SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

MuxWorld

Audit

Security Assessment
15. December, 2022

For

MUX

SolidProof_io

@solidproof_io

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	18
Source Units in Scope	19
Critical issues	21
High issues	21
Medium issues	21
Low issues	21
Informational issues	21
Audit Comments	24
SWC Attacks	25

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	15. December 2022	• Layout project • Automated- /Manual-Security Testing • Summary
	15. - 19. December 2022	• Checking the code
	20. December 2022	• Finishing the report

Network
Arbitrum

Website
https://mux.network/

Telegram
https://t.me/muxprotocol

Twitter
https://twitter.com/muxprotocol

Github
https://github.com/mux-world/mux-protocol

Discord
https://discord.gg/bd88NrzN3N

Description

The MUX Aggregator is a sub-protocol in the MUX protocol suite that automatically selects the most suitable liquidity route and minimizes the composite cost for traders while meeting the needs of opening positions. The aggregator can also supply additional margin for traders to raise the leverage up to 100x on aggregated underlying protocols.

Project Engagement

During the 14th of December 2022, **MuxWorld Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- Github
    - https://github.com/mux-world/mux-aggregator-protocol/tree/main/contracts/aggregators/gmx
    - Commit: 95b98ed3ba40ef4b7dc58045310367d037f48d44

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
   ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

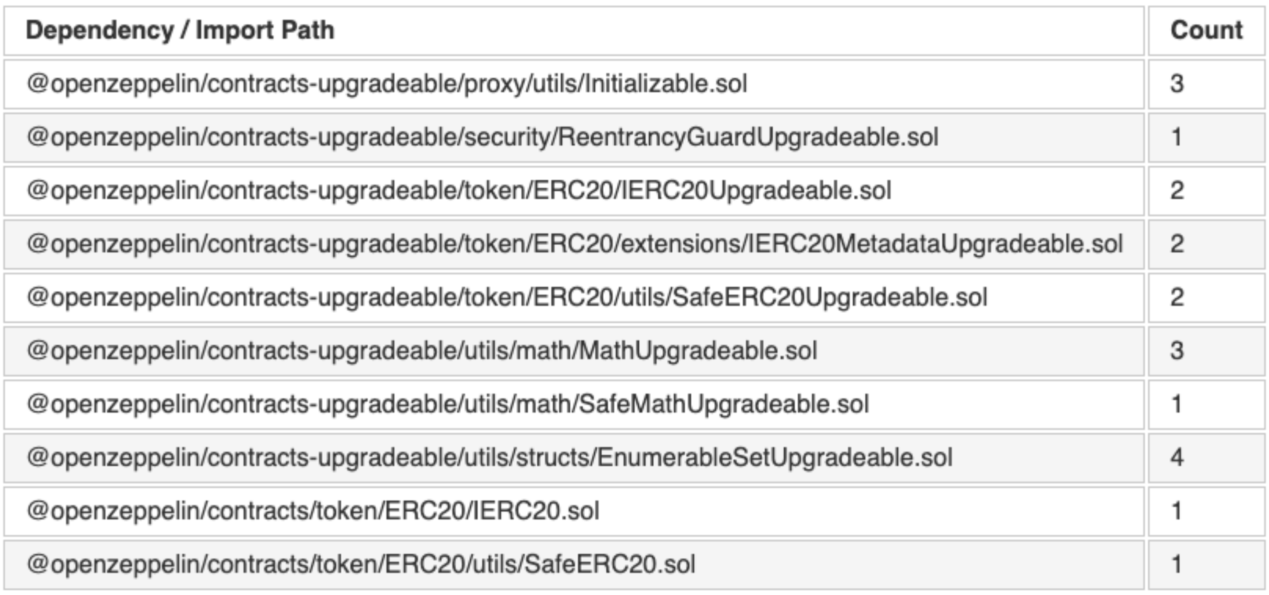
2. Testing and automated analysis that includes the following:
   i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:



Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	3
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	2
@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol	4
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	1

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

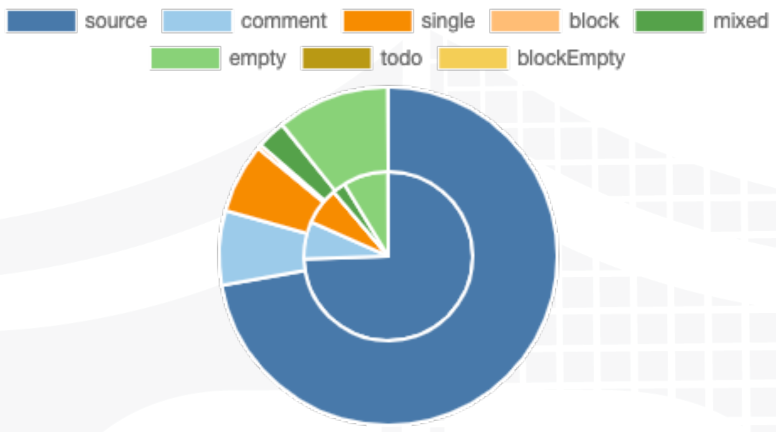
v1.0

File Name	SHA-1 Hash
contracts/interfaces/IGmxFastPriceFeed.sol	fb9b60f8f19b443e79e2ef4aa7072fa01f4f9d19
contracts/interfaces/IGmxBasePositionManager.sol	ee2e12732f575ea983aa456893de5073b67e304c
contracts/interfaces/IGmxPositionRouter.sol	c12cc52e4442e177178a274193fc6d465b8a4002
contracts/interfaces/IProxyFactory.sol	784375c736d5538f37507dde9ba2fb9c7e80abf2
contracts/interfaces/ILiquidityPool.sol	142da6c6eae6b9958719c61f857374f084c13075
contracts/interfaces/IChainLink.sol	f3591a9fbdd5abd06515fc95a7b90c91c4fb6a45
contracts/interfaces/IGmxRouter.sol	b9fff35c51c7e816ee89ff53549adc8900088703
contracts/interfaces/IGmxOrderBook.sol	6d6d1b54883503a0dd3268e4ec5564ac73394be6
contracts/interfaces/IWETH.sol	ed94123a87ea5609d9d1d42a5b58521f1de3a93f
contracts/interfaces/IGmxVault.sol	1fb0fc9b0654b9bd5c4f66afefee00199b1e6609
contracts/interfaces/IGmxReader.sol	90e732e6e44a7b3485f36e477007a7314ed7c1c1
contracts/interfaces/IGmxPositionManager.sol	17e2482dc4b35d56f2a213d50adb9763d0fdd562
contracts/interfaces/IReferralManager.sol	75981441b8ef882bb992aaf534ed6e19ef7be875
contracts/interfaces/IAggregator.sol	57b43c59602a1d7dc1e76dac1b373fcf1b1298ca
contracts/components/ImplementationGuard.sol	d2af93b9f117f95ac0ec32b6e602c5a979205931
contracts/aggregators/gmx/Storage.sol	a8ab18781389160b712d2d1f330d4b975a5fccbe
contracts/aggregators/gmx/libs/LibOracle.sol	7fa8f5b322f9fe9a3f96d886bfb9a8c0e7b0ff42
contracts/aggregators/gmx/libs/LibUtils.sol	885a47a5322ac6fe24c7565f8e1390e08528a6dc
contracts/aggregators/gmx/libs/LibGmx.sol	f38dfdb36696bde33bf26ed6e8990f5c2b87be3d
contracts/aggregators/gmx/Config.sol	c1c1d83280ae9d5771ede1004d28e6d243728817
contracts/aggregators/gmx/Debt.sol	4b095ca3cd82b3ee423b21acb1e092430f643ba9
contracts/aggregators/gmx/GmxAdapter.sol	371136385545e49609dd02792ee1db5fa781e928
contracts/aggregators/gmx/Position.sol	ea574bebe884a296e7bbde082b406a00df7811e3
contracts/aggregators/gmx/Type.sol	5cc9920b6ca49d26272fdf94e0caf17d91c84cd0

Metrics

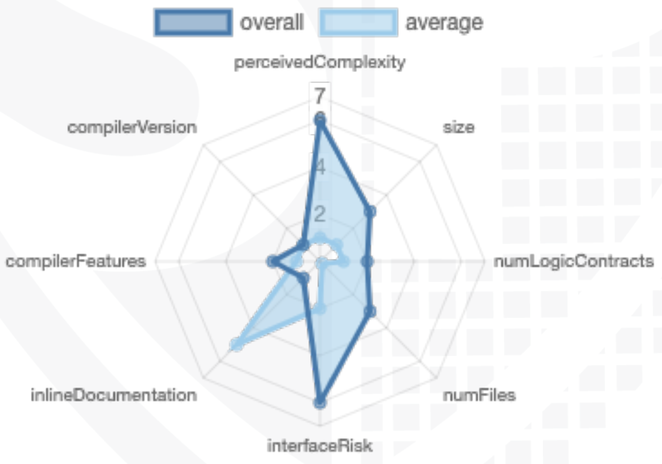
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	6	3	16	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	142	12

Version	External	Internal	Private	Pure	View
1.0	140	126	0	13	95

State Variables

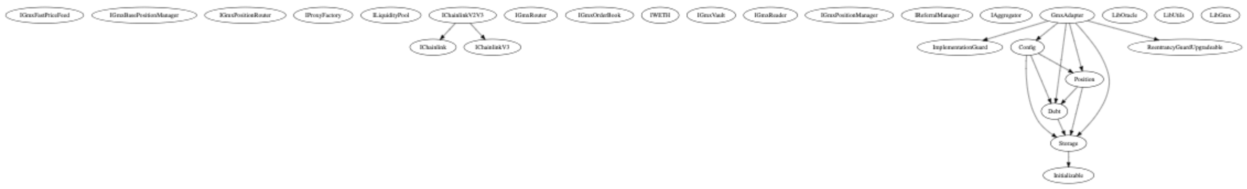
Version	Total	Public
1.0	18	0

Capabilities

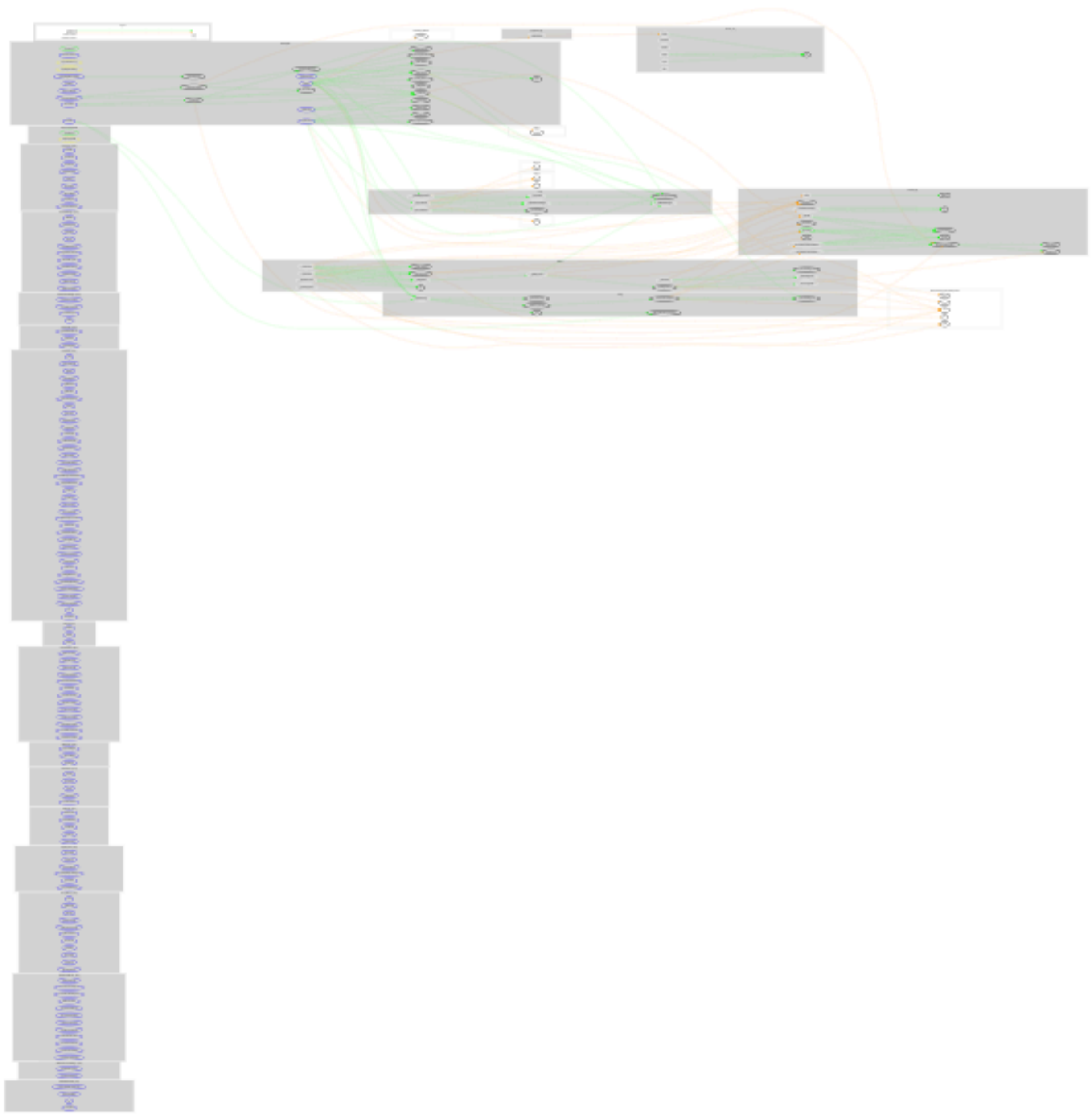
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.17`		`yes`		

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0				yes		

Inheritance Graph
v1.0



CallGraph
v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Is contract an upgradeable
2. Overall checkup (Smart Contract Security)

Is contract an upgradeable

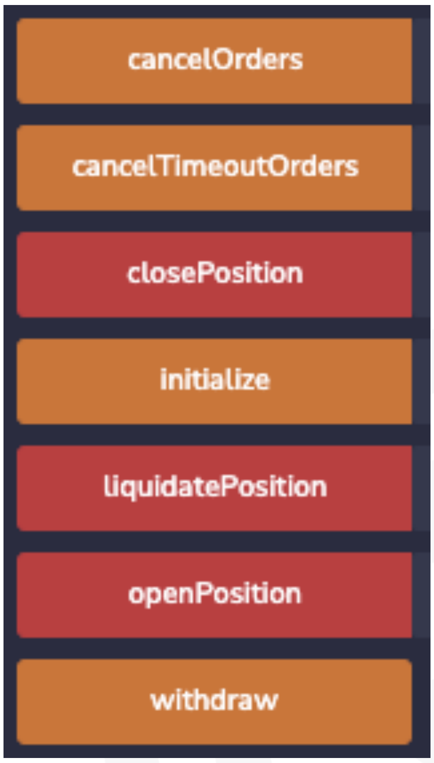
Name	
Is contract an upgradeable?	Yes

Comments:

v1.0

- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
    - Be aware of this and do your own research for the contract which is the contract pointing to

Write functions of contract v1.0



Overall checkup (Smart Contract Security)

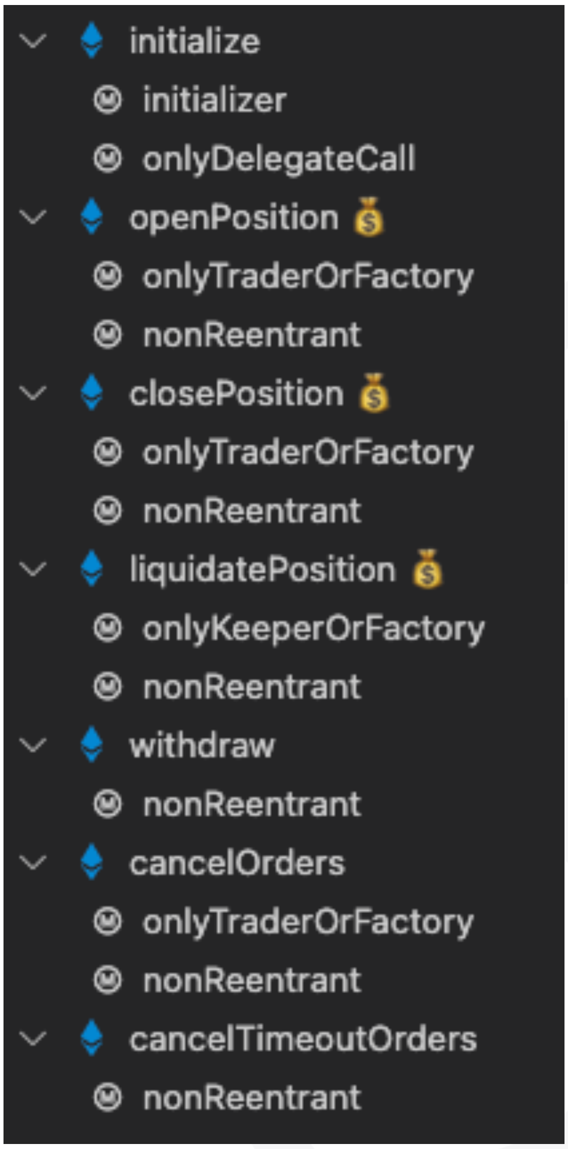
Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers and public functions

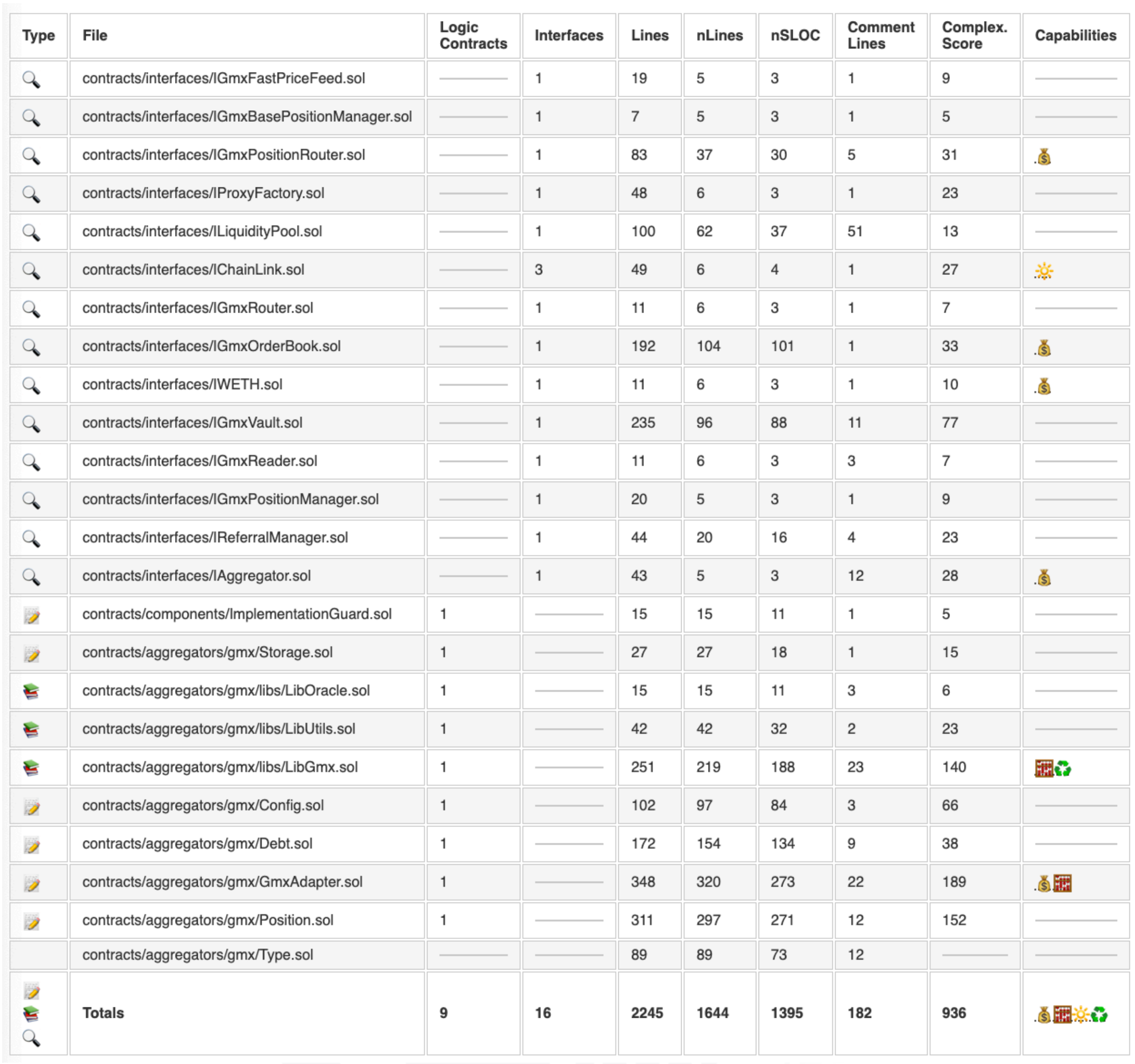
v1.0



- initialize
  - initializer
  - onlyDelegateCall
- openPosition 💰
  - onlyTraderOrFactory
  - nonReentrant
- closePosition 💰
  - onlyTraderOrFactory
  - nonReentrant
- liquidatePosition 💰
  - onlyKeeperOrFactory
  - nonReentrant
- withdraw
  - nonReentrant
- cancelOrders
  - onlyTraderOrFactory
  - nonReentrant
- cancelTimeoutOrders
  - nonReentrant

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope
v1.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/interfaces/IGmxFastPriceFeed.sol	—	1	19	5	3	1	9	—
🔍	contracts/interfaces/IGmxBasePositionManager.sol	—	1	7	5	3	1	5	—
🔍	contracts/interfaces/IGmxPositionRouter.sol	—	1	83	37	30	5	31	💰
🔍	contracts/interfaces/IProxyFactory.sol	—	1	48	6	3	1	23	—
🔍	contracts/interfaces/ILiquidityPool.sol	—	1	100	62	37	51	13	—
🔍	contracts/interfaces/IChainLink.sol	—	3	49	6	4	1	27	🔆
🔍	contracts/interfaces/IGmxRouter.sol	—	1	11	6	3	1	7	—
🔍	contracts/interfaces/IGmxOrderBook.sol	—	1	192	104	101	1	33	💰
🔍	contracts/interfaces/IWETH.sol	—	1	11	6	3	1	10	💰
🔍	contracts/interfaces/IGmxVault.sol	—	1	235	96	88	11	77	—
🔍	contracts/interfaces/IGmxReader.sol	—	1	11	6	3	3	7	—
🔍	contracts/interfaces/IGmxPositionManager.sol	—	1	20	5	3	1	9	—
🔍	contracts/interfaces/IReferralManager.sol	—	1	44	20	16	4	23	—
🔍	contracts/interfaces/IAggregator.sol	—	1	43	5	3	12	28	💰
📝	contracts/components/ImplementationGuard.sol	1	—	15	15	11	1	5	—
📝	contracts/aggregators/gmx/Storage.sol	1	—	27	27	18	1	15	—
📚	contracts/aggregators/gmx/libs/LibOracle.sol	1	—	15	15	11	3	6	—
📚	contracts/aggregators/gmx/libs/LibUtils.sol	1	—	42	42	32	2	23	—
📚	contracts/aggregators/gmx/libs/LibGmx.sol	1	—	251	219	188	23	140	🧮♻️
📝	contracts/aggregators/gmx/Config.sol	1	—	102	97	84	3	66	—
📝	contracts/aggregators/gmx/Debt.sol	1	—	172	154	134	9	38	—
📝	contracts/aggregators/gmx/GmxAdapter.sol	1	—	348	320	273	22	189	💰🧮
📝	contracts/aggregators/gmx/Position.sol	1	—	311	297	271	12	152	—
	contracts/aggregators/gmx/Type.sol	—	—	89	89	73	12	—	—
📝📚🔍	**Totals**	**9**	**16**	**2245**	**1644**	**1395**	**182**	**936**	💰🧮🔆♻️

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments

Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, …)

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Storage	State variable visibility is not set	14	It is best practice to set the visibility of state variables explicitly

Informational issues

Issue	File	Type	Line	Description
#1	Config	Unused return values	75, 79,	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#2	Position	Unused return values	144, 162, 204, 221, 250, 263	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	Config	Misspelling	See description	Change following words: - prevousOrderBook L65 - newPostitionRouter L66 Make sure to change it everywhere else as well.

#4	Type	Misspelling	See description	Change following words: - REFERRENCE_ORACLE L21 - REFERRENCE_ORACLE_DEVIATION L22 - referrenceOracle L40 Make sure to change it everywhere else as well.
#5	GmxAdapter	Misspelling	See description	Change following words: - Openning L106 - referrenceOracle L274, L277 - ZeroOralcePrice L278 Make sure to change it everywhere else as well.
#6	IGmxVault	Misspelling	See description	Change following words: - realisedPnlUsd L11 - realisedPnl L58, L68, L77, L145 - hasRealisedPnl L164 Make sure to change it everywhere else as well.
#7	All	NatSpec documentation missing	-	We recommend you to comment your code, also comment all other functions, variables etc. since the project is complex. This gives the investors or developers the opportunity to understand the code better.
#8	Debt	Returned value are not used	110	The returned “fundinfFee” of the “_updateMuxFundingFee” functionn is not used in the contract.
#9	Config	Remove function after deploying	97	Since the comment tells you to remove the function after deploying keep in mind to remove the function calls also in the contract. See _updateConfigs L36

#10	LibGmx	Require message is missing	196	We recommend you to add an error message to the requrie statement.
#11	LibGmx	Add zero check	198	Since the function “getPnl” is a public function in the contract we recommend you to add a zero check for the averagePriceUsd value in the function because of dividing by zero issue. Additionally use a better naming for the function because it can confuse investors

#12	LibGmx	Code simplyfying	200-204	Instead of using an “if-else” condition you can simplyfy the code as the following from
				
				if (isLong) {
				        hasProfit = priceUsd > averagePriceUsd;
				    } else {
				        hasProfit = averagePriceUsd > priceUsd;
				    }
				
				To
				
				hasProfit = isLong ? priceUsd > averagePriceUsd : averagePriceUsd > priceUsd

#13	Position	Values will not saved permanently	184-233	If you wanted to save the values permanently change the value to storage instead of memory.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

20. December 2022:

- Following contracts was not provided to solidproof and not part of the audit
    - PositionRouter
    - GmxOrderBook
    - Vault
- Read whole report and modifiers section for more information

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	**PASSED**
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**PASSED**
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY