AUDITED BY

LOVELY INSPECTOR

SMART CONTRACT

SECURITY AUDIT

ARKHAM TOKEN

inspector.lovely.finance

LOVELY INSPECTOR

TABLE OF CONTENTS

LOVELY INSPECTOR

DISCLAIMER

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws of the project's smart contract. Reading the full analysis report is essential to build your understanding of the project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully. DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. Scan and verify the report's presence in the GitHub repository by a QR code on the title page. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Inspector Lovely and its affiliates shall not be held responsible to you or anyone else, nor shall Inspector Lovely provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties, or other conditions other than as set forth in that exclusion Inspector Lovely excludes hereby all representations, warrants, conditions, and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Inspector Lovely disclaims all responsibility and responsibilities, and no claim against Inspector Lovely is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential, or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent). Security analysis is based only on smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

LOVELY INSPECTOR

AUDIT SCOPE

Name	Code Review and Security Analysis Report for Arkham Token Coin Smart Contract
Platform	Ethereum
File 1	ARKM.sol
Ethereum Code	0×2291323cf23d1553c6f79dc30b4a8865c03a90cf
Audit Date	November 8th, 2023

LOVELY INSPECTOR

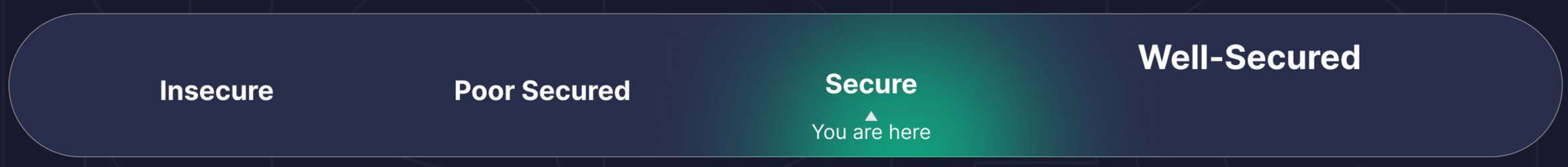
PROPOSED SMART CONTRACT FEATURES

Claimed Feature Detail	Our Observation
Tokenomics: • Name: Arkham • Symbol: AKRM • Decimals: 18 • Total Supply: 1 Billion	Validated
Ownership control: • The owner can pause/unpause the contract state. • Mint a new token. • Current owner can transfer the ownership. • Owner can renounce ownership	Validated

LOVELY INSPECTOR

AUDIT SUMMARY

According to the standard audit assessment, the Customer`s solidity-based smart contracts are **“Secured”**. Also, these contracts contain owner control, which does not make them fully decentralized.



We used various tools like Slither, Solhin,t, and Remix IDE. At the same time, this finding is based on a critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit Overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 0 low, and 0 very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner-controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

LOVELY INSPECTOR

KEY TECHNICAL METRICS

MAIN CATEGORY	SUBCATEGORY	RESULT
Contract Programming	Solidity version is not specified	Passed
	Solidity version is too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage is not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

LOVELY INSPECTOR

BUSINESS RISK ANALYSIS

CATEGORY	RESULT
Buy Tax	0%
Sell Tax	0%
Cannot Buy	Not Detected
Cannot Sell	Not Detected
Max Tax	0%
Modify Tax	Not Detected
Fee Check	No
Is Honeypot	Not Detected
Trading Cooldown	Not Detected
Can Pause Trade?	No
Pause Transfer?	No
Max Tax?	No
Is it Anti-whale?	No
Is Anti-bot?	Not Detected
Is it a Blacklist?	Not Detected
Blacklist Check	No
Can Mint?	No
Is it Proxy?	Not Detected
Can Take Ownership?	No
Hidden Owner?	Not Detected
Self Destruction?	Not Detected
Auditor Confidence	High

Overall Audit Result: PASSED

LOVELY INSPECTOR

CODE QUALITY

This audit scope has 1 smart contract. Smart contract contains Libraries, Smart contracts, inherits, and Interfaces. This is a compact and well written smart contract.

The libraries in Pendle Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties/methods can be reused many times by other contracts in the Pendle Token.

The EtherAuthority team has not provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

DOCUMENTATION

We were given a Pendle Token smart contract code in the form of an Etherscan web link.

As mentioned above, code parts are well commented on. and the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website: https://www.arkhamintelligence.com which provided rich information about the project architecture and tokenomics.

USE OF DEPENDENCIES

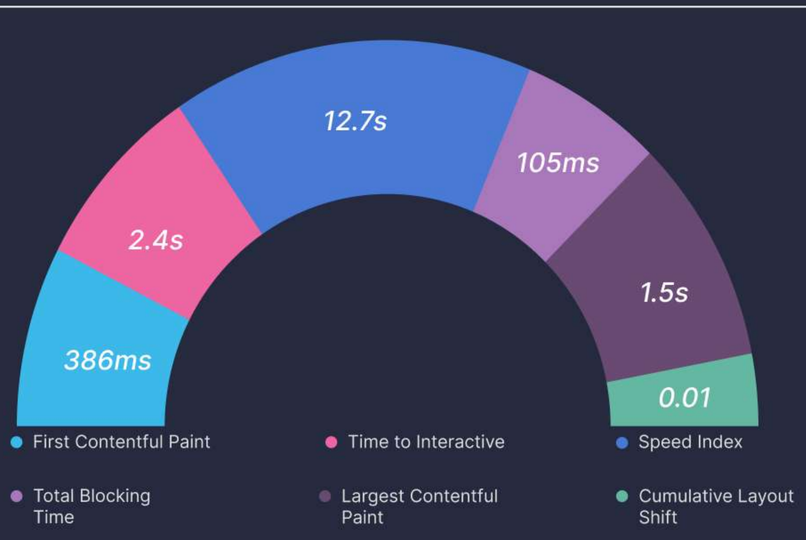
As per our observation, the libraries are used in this smart contract infrastructure that are based on well-known industry standard open-source projects.

Apart from libraries, its functions are not used in external smart contract calls.

LOVELY INSPECTOR

PROJECT WEBSITE PERFORMANCE AUDIT

Performance Metrics



Browser Timings

Redirect Duration **0ms**	Connection Duration **126ms**	Backend Duration **47ms**
Time to First Byte **173ms**	First Paint **387ms**	DOM Interactive Time **599ms**
DOM Content Loaded **624ms**	Onload Time **1.5s**	Fully Loaded Time **2.9s**

Grade

B	Performance 84%	Structure 86%

Web Vitals

LCP	TBT	CLS
1.5s	105ms	0.01

Top Issues

IMPECT	AUDIT
High	Avoid enormous network payloads (LCP)

URL	SIZE
https://d1e09wm0wupq53.cloudfront.net/arkham/webm/3.webm	12.0MB
https://d1e09wm0wupq53.cloudfront.net/arkham/webm/1.webm	9.9MB
https://d1e09wm0wupq53.cloudfront.net/arkham/webm/2.webm	8.34MB
https://d1e09wm0wupq53.cloudfront.net/arkham/webm/4.webm	7.09MB
https://assets.website-files.com/62879326fd745f7489b43224/62a0c38a3350bde261e4d2aa_ALERT-transcode.mp4	5.37MB
https://assets.website-files.com/62879326fd745f7489b43224/62a0c38a3350bde261e4d2aa_ALERT-transcode.mp4	1.18MB
https://assets-global.website-files.com/62879326fd745f7489b43224/62a3a7248e6a2b47b66bd26c_Arkham_pentagon_7_3_1-transcode.mp4	1.12MB
https://assets-global.website-files.com/62879326fd745f7489b43224/641e0dc90bc3e0bfb3d910a3_city-bg.webp	321KB
https://assets-global.website-files.com/6296255d9030be506dc09bb7/6421cc820a29f76a25e030d9_Screen%20Shot%202023-03-23%20at%202.03.34%20PM.png	297KB
https://assets-global.website-files.com/62879326fd745f7489b43224/js/arkham.ddd5eb7a3.js	165KB

LOVELY INSPECTOR

LEVEL OF CRITICALITY

RISK LEVEL	DESCRIPTION
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Med	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

LOVELY INSPECTOR

AUDIT FINDINGS TABLE

	Total	Resolved	UnResolved	Acknowledged
High Severity Issues Found	0	0	0	0
Moderate Severity Issues Found	0	0	0	0
Medium Severity Issues	0	0	0	0
Low Severity Issues	0	0	0	0
Informational Observations	0	0	0	0

The Arkham (ARKM) - Audit report identifies 0 issues with varying severity levels, discovered through manual review and static analysis techniques, alongside rigorous code reviews, highlighting the need for further investigation and vulnerability identification.

The smart contract is considered to **pass the audit,** as of the audit date, if no high severity or moderate severity issues are found.

LOVELY INSPECTOR

AUDIT FINDINGS

Critical Severity	No Critical severity vulnerabilities were found.
High Severity	No High severity vulnerabilities were found.
Medium	No Medium severity vulnerabilities were found.
Low	No Low severity vulnerabilities were found.
Very Low / Informational / Best practices:	No Very Low severity vulnerabilities were found.

LOVELY INSPECTOR

CENTRALIZATION

This smart contract has some functions that can be executed by the Admin (Owner) only. If the admin wallet's private key is compromised, then it would create trouble. Following are Admin functions:

ARKM.sol

- pause: The owner can trigger a stop.
- unpause: The owner can return to a normal state.
- mint: Mint a new token by the owner.
- _authorizeUpgrade: The owner can upgrade to a new implementation.

OwnableUpgradeable.sol

- renounce Ownership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: The current owner can transfer ownership of the contract to a new account..

UpgradeableBeacon.sol

- upgradeTo: Upgrades the beacon to a new implementation by the owner.

ProxyAdmin.sol

- changeProxyAdmin: Changes the admin of `proxy` to `newAdmin` by the owner.
- upgrade: Upgrades `proxy` to `implementation` by the owner.
- upgradeAndCall: Upgrades `proxy` to `implementation` and calls a function on the new implementation by the owner.

LOVELY INSPECTOR

TransparentUpgradeableProxy.sol

- admin: Returns the current admin by the Admin.
- implementation: Returns the current implementation by the Admin.
- changeAdmin: Changes the admin of the proxy by the Admin.
- upgradeTo: Upgrade the implementation of the proxy by the Admin.
- upgradeToAndCall: Upgrade the implementation of the proxy, and then call a function from the new implementation as specified by `data`, which should be an encoded function call by the Admin.

Ownable.sol

- renounce Ownership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: The current owner can transfer ownership of the contract to a new account..

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

LOVELY INSPECTOR

CONCLUSION

We were given a contract code in the form of Etherscan web links. And we have used all possible tests based on given objects as files. We had not observed any issues in the smart contracts. So, it's good to go for the production.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

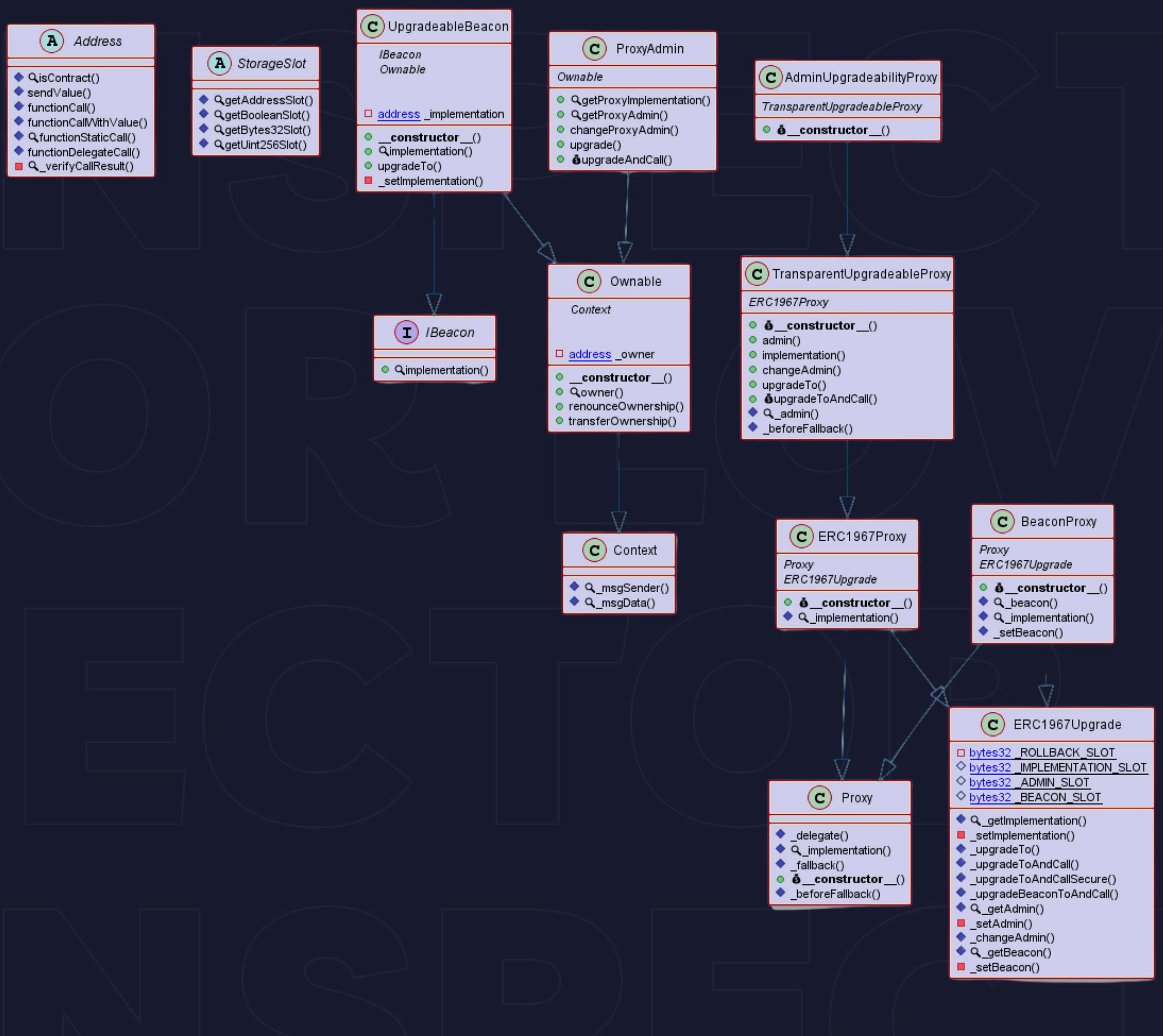
Security state of the reviewed smart contract, based on standard audit procedure scope, is **"Secured".**

LOVELY INSPECTOR

ADDENDUM

Code Flow Diagram

Arkham Token



LOVELY INSPECTOR

SECURITY ASSESSMENT REPORT

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project together. Below are the results.

Slither Log >> ERC1967Proxy.sol

```
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (ERC1967Proxy.sol#429-435) ignores return value by Address.functionDelegate
Call(newImplementation,data) (ERC1967Proxy.sol#433)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (ERC1967Proxy.sol#442-470) ignores return value by Address.functionDe
legateCall(newImplementation,data) (ERC1967Proxy.sol#448)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (ERC1967Proxy.sol#442-470) ignores return value by Address.functionDe
legateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (ERC1967Proxy.sol#456-462)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (ERC1967Proxy.sol#478-484) ignores return value by Address.functionDe
legateCall(IBeacon(newBeacon).implementation(),data) (ERC1967Proxy.sol#482)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

AdminUpgradeabilityProxy.constructor(address,address,bytes).admin (ERC1967Proxy.sol#849) shadows:
        - TransparentUpgradeableProxy.admin() (ERC1967Proxy.sol#653-655) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Modifier TransparentUpgradeableProxy.ifAdmin() (ERC1967Proxy.sol#636-642) does not always execute _; or revertReference: https:/
/github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier

Reentrancy in ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (ERC1967Proxy.sol#442-470):
        External calls:
        - Address.functionDelegateCall(newImplementation,data) (ERC1967Proxy.sol#448)
        - Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (ERC1967
Proxy.sol#456-462)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (ERC1967Proxy.sol#468)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

LOVELY INSPECTOR

```
Address.isContract(address) (PENDLE.sol#26-35) uses assembly
        - INLINE ASM (PENDLE.sol#33)
Address._verifyCallResult(bool,bytes,string) (PENDLE.sol#171-188) uses assembly
        - INLINE ASM (PENDLE.sol#180-183)
PENDLE.getChainId() (PENDLE.sol#1052-1058) uses assembly
        - INLINE ASM (PENDLE.sol#1054-1056)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.functionCall(address,bytes) (PENDLE.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (PENDLE.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (PENDLE.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (PENDLE.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (PENDLE.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (PENDLE.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (PENDLE.sol#53-59) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (PENDLE.sol#479-488) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (PENDLE.sol#495-498) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (PENDLE.sol#490-493) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (PENDLE.sol#468-470) is never used and should be removed
SafeMath.div(uint256,uint256,string) (PENDLE.sol#434-437) is never used and should be removed
SafeMath.mod(uint256,uint256) (PENDLE.sol#396-399) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (PENDLE.sol#454-457) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (PENDLE.sol#268-272) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (PENDLE.sol#304-307) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (PENDLE.sol#314-317) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (PENDLE.sol#289-297) is never used and should be removed
SafeMath.trySub(uint256,uint256) (PENDLE.sol#279-282) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.2<0.8.0 (PENDLE.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Pragma version^0.8.0 (ERC1967Proxy.sol#3) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (ERC1967Proxy.sol#49-55):
        - (success) = recipient.call{value: amount}() (ERC1967Proxy.sol#53)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ERC1967Proxy.sol#110-117):
        - (success,returndata) = target.call{value: value}(data) (ERC1967Proxy.sol#115)
Low level call in Address.functionStaticCall(address,bytes,string) (ERC1967Proxy.sol#135-141):
        - (success,returndata) = target.staticcall(data) (ERC1967Proxy.sol#139)
Low level call in Address.functionDelegateCall(address,bytes,string) (ERC1967Proxy.sol#159-165):
        - (success,returndata) = target.delegatecall(data) (ERC1967Proxy.sol#163)
Low level call in ProxyAdmin.getProxyImplementation(TransparentUpgradeableProxy) (ERC1967Proxy.sol#730-736):
        - (success,returndata) = address(proxy).staticcall(0x5c60da1b) (ERC1967Proxy.sol#733)
Low level call in ProxyAdmin.getProxyAdmin(TransparentUpgradeableProxy) (ERC1967Proxy.sol#745-751):
        - (success,returndata) = address(proxy).staticcall(0xf851a440) (ERC1967Proxy.sol#748)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Redundant expression "this (ERC1967Proxy.sol#256)" inContext (ERC1967Proxy.sol#250-259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable UpgradeableBeacon._implementation (ERC1967Proxy.sol#795) is too similar to UpgradeableBeacon.constructor(address).imple
mentation_ (ERC1967Proxy.sol#806)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
ERC1967Proxy.sol analyzed (13 contracts with 84 detectors), 40 result(s) found
```

LOVELY INSPECTOR

SOLIDITY STATIC ANALYSIS

Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

ARKM.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 287:12:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.

more

Pos: 779:50:

LOVELY INSPECTOR

Gas costs:

Gas requirement of function ARKM.mint is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1437:4:

Gas costs:

Gas requirement of function ARKM.pause is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1429:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1313:12:

LOVELY INSPECTOR

ERC1967Proxy.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 269:8:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.

more

Pos: 163:50:

Gas costs:

Gas requirement of function TransparentUpgradeableProxy.admin is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 653:4:

LOVELY INSPECTOR

ERC1967Proxy.sol

Gas costs:

Fallback function of contract AdminUpgradeabilityProxy requires too much gas (infinite). If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 309:4:

Gas costs:

Fallback function of contract BeaconProxy requires too much gas (infinite). If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 309:4:

Similar variable names:

UpgradeableBeacon.(address) : Variables have very similar names "_implementation" and "implementation_". Note: Modifiers are currently not considered by this static analysis.
Pos: 807:27:

No return:

Proxy._implementation(): Defines a return type but never explicitly returns a value.
Pos: 293:4:

LOVELY INSPECTOR

ERC1967Proxy.sol

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 749:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 840:8:

LOVELY INSPECTOR

COMPLIANCE ANALYSIS

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

ARKM.sol

Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement
Pos: 1:3
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:26
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:36
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:46
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:56
Error message for require is too long
Pos: 9:139
Error message for require is too long
Pos: 9:209
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 13:286
Error message for require is too long
Pos: 9:416
Error message for require is too long
Pos: 9:450
Error message for require is too long
Pos: 9:463
Error message for require is too long
Pos: 9:476
Function name must be in mixedCase
Pos: 5:499
Code contains empty blocks
Pos: 57:499
Function name must be in mixedCase
Pos: 5:502
Code contains empty blocks
Pos: 67:502
Function name must be in mixedCase
Pos: 5:528
Function name must be in mixedCase
Pos: 5:532
Error message for require is too long
Pos: 9:574
Function name must be in mixedCase
Pos: 5:597

LOVELY INSPECTOR

Code contains empty blocks
Pos: 64:597
Function name must be in mixedCase
Pos: 5:600
Code contains empty blocks
Pos: 74:600
Error message for require is too long
Pos: 9:628
Error message for require is too long
Pos: 17:675
Error message for revert is too long
Pos: 17:677
Error message for require is too long
Pos: 9:706
Error message for require is too long
Pos: 9:742
Error message for require is too long
Pos: 9:743
Error message for require is too long
Pos: 9:775
Function name must be in mixedCase
Pos: 5:803
Code contains empty blocks
Pos: 65:803
Function name must be in mixedCase
Pos: 5:806
Code contains empty blocks
Pos: 75:806
Error message for require is too long
Pos: 9:819
Error message for require is too long
Pos: 9:820
Error message for require is too long
Pos: 9:829
Function name must be in mixedCase
Pos: 5:915
Function name must be in mixedCase
Pos: 5:919
Function name must be in mixedCase
Pos: 5:1019
Function name must be in mixedCase
Pos: 5:1023
Error message for require is too long
Pos: 9:1173
Error message for require is too long
Pos: 9:1200
Error message for require is too long
Pos: 9:1201
Error message for require is too long
Pos: 9:1206
Error message for require is too long
Pos: 9:1255
Error message for require is too long

LOVELY INSPECTOR

Pos: 9:1260
Error message for require is too long
Pos: 9:1290
Error message for require is too long
Pos: 9:1291
Code contains empty blocks
Pos: 24:1337
Code contains empty blocks
Pos: 24:1357
Function name must be in mixedCase
Pos: 5:1373
Code contains empty blocks
Pos: 63:1373
Function name must be in mixedCase
Pos: 5:1376
Code contains empty blocks
Pos: 73:1376
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:1413
Visibility modifier must be first in list of modifiers
Pos: 39:1417
Code contains empty blocks
Pos: 5:1452

ERC1967Proxy.sol

Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement
Pos: 1:2
Error message for require is too long
Pos: 9:53
Error message for require is too long
Pos: 9:110
Error message for require is too long
Pos: 9:135
Error message for require is too long
Pos: 9:159
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:207
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:216
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:225
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:234
Code contains empty blocks
Pos: 49:326
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:338

LOVELY INSPECTOR

Error message for require is too long
Pos: 9:376
Error message for require is too long
Pos: 9:409
Error message for require is too long
Pos: 13:464
Error message for require is too long
Pos: 9:508
Error message for require is too long
Pos: 9:544
Error message for require is too long
Pos: 9:548
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:563
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:588
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:627
Error message for require is too long
Pos: 9:711
Provide an error message for require
Pos: 9:733
Provide an error message for require
Pos: 9:748
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:805
Error message for require is too long
Pos: 9:839
Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
Pos: 5:848
Code contains empty blocks
Pos: 122:848

LOVELY INSPECTOR

SOFTWARE ANALYSIS RESULT

This software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

inspector.lovely.finance

Audited by LOVELY INSPECTOR

LOVELY
INSPECTOR

INSPECTOR LOVELY

INFO

Website: Inspector.lovely.finance

Telegram community: t.me/inspectorlovely

Twitter: twitter.com/InspectorLovely

inspector.lovely.finance