), all the VC tier levels keys.

- *VCTierFacet* – is a facet that allows the management of the NFT and Sun tiers data.
  The contract functionality allows super admin to add VC tiers, to add Sp and NFT tokens to them.

  It allows users to obtain each tier information (NFT contract address, traditional (Gov) tier id, allowed NFTs for tier addresses, allowed Sun tokens for tier addresses); to check if tier added; to get user`s tier: all the NFTs tiers are looped and that is owned by the user and has the bigger LTV (Gov tier ltv of traditional tier) is chosen.

  The maximal amounts of Sp and NFT tokens for each tier are 30 per each.

User tier functionality (gathers all the tiers functionality):

- *UserTierFacet* – is a facet that allows users to obtain and verify their tiers.
  The contract functionality allows to obtain each user` Gov tier: the user`s Gov plus Gov Gov tokens balance is obtained, if it is bigger than the required amount for the least tier, the tier is obtained according to the balance, otherwise, set by admin tier for user is obtained from *GovTierFacet*.

  The contract contains a function that allows to obtain the maximal loan amount using the tier level percentages and collateral token amount in stable and using the tier defined by the user with the verification that the user has it.
  There is a function that checks if the token or NFT loan to be created is under user` tier (defined by user type).
- *LibUserTier* – is a library that provides functions for the validations if the loans are under tiers, used in the *UserTierFacet*.
  - Gov tier for ERC-20 tokens: checks if only single token loan allowed, if so, validates if staked collateral token number is 1, otherwise, reverts. Checks if loan amount is not greater than maximal loan amount according to collateral amount is stable and tier ltv.
  - Gov tier for ERC-721 tokens: checks if only single NFT token loan allowed, if so, validates if staked collateral NFT token number is 1, otherwise, reverts. Checks if loan amount is not greater than maximal loan amount according to collateral amount is stable and tier ltv.
  - NFT traditional tier for ERC-20 tokens: executes the Gov tier for ERC-20 tokens validation using the related traditional Gov tier.

- NFT traditional tier for ERC-721 tokens: executes the Gov tier for ERC-721 tokens validations using the related traditional Gov tier.
- NFT Sp tier for ERC-20 tokens: validates if multi tokens loan is allowed, if not, checks if staked token is 1; checks if all the staked collateral tokens belong to the allowed in the tier Sun tokens or Sp token. Checks if loan amount is not greater than maximal loan amount according to collateral amount is stable and tier ltv.
- NFT Sp tier for ERC-721 tokens: validates if multi NFT loan is allowed, if not, checks if staked NFT token is 1; checks if all the staked collateral tokens belong to the allowed in the tier NFT tokens. Checks if loan amount is not greater than maximal loan amount according to collateral amount is stable and tier ltv.
- VC tier for ERC-20 tokens: checks if only single token loan allowed, if so, validates if staked collateral token number is 1, otherwise, reverts; checks if all the staked collateral tokens belong to the allowed in the tier Sp tokens. Checks if loan amount is not greater than maximal loan amount according to collateral amount is stable and tier ltv.
- VC tier for ERC-721 tokens: checks if only single NFT token loan allowed, if so, validates if staked collateral token number is 1, otherwise, reverts; checks if all the staked collateral NFT tokens belong to the allowed in the tier Sp tokens. Checks if loan amount is not greater than maximal loan amount according to collateral amount is stable and tier ltv.

All the validating functions return *200* in case of success.

Pausing functionality:

- *PausableFacet* – is a facet that allows the management of the system pause status. It allows the system owner to pause and unpause the system. The pausing stops creating, updating, canceling and activating all the loans and the NFT loan payback before the term ends.
- *LibPausable* – is a library responsible for the system pause status storage definition and its management. It provides functions that allow to change the status, and the functionality that checks if the system is not paused.

Admin registry functionality:

- *LibAdminStorage* – is a library responsible for the admins registry storage definition. It defines the storing of approved admins and their permissions, super admin address, pending admins and the admins that approved each pending admin, pending information (pending keys). The permissions that an admin may have:
  - add Gov Intel,
  - edit Gov Intel,
  - add token,

  - edit token,
  - add Sp,
  - edit Sp,
  - add Gov admin,
  - edit Gov admin,
  - add Bridge,
  - edit Bridge,
  - add pool,
  - edit pool,
  - to be super admin.
- *AdminRegistryFacet* – is a facet that allows the management of the NFT and Sun tiers data.

  The system owner may initialize the contract. When the initialization, the admins with the following permissions are added: 1 super admin and 3 admins with all the permissions except the super admin; and the pending keys are set (add key: 0, edit key: 1, remove key: 2).

  The super admin may transfer the super admin permission to the other admin.

  The admins with the add Gov admin permission may add a new admin, then 51% of all the other admins with the add Gov admin permission should approve the addition.

  The admins with the edit Gov admin permission may edit and remove the admins, then 51% of all the other admins with the edit Gov admin permission should approve the edition or the removal. The admins with the edit Gov admin permission may reject the pending for adding, edition or removal admin if they have not approved them before.

  Only one pending admin at once is allowed.

  The contract allows to obtain pending admin keys, each admin permission, to check if the user has each permission, each pending for adding, editing of removal admin permissions, admins who approved each pending address, all the approved admins.

  The maximum number of admins is 30.
- *LibAdmin* – is a library that provides helper functionality for the *LibAdminStorage*, used in the *AdminRegistryFacet*: to make admins pending for removal, adding or edition; to remove, add or edit them.

Claim token functionality:

- *LibClaimTokenStorage* – is a library responsible for the claim tokens storage definition. It defines the storing of approved claim tokens and their data (token type, peg/sun tokens, peg/sun tokens percentages, dex router), sun tokens, and claim token of each sun token.
  The maximal amount of sun tokens of each claim token is 30.

- *ClaimTokenFacet* – is a facet that allows the management of the claim tokens.
  The super admin may add claim tokens information, update it, enable and disable claim tokens.
  The facet provides the functionality for obtaining each claim token data (token type, peg/sun tokens, peg/sun tokens percentages, dex router), claim token of each sun token, to check if claim token is enabled.

Gtokens functionality:

- *GTokenFactoryFacet* – is a facet that allows to deploy the *GToken* based on the sp token, used in the *ProtocolRegistryFacet*: deploys the G token for each Sp token which has VIP type. It sets the token name to "gov" + sp token name, the token symbol to "gov" + sp token symbol, transfers token ownership to the Diamond.
- *GToken* – is an ERC-20 token. Its name and symbol are defined when the deployment. It is deployed on base of Sp token, its maximal token supply is limited by the Sp token total supply, its decimals are obtained from the Sp token.

  The diamond address (set when the deployment) is allowed to mint tokens. When tokens are minted, they are approved to the contract owner.

  The token has a whitelist for the token receivers. The whitelist is managed by the system super admin. Tokens can be transferred and approved only to the whitelisted receivers. The *approve* function acts like an *increaseAllowance* function.

Markets functionality:

- *TokenMarketFacet* – is a facet that allows to create token loans and activate them.
  When the loan creation the user defines loan details: loan amount, stable coin to be borrowed address, loan terms in days, apy percentage, loan type (single token or multi token), collateral tokens addresses and amounts, tier type (Gov tier, NFT traditional tier, NFT sp tier, Vc tier), is loan private, is loan insured.
  The stable token to be borrowed is verified for being approved in the *ProtocolRegistryFacet*, collateral tokens number is being checked for non-exceeding the multi collateral limit defined in the *MarketRegistryFacet*, the amount to be borrowed is checked against the minimum loan amount defined in the *MarketRegistryFacet*; if the loan type is set to single collateral, the collaterals tokens number is checked for equality with the 1, it is checked if all the collateral tokens are allowed to create loans in the *ProtocolRegistryFacet* (and they may be claim tokens) with and are approved to the contracts in the defined amounts by the borrower. The collateral tokens are calculated in the collateral tokens and ltv is calculated and checked for being greater than the defined in the *MarketRegistryFacet* ltv percentage. The collateral tokens conversation calculation is

performed in the *LibTokenMarket* through the *PriceConsumerFacet*: all the token prices are summed upped: if the token has claim token, its claim token price is obtained from the defined in the claim token data dex and the defined sun/peg percentage of it gets, if the token does not have the claim token, its price is obtained from the Chainlink, the ltv is the percentage of this total value from the loan amount. The loan is checked for being the defined tier throughout the *UserTierFacet* using the obtained collateral in borrowed token amount. If all the verifications succeed, the loan offer is created with the INACTIVE status.

The borrowers may update their loan offers (amount to be borrowed, term length in days, apy offer, is private, is insured) if they are in the INACTIVE status, the data for the loan is verified in the same way as for the loan creation.

The borrowers may cancel their loan offers if they are in the INACTIVE status.

The liquidators may cancel the loans if they are in the INACTIVE status.

Canceling sets the loan status to CANCELLED.

Users may activate the loan that has INACTIVE status. When the activation, the loan status is set to ACTIVE. The borrowers can not activate their own loan offers. If the users are whitelisted for activation in the *MarketRegistryFacet*, they can activate an unlimited number of times; otherwise, they are limited with the loan activation limit defined in the *MarketRegistryFacet*. Users may activate many loan offers per time, defining the autosell indicator for each one. If the loan collateral token has claim token defined in the ClaimTokenFacet, the autosell can not be applied. The autosell indicates the type of the collateral refund to the user when the liquidation, if it's set to true: the collateral tokens are converted to borrowed stable token, if it's set to false, the collateral are paid in the tokens themselves. The max loan amount and LTV are recalculated, the loan activator defines the loan amount to be borrowed. If the new maximal loan amount is less than the loan amount to be borrowed defined in the borrow offer, the defined by the lender loan amount is set (should be equal or less for 3% than the maximal loan amount). The loan amount is transferred to the contract from the lended. The loan (the apy and the platform fee (its percentage is set in the ProtocolRegistryFacet) is deducted from it) is transferred to the borrower. The collateral tokens are transferred from the borrowed to the contract, if the token has synthetic mint on (ProtocolRegistryFacet), the appropriate GToken is minted to the borrower.

The facet allows to obtain the information for each loan offer, for each activated loan and the LTV calculations (ltv, maximal loan amount and the collateral tokens in the borrowed coin amount) for the defined collateral tokens, borrow stable coin, loan amount, borrower and tier type.

- *LibTokenMarket* – is a library that provides helping functionality for the *TokenMarketFacet*. It provides functions that check collateral approvals, provides the ltv calculations, transfer collateral tokens and mints synthetic.
- *LiquidatorStorage* – is a library responsible for the liquidator storage definition. It initializes the storage of the whitelisted liquidators, liquidators Sun token balances per each wallet, 1inch aggregator address, if the liquidator initialized.
- *LiquidatorFacet* – is a facet that provides functionality for the token loan repayments and liquidations.

  The system owner can initialize the contract, when the initialization, 2 provided addresses are set as liquidators.

  Superadmin is allowed to set the liquidators' statuses.
  The facet provides the functionality for obtaining all the liquidators and each liquidator status (enabled or not), total payment amount for each token loan id, if the liquidation is pending for each token id, liquidated sun tokens balances for each user and each sun token, the withdrawable amount of stable coins and collateral tokens.

  The liquidators can approve tokens to 1inch.

  The liquidators can liquidate the loan if it was not fully repaid and the loan terms in days + 1 day have passed. After the liquidation, the loan status is set to LIQUIDATED. Depending on the fact if the autosell is on for the loan, the liquidation differs:

  For the turned-on autosell: the collaterals are swapped to the borrowed coin through the 1inch, if each collateral is to mint, the appropriate amount of appropriate Gtoken is burned from the borrower. The apy fee is recalculated and if there is an unearned apy fee, the rest is left on the platform. The difference between the final swapped amount and the borrowed amount is left in the protocol, the autosell fee (autosell fee percentage is defined on the Market facet) is deducted from the amount and then the resulting amount is transferred to the lender.

  For the turned-off autosell: each collateral token price in the borrowed token is obtained. If the token has the claim token (*ClaimTokenFacet*), its claim token price is obtained from the defined in the claim token data dex and the defined sun/peg percentage of it is obtained. If each collateral were to mint, the appropriate amount of appropriate Gtoken would be burned from the borrower. The collateral tokens are transferred to the lender until their total

price in the borrowed token reaches the borrowed amount + threshold fee (threshold fee percentage defined in the *ProtocolRegistryFacet*). The rest of the collateral are left in the protocol. The apy fee is recalculated and if there is an unearned apy fee, the rest is left on the platform. The initial apy fee is transferred to the lender.

The facet allows borrowers to pay back their token loans if ltv is bigger than the ltv percentage defined in the MarketRegistryFacet and the loan term in days plus 1 day have not reached. The borrower may pay the loan partially, if the remaining unpaid amount is sufficient for the ltv calculation. If the borrower fully repays the loan, the collateral tokens are transferred to the borrower, and the minted GTokens are burned from the borrower. The loan and earned apy fee are transferred to the lender, an unearned apy remains in the protocol.

The facet allows super admin to withdraw stablecoins that remained as a platform fee when the loan activation and the collaterals that remained in the contract as exceeded amount when swapping tokens for the autosell liquidation.

- *LibLiquidator* – is a library that provides the functionality for the *LiquidatorFacet*.

  It provides the functions that set liquidators' statuses, liquidate the token loan with autosell, liquidate the token loan without autosell, calculate the ltv, check if the liquidation of the each loan is pending, calculate the total payback amount for each token loan, fully payback the loan.
- *NFTMarketFacet* – is a facet that allows to create NFT loans, activate, repay and liquidate them.
  When the loan creation the user defines loan details: loan amount, stable coin to be borrowed address, loan terms in days, apy percentage, loan type (single token or multi token), NFT collateral tokens addresses, their ids and prices, tier type (Gov tier, NFT traditional tier, NFT sp tier, Vc tier), is loan private, is loan insured.

  The stable token to be borrowed is verified for being approved in the *ProtocolRegistryFacet*, collateral tokens number is being checked for non-exceeding the multi collateral limit defined in the *MarketRegistryFacet*, the amount to be borrowed is checked against the minimum loan amount defined in the *MarketRegistryFacet*; if the loan type is set to single collateral, the collaterals tokens number is checked for equality with the 1. The collateral tokens prices (defined by the borrower) are summed up and the loan is checked for being the defined tier throughout the *UserTierFacet*. If all the verifications succeed, the loan offer is created with the INACTIVE status.

The borrowers may cancel their loan offers if they are in the INACTIVE status.

The liquidators may cancel the loans if they are in the INACTIVE status.

Canceling sets the loan status to CANCELLED.

The borrowers may update their loan offers (amount to be borrowed, term length in days, apy offer, is private, is insured) if they are in the INACTIVE status, the data for the loan is verified in the same way as for the loan creation.

Users may activate the loan that has INACTIVE status. When the activation, the loan status is set to ACTIVE. The borrowers can not activate their own loan offers. If the users are whitelisted for activation in the *MarketRegistryFacet*, they can activate an unlimited number of times; otherwise, they are limited with the loan activation limit defined in the *MarketRegistryFacet*. The loan amount is transferred to the contract from the lended. The loan (the apy and the platform fee (its percentage is set in the ProtocolRegistryFacet) is deducted from it) is transferred to the borrower. The collateral tokens are transferred from the borrowed to the contract.

The liquidators can liquidate the loan if it was not fully repaid and the loan terms in days + 1 day have passed. After the liquidation, the loan status is set to LIQUIDATED. The collaterals are transferred to the lender. The initial apy fee is transferred to the lender.
The facet allows to obtain the information for each loan offer, for each activated loan.

- *LibNFTMarket* — is a library that provides the helper functionality for the *NFTMarketFacet*.

  It provides the functions that check NFTs approvals and transfer them.
- *NFTMarketFacet* — is a facet that allows to create NFT loans, activate, repay and liquidate them.
- *NetworkMarketFacet* — is a facet that allows to create token loans, activate, repay and liquidate them. The collaterals are paid in ETH.
  When the loan creation the user defines loan details: loan amount, stable coin to be borrowed address, loan terms in days, apy percentage, is loan private, is loan insured.

  The amount to be borrowed is checked against the minimum loan amount defined in the *MarketRegistryFacet*; the borrowed stable is checked for being approved in the *ProtocolRegistryFacet*. The collaterals coins are transferred to the contract, the ltv is checked for being equal to or greater than the minimum ltv defined in the *MarketRegistryFacet*. The loan is checked for being not greater than the maximal allowed user loan calculated according to the user`s tier` ltv by gov balance. If all the verifications succeed, the ltv

is calculated due to the borrowed coin price in wETH, the loan offer is created with the INACTIVE status.

The borrowers may update their loan offers (amount to be borrowed, term length in days, apy offer, is private, is insured) if they are in the INACTIVE status, the data for the loan is verified in the same way as for the loan creation.

The borrowers may cancel their loan offers if they are in the INACTIVE status. The collaterals are then transferred back to them. Canceling sets the loan status to CANCELLED.

Users may activate the loan that has INACTIVE status. When the activation, the loan status is set to ACTIVE. The borrowers can not activate their own loan offers. If the users are whitelisted for activation in the *MarketRegistryFacet*, they can activate an unlimited number of times; otherwise, they are limited with the loan activation limit defined in the *MarketRegistryFacet*. The autosell indicates the type of the collateral refund to the user when the liquidation, if it is set to true: the collateral ETH is converted to borrowed stable token, if it is set to false, the collateral is paid in the ETH coins themselves. The max loan amount and LTV are recalculated, the loan activator defines the loan amount to be borrowed. If the new maximal loan amount is less than the loan amount to be borrowed defined in the borrow offer, the defined by the lender loan amount is set (should be equal or less for 3% than the maximal loan amount). The loan amount is transferred to the contract from the lender. The loan (the apy and the platform fee (its percentage is set in the ProtocolRegistryFacet) is deducted from it) is transferred to the borrower.

The facet allows borrowers to pay back their token loans if ltv is bigger than the ltv percentage defined in the *MarketRegistryFacet* and the loan term in days plus 1 day have not reached. The ltv is calculated due to the borrowed coin price in wETH. The borrower may pay the loan partially, if the remaining unpaid amount is sufficient for the ltv calculation. If the borrower fully repays the loan, the collateral coins are transferred to the borrower, and the minted *GTokens* are burned from the borrower. The loan and earned apy fee is transferred to the lender, an unearned apy remains in the protocol.

The liquidators can liquidate the loan if it was not fully repaid and the loan terms in days + 1 day have passed. After the liquidation, the loan status is set to LIQUIDATED. Depending on the fact if the autosell is on for the loan, the liquidation differs:

For the turned-on autosell: the collateral ETH is swapped to the borrowed coin through the 1inch. The apy fee is recalculated and if there is an unearned apy fee, the rest is left on the platform. The difference between the final swapped amount and the borrowed amount is left in the protocol, the autosell fee (autosell fee percentage is

defined on the Market facet) is deducted from the amount and then the resulting amount is transferred to the lender.

For the turned-off autosell: each collateral token price in the borrowed token is obtained. If the token has the claim token (*ClaimTokenFacet*), its claim token price is obtained from the defined in the claim token data dex and the defined sun/peg percentage of it is obtained. If each collateral was to mint, the appropriate amount of appropriate Gtoken is burned from the borrower. The collateral tokens are transferred to the lender until their total price in the borrowed token reaches the borrowed amount + threshold fee (threshold fee percentage defined in the *ProtocolRegistryFacet*). The rest of the collateral is left in the protocol. The apy fee is recalculated and if there is an unearned apy fee, the rest is left on the platform. The initial apy fee is transferred to the lender.

The facet allows super admin to withdraw exceeding coins.

The facet allows to obtain each created and activated loan information, the withdrawable exceeding coins amount, the amount of activated by user loans, total payback amount, check if the liquidation for the loan is pending, get the loan ltv, maximal loan amount according to the collateral amount and the borrower address.

- *LibNetworkMarket* – is a library that provides the helper functionality for the *NetworkMarketFacet*.

  It provides the functions that calculate payback amount, ltv, max loan value and check if the liquidation is pending.
- *LibMarketProvider* – is a library that provides the helper functions that calculate the resulting apy fee according to the loan amount, apy fee, loan term in days.
- *LibMarketStorage* – is a library that defines the storage of market data for each of Token, NFT and Network markets: current loan id, loan details for each loan id, details for each activated loan id, borrowers` loans, lenders` loans; and the common data: activated amount of loans be each user, withdrawable coins and tokens, liquidated sun token balances for each sun token and each user.

Prices obtaining functionality:

- *PriceConsumerFacet* – is a facet that provides tokens prices from Chailink and Uniswap in order to calculate the collaterals values, the facet calculates the ltv value according to the collateral tokens prices, borrowed token and their amounts.
- *LibPriceConsumer* – is a library that provides the helper functions for the *PriceConsumerFacet*. It allows to convert the pair to stable, get the pair and the reserves, and check if Chainlink feed has been added.
- *LibPriceConsumerStorage* – is a library that defines the storage of price consumer: Chainlink data feeds and Uniswap router.

- *IAdminRegistry, IClaimToken, IDexFactory, IDexPair, IDiamondCut, IDiamondLoupe, IERC20Extras, IGovNFTTier, IGovTier, IGToken, IGTokenFactory, IMarketRegistry, IPriceConsumer, IProtocolRegistry, IUniswapSwapInterface, IUniswapV2Router01, IUniswapV2Router02, IUserTier, IVCTier* – are the system contracts interfaces.
- *AddressStringUtil*, *Babylonian*, *BitMath*, *FixedPoint*, *FullMath*, *SafeERC20Namer*, *TransferHelper*, *Math*, *SafeMath*, *UQ112x112*, *UniswapV2ERC20*, *UniswapV2Factory*, *UniswapV2Pair*, *IUniswapV1Exchange*, *IUniswapV1Factory*, *SafeMath*, *UniswapV2Library*, UniswapV2OracleLibrary, UniswapV2Migrator, UniswapV2Router01, UniswapV2Router02 – are the contracts for the uniswap functionality.
- *IERC20 (/uniswap/v2-periphery-master/interfaces/), IUniswapV2Callee, IUniswapV2ERC20, IUniswapV2Factory, IUniswapV2Pair, IERC20 (/uniswap/v2-periphery-master/dependencies/uniswap-v2-core/contracts/interfaces/), IUniswapV2Migrator, IUniswapV2Router01, IUniswapV2Router02, IWETH* – are the contracts for the Uniswap functionality`s interfaces.

Privileged roles

- The owner of the system is allowed to:
  - *AddressProviderFacet* - set Gov and Gov Gov tokens addresses.
  - *ProtocolRegistryFacet* - initialize the contract, update data of approved tokens that have VIP type and not set g Token.
  - *AdminRegistryFacet* - initialize the contract.
  - *PausableFacet* - pause and unpause the system.
  - *LiquidatorFacet* - initialize the contract, withdraw stablecoins that remained as a platform fee when the loan activation and the collaterals that remained in the contract as exceeded amount when swapping tokens for the autosell liquidation.
  - *DiamondCutFacet* - perform diamond cuts.
  - *OwnershipFacet* - transfer ownership to another address.
  - *PriceConsumerFacet* - initialize the contract, set swap router.
- The Diamond is allowed to:
  - *GToken* - may mint tokens.
- The super admin of the system is allowed to:
  - *MarketRegistryFacet* - set the minimum loan amount allowed to create loan, loan limit amount for the activation, the LTV percentages limit, allowed multi collateral limit, 1inch aggregator address, set and update whitelisted for activation lenders addresses.
  - *ProtocolRegistryFacet* - set the limits within the defined limits: Gov platform fee <= 20%, Gov threshold fee <= 50%, Gov autosell fee <= 20%.

- *GovTierFacet* - set and update the tier lever for each user.
- *GovNFTTierFacet* - add NFT tier levels, add NFTs and Sun tokens to NFT tiers.
- *VCTierFacet* - add VC tier levels, add NFTs and Sun tokens to VC tiers.
- *AdminRegistryFacet* - transfer super admin permission to another admin.
- *ClaimTokenFacet* - add claim tokens information, update it, enable and disable claim tokens.
- *LiquidatorFacet* - set liquidators, withdraw exceeding tokens and coins obtained in the NFT and Token markets.
- *NetworkMarketFacet* - withdraw exceeding coins.
- *GToken* - set whitelisted receivers.

- The admins with the *editToken* permission are allowed to:
  - *ProtocolRegistryFacet* - set statuses of stable coins, change is mint token status.
- The admins with the *addToken* permission are allowed to:
  - *PriceConsumerFacet* - add tokens chainlink feeds, update price aggregators.
  - *ProtocolRegistryFacet* - add tokens.
- The admins with the *addSp* permission are allowed to:
  - *ProtocolRegistryFacet* - add Sps wallets to the token if its type is vip (one or bulk).
- The admins with the *editSp* permission are allowed to:
  - *ProtocolRegistryFacet* - remove or update Sps wallets to the token if its type is vip (one or bulk).
- The admins with the *EditTierLevel* permission are allowed to:
  - *GovTierFacet* - add, update, remove and save (update or add) tier levels.
  - *GovNFTTierFacet* - add and update SP tier levels, remove Sp and NFT tiers.
- The admins with the *addGovAdmin* permission are allowed to:
  - *AdminRegistryFacet* - add new admins and approve their addition.
- The admins with the *editGovAdmin* permission are allowed to:
  - *AdminRegistryFacet* - update and remove admins and approve their updating and removal, reject the pending for adding, removal of updating admins.
- The liquidators are allowed to:
  - *TokenMarketFacet* - cancel the offer loans with the *INACTIVE* status.
  - *NFTMarketFacet* - cancel the offer loans with the *INACTIVE* status, liquidate the loans.
  - *NetworkMarketFacet* - liquidate the loans.

- *LiquidatorFacet* - approve tokens to 1inch, liquidate the token (*TokenMarketFacet*) loans.

Findings

■■■■ Critical

C01. Access Control Violation

The super admin access verification in the *setWhitelistReceiver* function is commented.

Therefore, any users may manage the token whitelist and define which users can receive tokens.

Path: ./contracts/facets/token/GToken.sol : setWhitelistReceiver()

Recommendation: uncomment the access check verification.

Status: Fixed (Revised commit: ed54c7c)

C02. Data Consistency

When removing the sp wallet from the *approvedSps* mapping in the *ProtocolRegistryFacet.removeSp* function (*LibProtocolRegistry._removeSpKeyfromMapping*), the index to be deleted is determined via the *LibProtocolRegistry._getIndexofAddressfromArray* function, which obtains the index from the *allApprovedSps* array.

Therefore, the *ProtocolRegistryFacet.removeSp* will always remove the first value from the *approvedSps* mapping instead of the correct one. (As the *_getIndexofAddressfromArray* will always return *0* index at that moment of execution).

The same is applicable to *ProtocolRegistryFacet.removeBulkSps* *and* *LibProtocolRegistry._updateSp*.

Paths: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : removeSp(), removeBulkSps()

./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _updateSp()

Recommendation: ensure that the correct wallet addresses are deleted when removing and updating Sp wallets.

Status: Fixed (Revised commit: ed54c7c)

C03. Data Consistency

When updating Sp wallet in the *LibProtocolRegistry._updateSp* function, instead of removing the old wallet address (*_oldWalletAddress*) all the *s.approvedSps[_tokenAddress]* are removed in the loop.

Therefore, the *LibProtocolRegistry._updateS* will always remove the token Sp wallets.

The *_newWalletAddress* is added in the loop per each iteration.

Due to this *_newWalletAddress* will be added to the *approvedSps* and *allApprovedSps* *length* amount of times instead of *1*.

Paths: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _updateSp()

./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : updateSp()

Recommendation: delete only the old wallet address and add new wallet address once when updating.

Status: Fixed (Revised commit: ed54c7c)

C04. Data Consistency

When adding the claim tokens in the *ClaimTokenFacet.addClaimToken* function, the *claimTokenofSUN* for each one from *pegTokens* is updated.

Therefore, if there was a value for the peg token in the *claimTokenofSUN*, it will be re-written.

When updating the claim tokens information in the *ClaimTokenFacet.updateClaimToken* function, the *claimTokenofSUN* is not updated accordingly to new perTokens.

Therefore, the claimTokenofSUN will be incorrect for the pegTokens.

Due to this, the claimTokenofSUN and claimTokens.pegTokens will be inconsistent.

Path: ./contracts/facets/claimtoken/ClaimTokenFacet.sol : addClaimToken(), updateClaimToken()

Recommendation: rework the logic to ensure that the same peg token can not be defined for different claim tokens, update the peg tokens information properly when updating claim tokens.

Status: Fixed (Revised commit: 3c0d52c)

C05. Denial of Service Vulnerability

When repaying the token loans or their liquidation, the before minted *gTokens* are burnt from the borrower.

Therefore, users may decrease the approval from the owner, the tokens will not be burnt. Due to this, the functions will revert and the lender will not be able to get any refunds.

Paths: ./contracts/facets/token/GToken.sol : approve()

./contracts/facets/market/token/TokenMarketFacet.sol : fullLoanPaybackTokenEarly(), _liquidateCollateralAutoSellOff(), _liquidateCollateralAutoSellOn()

Recommendation: rework the logic to ensure that the lender will get repayments.

Status: Fixed (Revised commit: ed54c7c)

C06. Funds Lock

When borrowers partially repay the loans and the repaying time expires, it is impossible to withdraw these partially repaid funds. The loan repayment may be blocked due to the changed minimal ltv value.

Therefore, the funds will be locked in the contract.

Paths: ./contracts/facets/liquidator/LiquidatorFacet.sol : paybackToken()

./contracts/facets/market/network/NetworkMarketFacet.sol : paybackEth()

Recommendation: ensure that partially repaid loans are not locked in the system.

Status: Fixed (Revised commit: ed54c7c)

C07. Requirements Violation; Denial of Service Vulnerability

When obtaining the claim token in the activateLoanToken through the `IClaimToken.getClaimTokenofSUNToken` for the autosell check is incorrect because of `i` index.

Therefore, this will lead to the incorrect claim token verification for the autosell liquidation and may lead to the denial of service due to out of bounds exceptions.

Path: ./contracts/facets/market/token/TokenMarketFacet.sol : activateLoanToken()

Recommendation: add the loop for the `loanDetails.stakedCollateralTokens` when the verification.

Status: Fixed (Revised commit: ed54c7c)

C08. Requirements Violation

The user manually sets the price for each NFT collateral token without any validation by passing the corresponding `stakedNFTPrice` field value in the `LoanDetailsNFT` structure. The `stakedNFTPrice` is used to verify if the user has sufficient tier for the loan creation with the specified `loanAmount`. Users may create a loan and specify a high enough NFT price value to pass tier validation and set a high loan amount.

This leads to the possibility to open a loan with an arbitrary `loanAmount` neglecting the user tier by manipulating the specified NFT price.

Path: ./contracts/facets/market/nft/NFTMarketFacet.sol : createLoanNft()

Recommendation: rework the logic and add calculations related to the potential NFT prices, probably using the marketplace endpoints.

Status: Mitigated (The documentation: "*NFT prices that will be used while creating the NFT loans will be fetch from Opensea API on our backend server, so when borrower creating the loan or lender is activating the loan, nft prices will be available to everyone on the marketplace, so lender before activating the loan, will see the current sale and floor price of nfts, so to avoid any the lending with arbitrary loan amount.*")

C09. Flashloan Attack

The project significantly relies on the token prices from the decentralized exchanges during the calculations. The user may manipulate the token pair price on the DEX using the flashloan attack in order to deposit less collateral. This affects calculations related to LTV value, required amount of collateral for a loan and liquidation process.

This may lead to the funds' loss.

Paths: ./contracts/facets/oracle/PriceConsumerFacet.sol : getTokenPriceFromDex()

./contracts/facets/oracle/LibPriceConsumer.sol : convertToStableOrWeth()

Recommendation: consider using alternative price sources, such as other decentralized exchanges or centralized exchanges.

Status: Fixed (Revised commit: 3c0d52c)

C10. Data Consistency

The calculations in the *convertToStableOrWeth()* function are done by dividing the reserves amounts of the pair, but the reserves proportion does not represent the price for a specific amount of tokens. UniswapV2 compatible AMM DEX calculates the output price using the next formula:

amountIn - amount of tokens to be sold;
reserveIn - reserves of Token which is sold;
reserveOut - reserves of Token which is bought.
*amountOut = amountIn * reserveOut / (reserveIn + amountIn)*

but not:

*amountOut = amountIn * reserveOut / reserveIn*

This formula does not take into account the actual amount of tokens being traded.

This leads to the overestimation of the tokens which may be bought on the DEX.

Path: ./contracts/facets/oracle/LibPriceConsumer.sol : convertToStableOrWeth()

Recommendation: use the proper AMM formula to consistently calculate the price for a specific amount of tokens.

Status: Fixed (Revised commit: ed54c7c)

C11. Frontrunning

It is possible to frontrun loan activation transactions to change the loan parameters and decrease the APY Fee before the activation. In case of the Network or Token collateralized loans, it may be updated with specifying the new *_newAPYOffer* value.

This may lead to an unexpected low APY fee which is possible to earn by the lender.

Paths: ./contracts/facets/market/network/NetworkMarketFacet.sol : updateEthLoan()

./contracts/facets/market/token/TokenMarketFacet.sol : updateTokenLoan()

Recommendation: time lock or add the loan signature, which should be validated during the activation and updated on every loan details change.

Status: Fixed (Revised commit: ed54c7c)

C12. Requirements Violation

The comparison operator is used instead of the assignment operator when assigning the *stableCoinAmounts[i]*/*_stableCoinAmount* to *loanDetails.loanAmountInBorrowed*.

Therefore, the correct stable coin amount will not be set.

Paths: ./contracts/facets/market/network/NetworkMarketFacet.sol : activateLoanEth()

./contracts/facets/market/token/TokenMarketFacet.sol : activateLoanToken()

Recommendation: replace the comparison operator with the assignment operator.

Status: Fixed (Revised commit: ed54c7c)

C13. Data Consistency

The parameters are passed in the incorrect order when calling the functions for the price calculation. The function *getTokenPriceFromDex* is called inside the *getCollateralPriceinStable* function and *_stableCoin* address is passed as a second argument, but

getTokenPriceFromDex function expects *_stableCoin* address to be the first argument in the function.

This may lead to the incorrect computations and significantly affect the price calculations.

Path: ./contracts/facets/oracle/PriceConsumerFacet.sol : getCollateralPriceinStable(), getStablePriceInCollateral()

Recommendation: correct the order for calculation of the arguments which are passed to the functions.

Status: Fixed (Revised commit: 3c0d52c)

■■■ High

H01. Denial of Service Vulnerability

All the approved tokens (*LibProtocolStorage.protocolRegistryStorage().allapprovedTokenContracts*) are processed in a loop when adding new tokens (*ProtocolRegistryFacet.isTokenApproved*).

All the approved Sps (*LibProtocolStorage.protocolRegistryStorage().allApprovedSps*) are processed in a loop when checking if the Sp was added to new Sps (*LibProtocolRegistry._isAlreadyAddedSp*), when getting sp index (*LibProtocolRegistry._getIndexofAddressfromArray*) and when removing it (*LibProtocolRegistry._removeSpKey*).

All the *tokenAddress`* Sps (*LibProtocolStorage.protocolRegistryStorage().approvedSps[tokenAddress]*) are processed in a loop when getting the sp index from mapping (*LibProtocolRegistry._getWalletIndexfromMapping*) and when removing it (*LibProtocolRegistry._removeSpKeyfromMapping*).

If the numbers of elements in the arrays are large enough to increase the Gas required for executing the loop over the block Gas limit, all the mentioned functionality may become inoperable.

Paths: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : isTokenApproved()

./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _isAlreadyAddedSp(), _getIndexofAddressfromArray(), _getWalletIndexfromMapping(), _removeSpKeyfromMapping(), _removeSpKey()

Recommendation: fix the logic not to rely on the arrays` lengths.

Status: Fixed (Revised commit: 3c0d52c)

H02. Denial of Service Vulnerability

All the approved admins (*LibAdminStorage.adminRegistryStorage().allApprovedAdmins*) are

processed in a loop when transferring super admin (*AdminRegistryFacet.transferSuperAdmin*), checking if the operation was approved by all the required admins (*AdminRegistryFacet.isDoneByAll*), removing the admin from the array (*LibAdmin._removeIndex*), finding the index (*LibAdmin._getIndex*), checking if admin exists (*LibAdmin._addressExists*).

All the admins that approved the operation (*LibAdminStorage.adminRegistryStorage().areByAdmins[_key][_newAdmin]*) are processed in a loop when checking if the operation was approved by all the required admins (*AdminRegistryFacet.isDoneByAll*), when checking if the admin is not approved by another admin (*LibAdmin._notAvailable*) or when rejecting the admin (*AdminRegistryFacet.rejectAdmin*).

All the pending keys (*LibAdminStorage.adminRegistryStorage().PENDING_KEYS*) are processed in a loop when checking if the admin is not approved by another admin (*LibAdmin._notAvailable*).

The *AdminRegistryFacet.isDoneByAll* and *LibAdmin._notAvailable* functions perform 2 loops, one inside another.

If the numbers of elements in the arrays are large enough to increase the Gas required for executing the loop over the block Gas limit, all the mentioned functionality may become inoperable.

Paths: ./contracts/facets/admin/AdminRegistryFacet.sol : transferSuperAdmin(), isDoneByAll(), rejectAdmin()

./contracts/facets/admin/LibAdmin.sol : _notAvailable(), _removeIndex(), _getIndex(), _addressExists()

Recommendation: fix the logic not to rely on the arrays` lengths.

Status: Fixed (Revised commit: ed54c7c)

H03. Denial of Service Vulnerability

All the tier levels (*LibGovTierStorage.govTierStorage().allTierLevelKeys*) are processed in a loop when getting the maximal Gov tier level (*GovTierFacet.maxGovTierLevelIndex*), removing tier (*LibGovTier._removeTierLevelKey*), checking if tier had been added (*LibGovTier.isAlreadyTierLevel*), getting the index of tier (*LibGovTier._getIndex*).

All the wallets (*LibGovTierStorage.govTierStorage().allTierLevelbyAddress*) are processed in a loop when checking if wallet has an added tier (*GovTierFacet.isAlreadyAddedWalletTier*).

If the numbers of elements in the arrays are large enough to increase the Gas required for executing the loop over the block Gas limit, all the mentioned functionality may become inoperable.

Paths: ./contracts/facets/govTier/GovTierFacet.sol : maxGovTierLevelIndex(), isAlreadyAddedWalletTier()

./contracts/facets/govTier/LibGovTier.sol : _removeTierLevelKey(), _getIndex(), isAlreadyTierLevel()

Recommendation: fix the logic not to rely on the arrays` lengths.

Status: Fixed (Revised commit: ed54c7c)

H04. Denial of Service Vulnerability

All the whitelisted addresses (*LibMarketRegistryStorage.marketRegistryStorage().allWhitelistAddresses*) are processed in a loop when checking if user has ever been whitelisted (*MarketRegistryFacet.isWhitelisedLender*).

If the number of elements in the array is large enough to increase the Gas required for executing the loop over the block Gas limit, the mentioned functionality may become inoperable.

Path: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol : isWhitelisedLender()

Recommendation: fix the logic not to rely on the array` lengths.

Status: Fixed (Revised commit: ed54c7c)

H05. Denial of Service Vulnerability

All the NFT tiers (*LibGovNFTTierStorage.govNftTierStorage().nftTierLevelsKeys*) are processed in a loop when obtaining the user`s NFT tier (*GovNFTTierFacet.getUserNftTier*), removing tier (*LibGovNFTTier._removeNftTierLevelKey*), getting NFT tier index (*LibGovNFTTier._getIndexSpTier*), checking if the NFT tier had been added (*LibGovNFTTier.isAlreadyNftTier*).

All the Sp tiers (*LibGovNFTTierStorage.govNftTierStorage().spTierLevelKeys*) are processed in a loop when removing tier (*LibGovNFTTier._removeSingleSpTierLevelKey*), getting Sp tier index (*LibGovNFTTier._getIndexNftTier*)

If the numbers of elements in the arrays are large enough to increase the Gas required for executing the loop over the block Gas limit, all the mentioned functionality may become inoperable.

Paths: ./contracts/facets/nftTier/LibGovNFTTier.sol : _removeSingleSpTierLevelKey(), _removeNftTierLevelKey(), _getIndexSpTier(), _getIndexNftTier(), isAlreadyNftTier()

./contracts/facets/nftTier/GovNFTTierFacet.sol : getUserNftTier()

Recommendation: fix the logic not to rely on the arrays` lengths.

Status: Fixed (Revised commit: ed54c7c)

H06. Denial of Service Vulnerability

All the Gov tiers (*IGovTier(address(this)).getGovTierLevelKeys()*) are processed in a loop when getting tier by Gov balance (*UserTierFacet.tierDatabyGovBalance*) or when obtaining user tier set by admin (*UserTierFacet.getTierDatabyWallet*).

All the collaterals (*_stakedCollateralTokens*) and tier allowed suns (*_nftTierData.allowedSuns*) are processed in a loop when validating the NFT sp tier (*LibUserTier.validateNFTSpTier*).

All the collaterals (*_stakedCollateralTokens*) and tier allowed tokens (*_vcTier.spAllowedTokens*) are processed in a loop when validating the vc tier (*LibUserTier.validateVCTier*).

All the NFT collaterals (*_stakedCollateralNFTs*) and tier allowed NFTs (*_nftTierData.allowedNfts*) are processed in a loop when validating the NFT sp tier for NFTs (*LibUserTier.validateNFTSpTierforNFTs*).

All the NFT collaterals (*_stakedCollateralNFTs*) and tier allowed tokens (*_vcTier.spAllowedNFTs*) are processed in a loop when validating the vc tier for NFTs (*LibUserTier.validateVCTierForNFTs*).

If the numbers of elements in the arrays are large enough to increase the Gas required for executing the loop over the block Gas limit, all the mentioned functionality may become inoperable.

Paths: ./contracts/facets/userTier/UserTierFacet.sol : tierDatabyGovBalance(), getTierDatabyWallet()

./contracts/facets/userTier/LibUserTier.sol : validateNFTSpTier(), validateVCTier(), validateNFTSpTierforNFTs(), validateVCTierForNFTs()

Recommendation: fix the logic not to rely on the arrays` lengths.

Status: Fixed (Revised commit: 3c0d52c)

H07. Denial of Service Vulnerability

All the whitelisted addresses (*LibVCTierStorage.vcTierStorage().vcTiersKeys*) are processed in a loop when getting user vc tier (*VCTierFacet.getUserVCNFTTier*) and when checking the tier has been added (*VCTierFacet.isAlreadyVcTier*).

If the number of elements in the array is large enough to increase the Gas required for executing the loop over the block Gas limit, the mentioned functionality may become inoperable.

Path: ./contracts/facets/vcTier/VCTierFacet.sol : getUserVCNFTTier(), isAlreadyVcTier()

Recommendation: fix the logic not to rely on the array`s lengths.

Status: Fixed (Revised commit: 3c0d52c)

H08. Denial of Service Vulnerability

All the chainlink feeds (*LibPriceConsumerStorage.priceConsumerStorage().allFeedContractsChainlink*) are processed in a loop when checking if the feed has been added (*LibPriceConsumer._isAddedChainlinkFeedAddress*).

All the claim token peg tokens are processed in a loop when getting sun token in stable (*PriceConsumerFacet.getSunTokenInStable*) and when getting stable token in sun (*PriceConsumerFacet.getStableInSunToken*).

If the number of elements in the arrays is large enough to increase the Gas required for executing the loop over the block Gas limit, all the mentioned functionality may become inoperable.

Paths: ./contracts/facets/oracle/LibPriceConsumer.sol : _isAddedChainlinkFeedAddress()

./contracts/facets/oracle/PriceConsumerFacet.sol : getStableInSunToken(), getStableInSunToken()

Recommendation: fix the logic not to rely on the array`s lengths.

Status: Fixed (Revised commit: ed54c7c)

H09. Requirements Violation; Denial of Service Vulnerability

The *ProtocolRegistryFacet.updateTokens* function accepts the tokens and their new *Market* data and emits the *TokensUpdated* event with this data. However, the *LibProtocolRegistry._updateToken* function updates the token only if its type should be set to *TokenType.ISVIP* and *gToken* is zero address.

Therefore, it is impossible to update the other tokens in a different way.

Paths: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : updateTokens()

./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _updateToken()

Recommendation: clarify the tokens update requirements and implement the code according to them.

Status: Mitigated (The documentation: “*Tokens with VIP type cannot be updated as there Gtokens will be already deployed, and token with DEX type will also be updated only once to VIP token type and there Gtokens will also be deployed.*”)

H10. Data Consistency

The super admin manually sets the users` tiers with no balance checks.

Therefore, users` tiers may not be valid.

Paths: ./contracts/facets/govTier/GovTierFacet.sol : addWalletTierLevel(), updateWalletTier(), removeTierLevel()

./contracts/facets/userTier/UserTierFacet.sol : getTierDatabyGovBalance()

Recommendation: dynamically check the user tier and not manually set it.

Status: Fixed (Revised commit: 3c0d52c) (Such function behavior is documented)

H11. Data Consistency; Requirements Violation

When adding tier through the *saveTierLevel* function, it is not validated for the *govHoldings* amount as it is performed in the addTierLevel function.

Therefore, the govHolding value may not be greater than last tier level *govHoldings*. Due to this missed validation, the requirements may be violated.

Path: ./contracts/facets/govTier/GovTierFacet.sol : saveTierLevel()

Recommendation: perform the missed govHoldings validations for the *saveTierLevel* functionality.

Status: Fixed (Revised commit: ed54c7c)

H12. Denial of Service Vulnerability; Requirements Violation

In some cases, it is impossible to add new admins due to the fact that 51% of approvals are required and the approval cannot be called for self. If there is only 1 admin with the *addGovAdmin* access: there is a functionality that allows to add an admin when there is 1 admin with the *addGovAdmin* access and no other admins (*if (es.allApprovedAdmins.length == 1)* case in the *addAdmin* function), but if other admins exist (which will always be true, as the last admin with the *editGovAdmin* role can not be deleted) it impossible. And the addition is impossible if all the admins with the *addGovAdmin* access are deleted.

It is impossible to edit or remove admins if there is only 1 admin and they have the *editGovAdmin* role, as it is impossible to call the operation for self.

Therefore, if there is only 1 admin with the *addGovAdmin* role and 1 with the *editGovAdmin* role, they can not add new admins and the *editGovAdmin* admin can not be edited or removed. If the admin with the *editGovAdmin* role removes the admin with the *addGovAdmin* role, it would be impossible to manage the admins in any way.

This will lead to the restriction of the admins functionality in the system to the functionality that is enabled to the last admin with the *editGovAdmin* role.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : addAdmin(), removeAdmin(), editAdmin()

Recommendation: add the functionality that allows an admin with the *addGovAdmin* access to add new admin and ensure that the last admin with the *addGovAdmin* access can not be deleted.

Status: Fixed (Revised commit: ed54c7c)

H13. Highly Permissive Role Access; Undocumented Behavior

The *rejectAdmin* function allows each admin with *editGov* role to reject any pending operation for admins.

Such behavior is not described in the documentation and may lead to manipulations.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : rejectAdmin()

Recommendation: clarify the requirements and ensure that the implementation matches it.

Status: Fixed (Revised commit: 3c0d52c)

H14. Data Consistency

The *getMaxTotalSupply* function does not indicate the maximal total supply, it is not limited and may increase.

Therefore, this functionality may lead to incorrect assumptions about the token information.

Path: ./contracts/facets/token/GToken.sol : getMaxTotalSupply()

Recommendation: clarify the function purpose, ensure that it does not lead to the wrong information provided.

Status: Mitigated (The Customer notice: "*Gtokens supply will always be equal to the sp token supply, so whenever the loan is create and token type is VIP then gtokens will be minted and supply of gtokens will be check with sp token supply, so it will not cause any issue in the system.*")

H15. Requirements Violation

The loans have the *isPrivate* parameter, but it does not change the loans functionality behavior. The comment in the code states that the "*private loans will not appear on the loan market*".

Therefore, the requirements are violated, and this may lead to the access control violations.

Path: ./contracts/facets/market/libraries/LibMarketStorage.sol : LoanDetailsToken.isPrivate, LoanDetailsNFT.isPrivate, LoanDetailsNetwork.isPrivate

Recommendation: clarify the private loan requirements and implement the code accordingly to them.

Status: Fixed (Revised commit: ed54c7c)

H16. Highly Permissive Role Access

The super admin may set the minimum ltv percentage (*LibMarketRegistryStorage.marketRegistryStorage().ltvPercentage*) through the *MarketRegistryFacet.setLTVPercentage* function.

The *NetworkMarketFacet* and *TokenMarketFacet* (*LiquidatorFacet*) do not allow to repay the loan if the ltv is less than the defined in the *MarketRegistryFacet* ltv limit. Therefore, the increasing of the limit may lead to the block of the loans repaying.

Such behavior is not described in the documentation.

Paths: ./contracts/facets/liquidator/LibLiquidator.sol : isLiquidationPending()

./contracts/facets/market/libraries/LibNetworkMarket.sol : isLiquidationPending()

./contracts/facets/marketRegistry/MarketRegistryFacet.sol : setLTVPercentage()

Recommendation: ensure that the implementation matches the requirements.

Status: Fixed (Revised commit: ed54c7c)

H17. Requirements Violation

The *NetworkMarketFacet* and *TokenMarketFacet* (*LiquidatorFacet*) do not allow to repay the loan and allow the loan liquidation if the ltv is less than the defined in the *MarketRegistryFacet* ltv limit, but this functionality is not implemented in the *NFTMarketFacet*.

This may indicate that the code is not finalized and the requirements are violated.

Paths: ./contracts/facets/liquidator/LibLiquidator.sol : isLiquidationPending()

./contracts/facets/market/libraries/LibNetworkMarket.sol : isLiquidationPending()

Recommendation: clarify the requirements and implement the code accordingly to them.

Status: Mitigated (The Customer notice: "*no need to perform LTV calculations in NFT Market.*")

H18. Requirements Violation; Undocumented Behavior

The apy fee that is not earned by the time (*unEarnedAPYFee*) goes to the platform.

Therefore, the lender does not get all the apy fee, though it is fully paid by the borrower.

Such behavior is not described in the documentation.

Paths: ./contracts/facets/liquidator/LibLiquidator.sol : _liquidateCollateralAutoSellOn(), _liquidateCollateralAutoSellOff(), fullLoanPaybackTokenEarly()

./contracts/facets/market/network/NetworkMarketFacet.sol : fullLoanPaybackEthEarly(), liquidateLoanNetwork(), liquidateLoanNetwork()

Recommendation: ensure that the lender gets the correct amount of apy fee, ensure that the implementation matches the requirements.

Status: Mitigated (The documentation: "*The "unearned APY%" is left in the diamond contract, which can be voted on by the community to determine the best use of these funds.*")

H19. Front-Running Attack

The liquidation function for the token loans swaps the funds using the UniswapV2. The min return amount after swap is not specified during operations with Uniswap router.

"Sandwich" attack is possible in such a case.

Path: ./contracts/facets/liquidator/LibLiquidator.sol : _liquidateCollateralAutoSellOn()

Recommendation: calculate and provide the min return amount as a parameter.

Status: Fixed (Revised commit: 3c0d52c) (Functionality removed)

H20. Insufficient Funds; Denial of Service Vulnerability

The total swapped amount is not validated for the amount in case it is less than the *loanAmountInBorrowed*.

This may lead to the Denial of Service vulnerabilities, when the *(loanDetails.loanAmountInBorrowed + earnedAPYFee) - autosellFeeinStable* can not be transferred due to the insufficient funds or to the situation when the funds that are intended for other refunds are used here.

Paths: ./contracts/facets/liquidator/LibLiquidator.sol : _liquidateCollateralAutoSellOn()

./contracts/facets/market/network/NetworkMarketFacet.sol : liquidateLoanNetwork()

Recommendation: ensure that the total swapped amount is enough for the refund.

Status: Fixed (Revised commit: ed54c7c)

H21. Data Consistency

The collateral tokens in the *TokenMarketFacet* and *NFTMarketFacet* are only transferred when activation.

Therefore, the loan offers may not be actual.

Paths: ./contracts/facets/market/token/TokenMarketFacet.sol : activateLoanToken()

./contracts/facets/market/nft/NFTMarketFacet.sol : activateNFTLoan()

Recommendation: transfer collaterals when creating the loan offer.

Status: Mitigated (The documentation: "*Transfer of collateral tokens from the borrower wallet to the GovWorld diamond contract will occur upon loan activation from a stablecoin lender funding a loan offer.*")

H22. Undocumented Behavior

When activating the loans, the lender defines the loan amount to be borrowed, and the actual loan offer amount may be re-written in case if the *maxLoanAmount* is less than the *loanAmountInBorrowed*.

Due to this, the borrower may get less tokens as was defined.

Such behavior is not described in the documentation.

Paths: ./contracts/facets/market/network/NetworkMarketFacet.sol : activateLoanEth()

./contracts/facets/market/nft/NFTMarketFacet.sol : activateNFTLoan()

./contracts/facets/market/token/TokenMarketFacet.sol : activateLoanToken()

Recommendation: ensure that the borrowers get the amount of tokens as per requirements.

Status: Mitigated (The documentation: "*Upon loan activation, if the collateral value of the borrower has decreased from the time the original loan offer was made, and the borrower is at the max LTV% determined by their GOV tier then, the borrower will getMaxLoanAmount that will be less than the originally requested loan amount by the borrower.*")

H23. Unfinalized Functionality

The *msgSender* function usage indicates that the meta-transactions were implied, but the functionality for them is not implemented.

This may indicate that the functionality is not implemented or that there is a redundant library usage in the project.

Path: ./contracts/shared/libraries/LibMeta.sol : msgSender()

Recommendation: clarify the requirements and finalize the code or remove the redundant code.

Status: Fixed (Revised commit: ed54c7c)

H24. Requirements Violation; Undocumented Behavior

After the super admin change - the previous super admin still has all the access rights, and the new admin is not granted all the permissions.

This may indicate that the requirements are violated.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : transferSuperAdmin()

Recommendation: clarify the requirements and ensure that the implementation matches it.

Status: Fixed (Revised commit: ed54c7c)

H25. Requirements Violation; Undocumented Behavior

When creating the loan, it is checked if the collaterals sent (*msg.value*) is greater than or equal (>=) to the price.

Users can mistakenly send more coins, will pay more than needed, and the coins will be locked.

Path: ./contracts/facets/market/network/NetworkMarketFacet.sol : createLoanEth()

Recommendation: use "equal to" operator for the sent collaterals amount checking.

Status: Fixed (Revised commit: ed54c7c)

H26. Inconsistent Data

The *changeTokensStatus* function allows to set the *isTokenEnabledAsCollateral* status when the token is not approved.

This may lead to the inconsistent contract state and the token access violations. The token will be added to the *approvedTokens*, but not to the *allapprovedTokenContracts*.

Path: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : changeTokensStatus()

Recommendation: verify if the token is approved before setting its *isTokenEnabledAsCollateral* status.

Status: Fixed (Revised commit: ed54c7c)

H27. Requirements Violation; Inconsistent Data

When validating the tiers in the *LibUserTier*, the amounts of collaterals are not validated for the *multiToken/multiNFT* tier type.

Therefore, the incorrect amount of collateral tokens may be used (1 token for the *multiToken/multiNFT*).

The functionality allows to set both *multiToken/multiNFT* and *singleToken/singleNft*, which contradicts itself and breaks the validations.

Paths: ./contracts/facets/userTier/LibUserTier.sol : validateGovHoldingTierForToken(), validateNFTTier(), validateNFTSpTier(), validateVCTier(), validateGovHoldingTierForNFT(), validateNFTSpTierforNFTs(), validateVCTierForNFTs()

./contracts/facets/nftTier/LibGovNFTTierStorage.sol : SingleSPTierData

./contracts/facets/govTier/LibGovTierStorage.sol : TierData

Recommendation: ensure that the one type of the tier is defined and validate it.

Status: Fixed (Revised commit: 5ff534b)

H28. Frontrunning

It is possible to frontrun loan activation transactions to change the loan parameters and increase the amount of tokens which are sent to the borrower during the activation. In case of the NFT collateralized loans, it may be updated by specifying the new *loanAmountInBorrowed* value and adjusting an inactive NFT loan with the tokens that have high *stakedNFTPrice* value specified by the borrower to pass the validations.

This may lead to an unexpected token transfer from the user's wallet to the borrower in case the allowance is higher than the initial *loanAmountInBorrowed*.

Path: ./contracts/facets/market/nft/NetworkMarketFacet.sol : updateNftLoan()

Recommendation: time lock or add the loan signature, which should be validated during the activation and updated on every loan details change.

Status: Fixed (Revised commit: ed54c7c)

H29. Data Consistency

It is possible to update the Chainlink Oracles addresses by the admin when the contract is not paused and feed is enabled. The function responsible for adding Oracle's feed address *addTokenChainlinkFeed* is

also capable of updating price feeds due to the lack of validation. The existing validation in the *_isAddedChainlinkFeedAddress* function only checks if a specific feed is added to any token or not.

This may lead to setting wrong addresses, which may cause unexpected behavior, DoS, or funds loss.

Path: ./contracts/facets/oracle/PriceConsumerFacet.sol : addTokenChainlinkFeed(), addBatchTokenChainlinkFeed()

Recommendation: keep the state of the system consistent. Disable the ability to update the price feeds in the function which is responsible for adding it, add function which is responsible for updating the price feed with a validation, which checks if the feed is enabled or contract is paused and disallows update otherwise.

Status: Fixed (Revised commit: 3c0d52c)

H30. Data Consistency

Uniswap V2 router is non-upgradable contract which is deployed to the mainnet. This means that the address may not be changed, but the project has the functionality which allows resetting the address of the Uniswap router.

This may lead to unexpected behavior, DoS or funds loss.

Path: ./contracts/facets/oracle/PriceConsumerFacet.sol : setSwapRouter()

Recommendation: hardcode the Uniswap address in the contract code.

Status: Fixed (Revised commit: 5ff534b)

H31. Denial of Service Vulnerability

All the liquidators are processed in the loop.

If the number of elements in the array is large enough to increase the Gas required for executing the loop over the block Gas limit, the mentioned functionality may become inoperable.

Paths: ./contracts/facets/liquidator/LibLiquidator.sol : isAlreadyAddedLiquidator()

./facets/liquidator/LiquidatorFacet.sol : getAllLiquidators() (when calling from another contract)

Recommendation: fix the logic not to rely on the array length.

Status: Fixed (Revised commit: 5ff534b)

H32. EIP Standard Violation

The GToken approve function acts like increaseAllowance function.

The EIP-20 approve function should overwrite the current allowance with the new value, therefore, the EIP standard is violated.

Path: ./contracts/facets/token/GToken.sol : approve()

Recommendation: implement the approve function according to the EIP-20 standard. Introduce the specific function that will increase the allowance for the loans creation purposes.

Status: Fixed (Revised commit: 3c0d52c)

H33. Requirements Violation

The verifications if the amounts do not exceed the ltv were removed.

The verification if the *maxLoanAmount != 0* was removed from the *TokenMarketFacet.activateLoanToken* function.

Therefore, the requirements are violated.

Paths: ./contracts/facets/market/token/TokenMarketFacet.sol : createLoanToken(), updateTokenLoan(), activateLoanToken();

./contracts/facets/market/network/NetworkMarketFacet.sol : updateEthLoan(), createLoanEth();

./contracts/facets/liquidator/LiquidatorFacet.sol : paybackToken();

Recommendation: clarify the ltv requirements and implement the code according to them.

Status: Fixed (Revised commit: 5ff534b)

H34. Ambiguous Third-Party Integration

The swap is performed using an external *call* with any *msg.data*, so it is possible to call any functions.

This may lead to calling the incorrect function, which will result in unexpected behavior.

Paths: ./contracts/facets/liquidator/LibLiquidator.sol : _liquidateCollateralAutoSellOn();

./contracts/facets/market/network/NetworkMarketFacet.sol : _liquidateAutoSellOn();

Recommendation: directly call the correct function using the known interface for *swap* function.

Status: Fixed (Revised commit: 5ff534b)

Medium

M01. Inconsistent Data

Critical state changes should emit events for tracking things off-chain.

The following functions do not emit events on change of important values.

Paths:

./contracts/facets/addressprovider/AddressProviderFacet.sol : setGovToken(), setgovGovToken()

./contracts/facets/marketRegistry/MarketRegistryFacet.sol : setMinLoanAmount(), setWhilelistAddress(), updateWhilelistAddress(), setAllowedMultiCollateralLimit(), set1InchAggregator()

./contracts/facets/nftTier/GovNFTTierFacet.sol : addSingleSpTierLevel(), addNftTierLevel(), addNFTTokensinNftTier(), addNFTSunTokensinNftTier(), updateSingleSpTierLevel(), removeSingleSpTierLevel(), removeNftTierLevel()

./contracts/facets/vcTier/VCTierFacet.sol : addVCNFTTier(), addVCSpTokens(), addVCNftTokens()

./contracts/facets/claimtoken/ClaimTokenFacet.sol : addClaimToken(), updateClaimToken(), enableClaimToken()

./contracts/facets/govTier/GovTierFacet.sol : govTierFacetInit()

./contracts/facets/liquidator/LiquidatorFacet.sol : liquidatorFacetInit()

./contracts/facets/oracle/PriceConsumerFacet.sol : priceConsumerFacetInit(), setSwapRouter()

./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : protocolRegistryFacetInit(), removeBulkSps()

Recommendation: emit the event whenever the corresponding action happens.

Status: Fixed (Revised commit: 3c0d52c)

M02. Inconsistent Data; Best Practice Violation

The *LibProtocolRegistry*, *PriceConsumerFacet*, contract can be initialized an unlimited number of times and the contracts are operable without the initialization. The *LiquidatorFacet*, GovTierFacet, AdminRegistryFacet contracts are operable without the initialization.

Such behavior violates the best practices and may lead to an inconsistent contract state.

Paths: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : protocolRegistryFacetInit()

./contracts/facets/oracle/PriceConsumerFacet.sol : priceConsumerFacetInit()

./contracts/facets/liquidator/LiquidatorFacet.sol : liquidatorFacetInit()

./contracts/facets/govTier/GovTierFacet.sol : govTierFacetInit()

./contracts/facets/admin/AdminRegistryFacet.sol : adminRegistryInit()

Recommendation: ensure that the contract can be initialized once and can not be operable without the initialization.

Status: Fixed (Revised commit: 3c0d52c)

M03. Best Practice Violation

It is not checked in the *MarketRegistryFacet.changeTokensStatus* function if the lengths of the *_tokenAddress* and *_tokenStatus* arrays are equal.

It is not checked in the *LibLiquidator.setLiquidator* function if the lengths of the *_newLiquidators* and *_newLiquidators* arrays are equal.

This may lead to incorrect input data processing.

Paths: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol : changeTokensStatus()

./contracts/facets/liquidator/LiquidatorFacet.sol : setLiquidator()

Recommendation: verify if the lengths of the *_tokenAddress*, *_tokenStatus* arrays in the *MarketRegistryFacet.changeTokensStatus* and *_newLiquidators*, *_liquidatorRole* in the *LibLiquidator.changeTokensStatus* function are equal.

Status: Fixed (Revised commit: ed54c7c)

M04. Contradiction

According to the comment, the *getSingleApproveTokenData* function should return *Market* struct, but it does not. ("*@dev get data of single approved token address return Market Struct*")

This may indicate that the requirements are violated.

Path: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol : getSingleApproveTokenData()

Recommendation: fix the mismatch.

Status: Fixed (Revised commit: ed54c7c)

M05. Inefficient Gas Model

The functions that remove elements from the array by the index, replace all the elements after the defined index in the loop.

This will lead to the redundant Gas consumption.

Paths: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _removeSpKey(), _removeSpKeyfromMapping()

./contracts/facets/nftTier/LibGovNFTTier.sol : _removeSingleSpTierLevelKey(), _removeNftTierLevelKey()

./contracts/facets/govTier/LibGovTier.sol : _removeTierLevelKey()

./contracts/facets/admin/LibAdmin.sol : _removeIndex(), _removePendingIndex()

Recommendation: replace the last array element to the index that should be removed and then remove the last element.

Status: Fixed (Revised commit: 3c0d52c)

M06. Inconsistent Data

The input tiers govHoldings are not checked for being less than the *govToken* total supply.

Therefore, the added tier may be unreachable and the requirements may be violated.

Path: ./contracts/facets/govTier/GovTierFacet.sol : govTierFacetInit()

Recommendation: ensure that the input tiers *govHoldings* are less than the govToken total supply.

Status: Fixed (Revised commit: ed54c7c)

M07. Inconsistent Data

The input tiers (*_bronze*, *_silver*, *_gold*, *_platinum*) are not checked for non-equality in the *GovTierFacet.govTierFacetInit* function.

As they are added through the *LibGovTier._addTierLevel* function without the *!LibGovTier.isAlreadyTierLevel* validation, this may lead to the repeated addition of tier to the *LibGovTierStorage.govTierStorage().allTierLevelKeys*.

Path: ./contracts/facets/govTier/GovTierFacet.sol : govTierFacetInit()

Recommendation: ensure that the input tiers levels are not equal.

Status: Fixed (Revised commit: ed54c7c)

M08. Inconsistent Data

The *LibMarketRegistryStorage.marketRegistryStorage().allWhitelistAddresses* array stores the addresses that are not whitelisted, as when updating a whitelisted address status to *false* through the *updateWhilelistAddress*, it is not removed from this array.

The *getAllWhitelisedLenders* function returns this array. Therefore, it does not indicate the real whitelisted users and may lead to the incorrect assumptions.

When adding the whitelisted users, the *setWhilelistAddress* function allows defining the status (*_value*).

Due to the fact that adding to the whitelist is implied to set the status to *true*, the parameter is redundant and allows to make the contract state inconsistent as the address will be added to the *LibMarketRegistryStorage.marketRegistryStorage().allWhitelistAddresses*.

Path: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol : setWhilelistAddress(), updateWhilelistAddress(), isWhitelisedLender(), getAllWhitelisedLenders()

Recommendation: automatically set status to true in the *setWhilelistAddress* function. Ensure that the *LibMarketRegistryStorage.marketRegistryStorage().allWhitelistAddresses* array stores the whitelisted users.

Status: Fixed (Revised commit: 3c0d52c)

M09. Inconsistent Data; Funds Lock

The *LiquidatorFacet.paybackToken* verifies if the *_paybackAmount <= ms.borrowerLoanNetwork[_loanId].loanAmountInBorrowed* without taking into account the already paid amount. The following verification *totalPayback >= loanDetails.loanAmountInBorrowed* allows the total payback amount to be greater than the borrowed amount.

Therefore, users may mistakenly pay more and the funds will be locked in the system.

Path: ./contracts/facets/liquidator/LiquidatorFacet.sol : paybackToken()

Recommendation: ensure that users can not pay more than required when repaying the loans.

Status: Fixed (Revised commit: ed54c7c)

M10. Inefficient Gas Model

The contracts with the ^0.8.3 Solidity version use the *SafeMath*.

Starting with Solidity ^0.8.0, *SafeMath* functions are built-in. In such a way, the library is redundant.

Path: ./contracts/uniswap/v2-periphery-master/*

Recommendation: remove the redundant functionality.

Status: Mitigated (The Customer notice: "*safeMath used in uniswap v2 is just for testing purposes, not in production/mainnet.*")

M11. Inconsistent Data

When assigning the tier to the wallet in the *addWalletTierLevel* function and updating it in the *updateWalletTier* function, it is not checked if the tier exists.

Therefore, this functionality may lead to the invalid contract state.

Path: ./contracts/facets/govTier/GovTierFacet.sol : updateWalletTier(), addWalletTierLevel()

Recommendation: validate if the tier exists when setting it to the wallet.

Status: Fixed (Revised commit: ed54c7c)

M12. Inconsistent Data

The *GovNFTTierFacet.addNFTSunTokensinNftTier*, *GovNFTTierFacet.addNFTTokensinNftTier*, *VCTierFacet.addVCSpTokens*, *VCTierFacet.addNFTSunTokensinNftTier* functions allow adding tokens to the tier when it is not added.

This may lead to the inconsistent contract state and the tiers' access and data violations.

Paths: ./contracts/facets/nftTier/GovNFTTierFacet.sol : addNFTTokensinNftTier(), addNFTSunTokensinNftTier()

./contracts/facets/vcTier/VCTierFacet.sol : addVCSpTokens(), addVCNftTokens()

Recommendation: verify if the tier is created before adding tokens to it.

Status: Fixed (Revised commit: ed54c7c)

M13. Contradiction

According to the function title, the *getSingleSpTierKeys* function should return the tier for a single Sp tier, but it returns all the Sp tiers, duplicating the *getAllSpTierKeys* function.

Path: ./contracts/facets/nftTier/GovNFTTierFacet.sol : getSingleSpTierKeys()

Recommendation: clarify the *getSingleSpTierKeys* function requirements and make the functionality fit it.

Status: Fixed (Revised commit: ed54c7c)

M14. Inefficient Gas Model

The *else if* case in the *tierDatabyGovBalance* function checks if the user has the highest tier in each iteration.

This will lead to redundant Gas consumption.

Path: ./contracts/facets/userTier/UserTierFacet.sol : tierDatabyGovBalance()

Recommendation: check if the user has the highest tier once before the loop.

Status: Fixed (Revised commit: ed54c7c)

M15. Inefficient Gas Model

All the user`s tiers are obtained in the *getMaxLoanAmountToValue*, *isCreateLoanTokenUnderTier*, *isCreateLoanNftUnderTier*, but the chosen be the caller type is used.

Therefore, this functionality will lead to redundant Gas consumption.

Path: ./contracts/facets/userTier/UserTierFacet.sol : getMaxLoanAmountToValue(), isCreateLoanTokenUnderTier(), isCreateLoanNftUnderTier()

Recommendation: obtain the defined tier type and process it.

Status: Fixed (Revised commit: ed54c7c)

M16. Inefficient Gas Model

The *NFTTierData.nftContract* parameter is redundant as it is the key in the *GovNFTTierStorage.nftTierLevels*.

The repeated information storage decreases the code readability and leads to the redundant Gas consumption.

Path: ./contracts/facets/nftTier/LibGovNFTTierStorage.sol : NFTTierData()

Recommendation: do not store the *NFTTierData.nftContract.*

Status: Fixed (Revised commit: 3c0d52c)

M17. Inconsistent Data

The input admins (*_superAdmin*, *_admin1*, *_admin2*, *_admin3*) are not checked for non-equality in the *AdminRegistryFacet.adminRegistryInit* function.

As they are added through the *LibAdmin._makeDefaultApproved* function without the *!LibAdmin._addressExists* validation, this may lead to the

repeated addition of admin to the *LibAdminStorage.adminRegistryStorage().allApprovedAdmins*.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : adminRegistryInit()

Recommendation: verify if the input admins are not the same in the *AdminRegistryFacet.adminRegistryInit* function.

Status: Fixed (Revised commit: ed54c7c)

M18. Contradiction

The functionality limits the adding, editing, or removing only one admin per time; therefore, only one pending admin is allowed. However, the storage variables are designed to store more than one address: *LibAdminStorage.AdminStorage().pendingAdminKeys* are the arrays for each pending key, and the title indicates the multiple addresses storage. The same is applicable to the *AdminRegistryFacet.getAllPendingAddedAdminKeys*, *AdminRegistryFacet.getAllPendingEditAdminKeys*, *AdminRegistryFacet.getAllPendingRemoveAdminKeys* functions` titles.

This may indicate that the requirements are violated and the functionality is not correctly implemented.

Paths: ./contracts/facets/admin/AdminRegistryFacet.sol : getAllPendingAddedAdminKeys(), getAllPendingEditAdminKeys(), getAllPendingRemoveAdminKeys()

./contracts/facets/admin/LibAdminStorage.sol : AdminStorage.pendingAdminKeys

Recommendation: clarify the requirements and ensure that the implementation does not contradict them.

Status: Fixed (Revised commit: 3c0d52c)

M19. Inefficient Gas Model

The *isPending* checks are redundant as the msg.sender can not be pending due to "call for self" checks, and since only one admin can be added, edited, removed at once.

Therefore, this functionality will lead to redundant Gas consumption.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : isPending(), approveAddedAdmin(), rejectAdmin(), approveRemovedAdmin(), approveEditAdmin()

Recommendation: remove the redundant functionality.

Status: Fixed (Revised commit: 3c0d52c)

M20. Inconsistent Data

There are events for each key when the rejection (*LibAdmin*: *AddAdminRejected*, *EditAdminRejected*, *RemoveAdminRejected()*), but *AddAdminRejected* event is emitted for any key in the *AdminRegistryFacet.rejectAdmin* function.

This will lead to incorrect assumptions about the admins rejection.

Paths: ./contracts/facets/admin/LibAdmin.sol : event AddAdminRejected(), event EditAdminRejected(), event RemoveAdminRejected()

./contracts/facets/admin/AdminRegistryFacet.sol : rejectAdmin()

Recommendation: emit the corresponding events when rejecting the admin to be added, edited, removed.

Status: Fixed (Revised commit: ed54c7c)

M21. Contradiction

The function *enableClaimToken* title indicates that the function should set token status to *true*, the function description tells that the function sets token status to *false* ("*function to make claim token false*"), but the function can both set *true* and *false* statuses.

This may indicate that the requirements are violated.

Path: ./contracts/facets/claimtoken/ClaimTokenFacet.sol : enableClaimToken()

Recommendation: fix the mismatch.

Status: Fixed (Revised commit: 3c0d52c)

M22. Inefficient Gas Model

The *calculateLTV* function call from the *PriceConsumerFacet* is redundant, as this value may be calculated using the already obtained *collatetralInBorrowed* value.

Therefore, this functionality will lead to the redundant Gas consumption.

Path: ./contracts/facets/market/libraries/LibTokenMarket.sol : getltvCalculations()

Recommendation: use the *collatetralInBorrowed* value for the ltv calculations.

Status: Fixed (Revised commit: ed54c7c)

M23. Inefficient Gas Model

The loan type setting is redundant as it can be defined based on the collateral tokens amount.

Therefore, this functionality will lead to the redundant Gas consumption.

Path: ./contracts/facets/market/token/TokenMarketFacet.sol : LoanDetailsToken.loanType, LoanDetailsNFT.loanType

Recommendation: remove the redundant functionality.

Status: Fixed (Revised commit: ed54c7c)

M24. Inefficient Gas Model

The *loanIds*, *stableCoinAmounts* and *_autoSell* lengths equality validation is performed in each loop iteration.

This will lead to the redundant Gas consumption.

Path: ./contracts/facets/market/token/TokenMarketFacet.sol : activateLoanToken()

Recommendation: verify the *loanIds*, *stableCoinAmounts* and *_autoSell* lengths equality once.

Status: Fixed (Revised commit: ed54c7c)

M25. Inconsistent Data

When emitting the *LibTokenMarket.TokenLoanOfferActivated* event in the *TokenMarketFacet.activateLoanToken* function, the *stableCoinAmounts[i]* is passed as a *_stableCoinAmount* parameter, but it only may be equal to the borrowed amount if *maxLoanAmount* < *loanDetails.loanAmountInBorrowed*. In other cases, the *loanDetails.loanAmountInBorrowed* is borrowed.

The *LibTokenMarket.TokenLoanOfferActivated._stableCoinAmount*, *LibNFTMarket.NFTLoanOfferActivated._loanAmount*, *LibNetworkMarket.LoanOfferActivated._stableCoinAmount* amounts do not indicate the real borrowed amount with fee cuts.

Therefore, the passed value may be incorrect and may lead to wrong assumptions on the activation result.

Paths: ./contracts/facets/market/token/TokenMarketFacet.sol : activateLoanToken()

./contracts/facets/market/libraries/LibTokenMarket.sol : TokenLoanOfferActivated(),

./contracts/facets/market/libraries/LibNFTMarket.sol : NFTLoanOfferActivated(),

./contracts/facets/market/libraries/LibNetworkMarket.sol : LoanOfferActivated()

Recommendation: use *loanDetails.loanAmountInBorrowed* instead of *stableCoinAmounts[i]* when emitting the *LibTokenMarket.TokenLoanOfferActivated* event in the *TokenMarketFacet.activateLoanToken* function. Ensure that the activation events provide the information about the actual borrowed amount with fee cuts.

Status: Fixed (Revised commit: ed54c7c)

M26. Inconsistent Data

When setting the liquidator role through the *LiquidatorFacet.setLiquidator* function, it is performed through the *LibLiquidator._makeDefaultApproved* function, and the *LibLiquidator._makeDefaultApproved* function pushes the liquidator address to the *LibLiquidatorStorage().whitelistedLiquidators*.

Therefore, when updating the liquidator role, the function will repeat the liquidator record.

Path: ./contracts/facets/liquidator/LibLiquidator.sol : _makeDefaultApproved()

Recommendation: ensure that the liquidator can not be added to the *LibLiquidatorStorage().whitelistedLiquidators* more than once.

Status: Fixed (Revised commit: 3c0d52c)

M27. Inconsistent Data

The *LibMarketRegistryStorage.LiquidatorStorage().whitelistedLiquidators* array stores the addresses that are not whitelisted, as when setting a whitelisted address status to *false* through the *setLiquidator*, it is added this array and not removed if it was present.

The *getAllLiquidators* function returns this array. Therefore, it does not indicate the real whitelisted liquidators and may lead to the incorrect assumptions.

Path: ./contracts/facets/liquidator/LiquidatorFacet.sol : getAllLiquidators(), setLiquidator()

Recommendation: ensure that the *LibMarketRegistryStorage.LiquidatorStorage().whitelistedLiquidators* array stores the whitelisted addresses.

Status: Fixed (Revised commit: ed54c7c)

M28. Denial of Service Vulnerability

The *(type(uint8).max - d + 1)* calculation is incorrect as it will overflow on d values that are bigger than the *type(uint8).max*.

This will make the functionality inoperable with some input data.

Path: ./contracts/uniswap/v2-periphery-master/dependencies/uniswap-lib/contracts/libraries/FullMath.sol : fullDiv()

Recommendation: do not change the external libraries and their versions, import them.

Status: Mitigated (The Customer notice: "*used in uniswap v2 just for testing purposes, not in production/mainnet.*")

M29. Inefficient Gas Model

APY Fee + Platform Fee transferring to the same contract is redundant.

This functionality will lead to the redundant Gas consumption and code readability decreasing.

Path: ./contracts/facets/market/token/TokenMarketFacet.sol : activateLoanToken()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: ed54c7c)

M30. Tests Failing

Some of the tests are failing. (*PriceConsumer*: *Should get the latest usd price from chainlink*, *Should get the latest usd prices in batch from chainlink*, *Should get the altcoin price from chainlink*).

All the tests should pass.

Path: ./test/5_priceConsumer.test.js

Recommendation: ensure that all tests are passing.

Status: Fixed (Revised commit: ed54c7c)

M31. Best Practice Violation; Inconsistent Data

The functions that find indexes return *0* for the found *0* index and for the not found index.

Therefore, the function result is not accurate and may lead to incorrect data processing.

Paths: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _getIndexofAddressfromArray(), _getWalletIndexfromMapping()

./contracts/facets/admin/LibAdmin.sol : _getIndex()

./contracts/facets/govTier/LibGovTier : _getIndex()

./contracts/facets/nftTier/LibGovNFTTier.sol : _getIndexSpTier(), _getIndexNftTier()

Recommendation: revert when the index is not found.

Status: Mitigated (The Customer notice: "*checking index already, no need for revert*")

M32. Inconsistent Data

It is considered to keep any data as accurately as possible until losses are quite small.

The functions have the code which performs division before multiplication during the calculation.

This may lead to a loss of precision.

Paths: ./contracts/facets/market/libraries/LibMarketProvider.sol : getautosellAPYFee(), getAPYFee()

./contracts/facets/liquidator/LibLiquidator.sol : getTotalPaybackAmount()

./contracts/facets/market/libraries/LibNetworkMarket.sol : getTotalPaybackAmount()

Recommendation: keep data actual to current system state, perform multiplication before division, multiply value by 10**decimals to keep some decimals.

Status: Fixed (Revised commit: 3c0d52c)

M33. Inefficient Gas Model

It is considered to avoid inefficient Gas models. The project has fields in the data structures that may be considered as redundant. *loanType* is a redundant field in the structure for NFT collateralized loans because it is possible to detect the collateral type using the length of *stakedCollateralNFTId* array, same for the loan type and *LoanTypeToken* enum.

This may lead to higher deployment Gas expenses and higher contract calls Gas expenses.

Path: ./contracts/facets/market/LibMarketStorage.sol : LoanDetailsToken.loanType, LoanDetailsNFT.loanType

Recommendation: rework the logic to remove the redundant fields and interactions with them.

Status: Fixed (Revised commit: 3c0d52c)

M34. Redundant Variable

It is considered to avoid inefficient Gas models. The project has redundant fields. The project stores the loan ids in the array similar to *loanOfferIdsNFTs*, which makes the *loanIdNft* value redundant as it is possible to extract the amount of loans using the array length value.

This may lead to higher deployment Gas expenses and higher contract calls Gas expenses.

Paths: ./contracts/facets/market/libraries/LibMarketStorage.sol : loanIdNetwork

./contracts/facets/market/libraries/LibMarketStorage.sol : loanIdNft

./contracts/facets/market/libraries/LibMarketStorage.sol : loanIdToken

Recommendation: rework the logic to remove the redundant fields and interactions with them.

Status: Fixed (Revised commit: ed54c7c)

M35. Best Practice Violation

It is considered following best practices to avoid unclear situations and prevent common attack vectors.

The Checks-Effects-Interactions pattern is violated. Some state variables are updated after the external calls.

This may lead to reentrancies, race conditions, and denial of service vulnerabilities during implementation of new functionality.

Paths: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _updateToken()

./contracts/facets/market/network/NetworkMarketFacet.sol : _liquidateAutoSellOn()

./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _addToken()

Recommendation: implement the code according to the Checks-Effects-Interactions pattern.

Status: Fixed (Revised commit: 5ff534b)

M36. Denial of Service Vulnerability

All the staked collaterals are processed in the loops when transferring them to the contract and to the lender in the *LibLiquidator*, *LibTokenMarket*, *LibNFTMarket* functions.

If the number of elements in the array is large enough to increase the Gas required for executing the loop over the block Gas limit, the mentioned functionality may become inoperable.

Paths: ./contracts/facets/liquidator/LibLiquidator.sol : _liquidateCollateralAutoSellOn(), _liquidateCollateralAutoSellOff(), fullLoanPaybackTokenEarly()

./contracts/facets/market/libraries/LibNFTMarket.sol : checkApprovedandTransferNFTs()

./contracts/facets/market/libraries/LibTokenMarket.sol : transferCollateralsandMintSynthetic()

./contracts/facets/oracle/PriceConsumerFacet.sol : calculateLTV()

Recommendation: fix the logic not to rely on the array`s lengths.

Status: Fixed (Revised commit: ed54c7c)

M37. Best Practice Violation

The input *_newOwner* parameter in the *transferOwnership* function is not checked for being non-zero address.

This may lead to the ownership transferring to the zero address.

Path: ./contracts/shared/facets/OwnershipFacet.sol : transferOwnership()

Recommendation: validate the *_newOwner* for being non-zero address.

Status: Fixed (Revised commit: ed54c7c)

M38. Inefficient Gas Model

Both *_saveTierLevel* and *_addTierLevel* functions perform the total supply verification.

This leads to the redundant Gas consumption.

Path: ./contracts/facets/govTier/LibGovTier.sol : _saveTierLevel(), _addTierLevel()

Recommendation: ensure that the verification is not duplicated.

Status: Fixed (Revised commit: 3c0d52c)

M39. Inconsistent Data

The *_newClaimtokendata.pegOrSunTokens.length* and *_newClaimtokendata.pegOrSunTokensPricePercentage.length* are not checked for equality.

This may lead to the inconsistent contract state.

Path: ./contracts/facets/claimtoken/ClaimTokenFacet.sol : updateClaimToken()

Recommendation: verify the arrays lengths for equality.

Status: Fixed (Revised commit: 3c0d52c)

M40. Inconsistent Data

Each sp may belong to different tokens.

This may lead to the inconsistent contract state.

Path: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : addSp()

Recommendation: clarify this point in the requirements and ensure that the implementation matches it.

Status: Mitigated (The Customer notice: "*each sp(strategic partner) wallet can also belong to another sp token, so like if there is one sp wallet that is also the part of some other sp token, can also be part of another sp token.*")

M41. Contradiction

The term of days parameters have different variable types.

This may indicate that the requirements are violated.

Paths: ./contracts/facets/market/libraries/LibMarketStorage.sol : LoanDetailsTokenData, LoanDetailsNFTData, LoanDetailsNetworkData, LoanDetailsToken, LoanDetailsNFT, LoanDetailsNetwork

./contracts/facets/market/network/NetworkMarketFacet.sol : updateEthLoan()

./contracts/facets/market/nft/NFTMarketFacet.sol : updateNftLoan()

./contracts/facets/market/token/TokenMarketFacet.sol : updateTokenLoan()

Recommendation: fix the types.

Status: Fixed (Revised commit: 3c0d52c)

Low

L01. Redundant Imports

The contracts contain redundant imports.

Redundant code decreased code readability.

Paths: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol : IProtocolRegistry, AppStorage

./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : LibAppStorage, AppStorage

./contracts/facets/vcTier/VCTierFacet.sol : LibGovTierStorage

./contracts/facets/liquidator/LiquidatorFacet.sol : LibAppStorage, LibTokenMarket, IUserTier, IClaimToken, LibMarketProvider

./contracts/facets/liquidator/LibLiquidator.sol : LibAppStorage

./contracts/facets/market/libraries/LibNetworkMarket.sol : IERC20Metadata, LibAppStorage

./contracts/facets/market/libraries/LibTokenMarket.sol : LibAppStorage, IMarketRegistry

./contracts/facets/nftTier/LibGovNFTTier.sol : IERC20, LibGovTierStorage

./contracts/facets/oracle/PriceConsumerFacet.sol : IDexFactory, IDexPair, IPriceConsumer

./contracts/facets/oracle/LibPriceConsumerStorage.sol : IClaimToken

./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : LibAppStorage

./contracts/facets/userTier/LibUserTier.sol : Modifiers, LibAppStorage

./contracts/shared/libraries/LibAppStorage.sol : LibProtocolStorage

Recommendation: remove the redundant imports.

Status: Fixed (Revised commit: 3c0d52c)

L02. Incorrect Functions Titles

The titles *setWhilelistAddress* and *updateWhilelistAddress* have the incorrect word "while".

Incorrect functions` titles decrease the code readability.

Path: ./contracts/facets/marketRegistry/MarketRegistryFacet.sol : setWhilelistAddress(), updateWhilelistAddress()

Recommendation: fix the functions titles.

Status: Fixed (Revised commit: 3c0d52c)

L03. Floating Pragma

The contracts use the 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. Using an outdated compiler version can be problematic, especially if publicly disclosed bugs and issues affect the current compiler version.

Paths: all the contracts

Recommendation: lock the pragma.

Status: Fixed (Revised commit: 3c0d52c)

L04. Contradiction

The *walletAddresses* parameter of the *BulkSpWalletAdded* event is used to represent one wallet address.

This may indicate that more than one address should be used in the *walletAddresses* parameter or that the parameter title is incorrect.

Path: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : BulkSpWalletAdded()

Recommendation: fix the mismatch.

Status: Fixed (Revised commit: ed54c7c)

L05. Unused Events

There are events in the contracts that are not used.

The redundant events decrease the code readability.

Path: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : AdminPercentageUpdated(), UpdatedUnearnedAPYPer()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: ed54c7c)

L06. Public Functions That Could Be Declared External

There are *public* functions in the contracts that are not called inside the system.

Functions with *external* visibility use less Gas.

Paths: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : setAutosellFee()

./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : setGovPlatfromFee(), setThresholdFee(), setAutosellFee()

./contracts/facets/market/token/TokenMarketFacet.sol : createLoanToken(), updateTokenLoan(), tokenLoanOfferCancel(), activateLoanToken()

./contracts/facets/liquidator/LiquidatorFacet.sol : paybackToken(), getLenderSUNTokenBalances()

./contracts/facets/market/network/NetworkMarketFacet.sol : createLoanEth(), updateEthLoan(), ethLoanOfferCancel(), activateLoanEth(), paybackEth()

./contracts/facets/market/nft/NFTMarketFacet.sol : createLoanNft(), nftloanOfferCancel(), updateNftLoan(), activateNFTLoan(), nftLoanPaybackBeforeTermEnd(), liquidateBorrowerNFT()

Recommendation: use *external* visibility for the functions that are never used inside the contracts.

Status: Fixed (Revised commit: 3c0d52c)

L07. Code Duplication

The *getallApprovedTokens* and *getTokenMarket* functions execute the same logic.

Duplicated code decreases the code readability.

Path: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : getallApprovedTokens(), getTokenMarket()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: ed54c7c)

L08. Redundant Property

The *addressProvider* property of the *GovTierStorage* struct is never used.

Redundant code decreases the code readability.

Path: ./contracts/facets/govTier/LibGovTierStorage.sol : GovTierStorage.addressProvider

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: ed54c7c)

L09. Redundant Property

The *reverseLoan* property of the *TierData* struct is never used.

Redundant code decreases the code readability.

Path: ./contracts/facets/govTier/LibGovTierStorage.sol : TierData.reverseLoan

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: ed54c7c)

L10. Commented Code

There is a commented code in the *UniswapV2Library* contract.

This decreases the code readability.

Path: ./contracts/uniswap/v2-periphery-master/libraries/UniswapV2Library.sol

Recommendation: remove the commented code.

Status: Mitigated (The Customer notice: "*used for testing purposes, not in production/mainnet.*")

L11. Incorrect Function Title

The *getAllTierlevelbyAddress* function returns all tier levels by all the addresses, but the title indicates that the function should return a tier for the specific address.

This may lead to the incorrect assumption of the function purpose and decrease the code readability.

Path: ./contracts/facets/govTier/GovTierFacet.sol : getAllTierlevelbyAddress()

Recommendation: fix the function title to match its functionality.

Status: Fixed (Revised commit: ed54c7c)

L12. Redundant Variable Reference

When returning the function result, the *es.allTierLevelbyAddress* is obtained, but the *_tierLevels* can be used.

This leads to the redundant Gas consumption.

Path: ./contracts/facets/govTier/GovTierFacet.sol : getAllTierlevelbyAddress()

Recommendation: remove the redundant variable reference.

Status: Fixed (Revised commit: ed54c7c)

L13. Unclear Error Message

The "*allowed sp token*" error messages do not indicate revert reason.

Therefore, the failing reason is not clear.

Path: ./contracts/facets/vcTier/VCTierFacet.sol : addVCSpTokens(), addVCNftTokens()

Recommendation: make the revert reason comprehensive.

Status: Fixed (Revised commit: 3c0d52c)

L14. Unclear Error Message

The "*GTL: set govHolding error*" error messages do not indicate revert reason.

Therefore, the failing reason is not clear.

Paths: ./contracts/facets/govTier/LibGovTier.sol : _saveTierLevel(), _addTierLevel()

./contracts/facets/govTier/GovTierFacet.sol : addTierLevel(), updateTierLevel()

Recommendation: make the revert reason comprehensive.

Status: Fixed (Revised commit: 3c0d52c)

L15. Redundant Check

The verification if the new admin is not a *msg.sender* in the *approveAddedAdmin* function is redundant as the already presented admin can not be added in the *addAdmin* function.

Therefore, this will lead to redundant Gas consumption.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : approveAddedAdmin()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: 3c0d52c)

L16. Redundant Check

The verification if the *es.allApprovedAdmins[i] != _newAdmin* is redundant for the *PENDING_ADD_ADMIN_KEY* as the admin to be added can not be present in the *allApprovedAdmins* array.

Therefore, this will lead to redundant Gas consumption.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : isDoneByAll()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: 3c0d52c)

L17. Contradiction

The function *isPending* returns *true* if *_sender* is not pending and *false* in case it is.

Therefore, the function title indicates the opposite of actual function behavior and may lead to the incorrect assumption of the function purpose.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : isPending()

Recommendation: fix the title to match the functionality.

Status: Fixed (Revised commit: 3c0d52c)

L18. Contradiction

The function _notAvailable returns *true* if *_sender* is available and *false* in case it is not.

Therefore, the function title indicates the opposite of actual function behavior and may lead to the incorrect assumption of the function purpose.

Path: ./contracts/facets/admin/LibAdmin.sol : _notAvailable()

Recommendation: fix the title to match the functionality.

Status: Fixed (Revised commit: ed54c7c)

L19. Redundant Parameter

The *tokenType* field is never used.

The redundant code decreases the code readability.

Path: ./contracts/facets/claimtoken/LibClaimTokenStorage.sol : ClaimTokenData.tokenType

Recommendation: remove the redundant code.

Status: Mitigated (The Customer notice: "*tokenType is just a representation for peg or sun token, to show token type for external use.*")

L20. Contradiction

The *pegTokens* and *pegTokensPricePercentage* may contain information for both peg and sun tokens, but the titles do not contain the "sun".

Therefore, the titles do not indicate the variables' purposes, which decreases the code readability.

Path: ./contracts/facets/claimtoken/LibClaimTokenStorage.sol : ClaimTokenData.pegTokens, ClaimTokenData.pegTokensPricePercentage

Recommendation: fix the variables' titles.

Status: Fixed (Revised commit: ed54c7c)

L21. Redundant Functionality

The *aggregator1Inch* value is never used, but the functionality for its setting and storage is implemented.

The redundant functionality decreases the code readability and leads to the unnecessary Gas consumption.

Paths: ./contracts/facets/liquidator/LiquidatorFacet.sol : approveCollateralTo1inch()

./contracts/facets/liquidator/LibLiquidatorStorage.sol : LiquidatorStorage.aggregator1Inch

Recommendation: remove the redundant functionality.

Status: Fixed (Revised commit: 3c0d52c)

L22. Redundant Parameter

The *_stableCoinAmount* parameter is redundant as this value may be obtained from *loanDetailsNFT.loanAmountInBorrowed*.

Therefore, the usage of this parameter will lead to the unnecessary Gas consumption.

Path: ./contracts/facets/market/nft/NFTMarketFacet.sol : activateNFTLoan()

Recommendation: remove the redundant functionality.

Status: Fixed (Revised commit: ed54c7c)

L23. Contradiction

The comment states that the "*contract will the repay staked collateral tokens to the borrower*", but the tokens are transferred to the lender.

Incorrect comments decrease the code readability.

Path: ./contracts/facets/market/nft/NFTMarketFacet.sol : liquidateBorrowerNFT()

Recommendation: fix the mismatch.

Status: Fixed (Revised commit: ed54c7c)

L24. Unclear Error Message

The "*loan status inactive*" error messages do not indicate revert reason. They indicate that the current loan state is inactive, but it should indicate that the loan status has to be active, and it is not.

Therefore, the failing reason is not clear.

Paths: ./contracts/facets/market/token/TokenMarketFacet.sol : updateTokenLoan()

./contracts/facets/market/nft/NFTMarketFacet.sol : updateNftLoan()

./contracts/facets/market/network/NetworkMarketFacet.sol : updateEthLoan()

Recommendation: fix the mismatch.

Status: Fixed (Revised commit: 3c0d52c)

L25. Redundant Variable

The *s* variable is never used.

Redundant variables decrease the code readability.

Path: ./contracts/shared/libraries/LibAppStorage.sol : Modifiers.s

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: ed54c7c)

L26. Code Duplication

The *getautosellAPYFee* and *getAPYFee* functions execute the same logic.

Duplicated code decreases the code readability.

Path: ./contracts/facets/market/libraries/LibMarketProvider.sol : getautosellAPYFee(), getAPYFee()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: ed54c7c)

L27. Misleading Variable Title

The project has a variable whose name contradicts the purpose. The *loanActivateLimit* stores the number which represents the total number of ever-activated loans, but this number does not decrease on loan deactivation.

Path: ./contracts/facets/market/libraries/LibMarketStorage.sol : loanActivateLimit

Recommendation: rename the variable according to the purpose (e.g. *loanActivatedLimit*) or add the logic to decrease the counter after loan is closed.

Status: Fixed (Revised commit: ed54c7c)

L28. Best Practice Violation

The Sp and NFT tokens are not verified for being unique.

This may lead to the inconsistent contract state.

Path: ./contracts/facets/vcTier/VCTierFacet.sol : addVCNFTTier(), addVCSpTokens(), addVCNftTokens()

Recommendation: verify that tokens are unique.

Status: Fixed (Revised commit: 3c0d52c)

L29. Code Duplication

The *checkApprovedAdmins* can be used in the *_key == es.PENDING_EDIT_ADMIN_KEY* condition.

The duplicated code decreases the code readability.

Path: ./contracts/facets/admin/AdminRegistryFacet.sol : isDoneByAll()

Recommendation: use the *checkApprovedAdmins* in the *_key == es.PENDING_EDIT_ADMIN_KEY* condition.

Status: Fixed (Revised commit: 3c0d52c)

L30. Code Duplication

The *getSingleApproveToken()* and *getSingleApproveTokenData()* functions execute the same logic.

The duplicated code decreases the code readability.

Path: ./contracts/facets/protocolRegistry/ProtocolRegistryFacet.sol : getSingleApproveToken(), getSingleApproveTokenData()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: 3c0d52c)

L31. Inefficient Gas Model

The *key* variable may be used in the *Max SP Tier Keys Exceeded* require statement.

The redundant variable reference increases the Gas consumption.

Path: ./contracts/facets/nftTier/GovNFTTierFacet.sol : addSingleSpTierLevel()

Recommendation: use the *key* variable in the require statement.

Status: Fixed (Revised commit: 3c0d52c)

L32. Redundant Import

The contract contains the redundant import.

This decreases the code readability.

Path: ./contracts/facets/market/libraries/LibNFTMarket.sol : IProtocolRegistry()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: 3c0d52c)

L33. Data Consistency

The verification if the sp has been already added was removed.

This may lead to the duplications and inconsistent contract state.

Path: ./contracts/facets/protocolRegistry/LibProtocolRegistry.sol : _addSp

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: 3c0d52c)

L34. Data Consistency

The verification if the tier has been already added was removed.

This may lead to the duplications and inconsistent contract state.

Path: ./contracts/facets/govTier/GovTierFacet.sol : addTierLevel()

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: 3c0d52c)

L35. Redundant Imports

There are redundant imports in the contract.

Redundant code decreases the code readability.

Path: ./contracts/facets/oracle/LibPriceConsumer.sol : IDexPair, IERC20Extras

Recommendation: remove the redundant code.

Status: Fixed (Revised commit: 5ff534b)

L36. Code Duplication

The getPool function and *DEFAULT_FEE*, *FEE_1*, *FEE_2* variables are present in both *UniswapOracleV3* and *LibPriceConsumer*.

Duplicated code decreases the code readability.

Paths: ./contracts/facets/oracle/UniswapOracleV3.sol : getPool(), DEFAULT_FEE, FEE_1, FEE_2;

./contracts/facets/oracle/LibPriceConsumer.sol: getPool(), DEFAULT_FEE, FEE_1, FEE_2;

Recommendation: remove the duplicated code.

Status: Fixed (Revised commit: 5ff534b)

L37. Default Visibility Usage

The visibility is not set for the *DEFAULT_FEE*, *FEE_1*, *FEE_2* variables.

This is a best practice violation and can lead to incorrect access to the variable.

Path: ./contracts/facets/oracle/LibPriceConsumer.sol: DEFAULT_FEE, FEE_1, FEE_2;

Recommendation: set the visibility to the variables.

Status: Fixed (Revised commit: 5ff534b)

L38. Incorrect Comment

The `_claimTokenAddress the key to remove` comment is incorrect as the `_claimTokenAddress` is not always removed.

This may lead to incorrect assumptions on the code's purposes.

Path: ./contracts/facets/claimtoken/ClaimTokenFacet.sol : enableClaimToken()

Recommendation: fix the comment.

Status: Fixed (Revised commit: 5ff534b)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing 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 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, the Consultant cannot guarantee the explicit security of the audited smart contracts.