INSPECTOR LOVELY

SMART CONTRACT

SECURITY AUDIT

FIRST DIGITAL USD

inspector.lovely.finance

INSPECTOR LOVELY

TABLE OF CONTENTS

INSPECTOR
LOVELY

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 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 the 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 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 and 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 the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

INSPECTOR LOVELY

AUDIT SCOPE

Name	Code Review and Security Analysis Report for First Digital USD Token Coin Smart Contract
Platform	Ethereum
Language	Solidity
File	Stablecoin.sol
Ethereum Code	0xda1814d75ef1c42d0a4e6abe0d43d49a1d300c8d
Audit Date	November 8th, 2023

INSPECTOR LOVELY

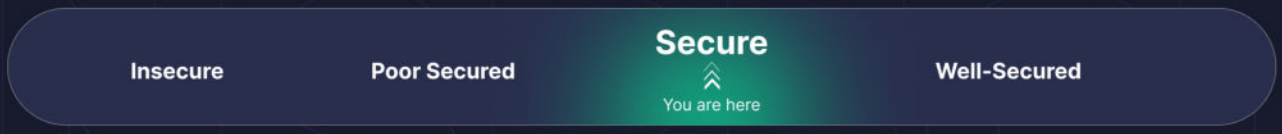
PROPOSED SMART CONTRACT FEATURES

Claimed Feature Detail	Our Observation
Tokenomics: • Name: First Digital USD • Symbol: FDUSD • Decimals: 18	Validated
Ownership control: • The owner can pause/unpause the contract state. • Burn amount. • Updates account to frozen state. • Mint a new token. • Current owner can transfer the ownership. • The new owner accepts the ownership transfer	Validated

INSPECTOR LOVELY

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, Solhint, 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.

INSPECTOR LOVELY

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

INSPECTOR LOVELY

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?	Yes
Pause Transfer?	Yes
Max Tax?	No
Is it Anti-whale?	No
Is Anti-bot?	Not Detected
Is it a Blacklist?	Not Detected
Blacklist Check	No
Can Mint?	Yes
Is it Proxy?	Yes
Can Take Ownership?	Yes
Hidden Owner?	Not Detected
Self Destruction?	Not Detected
Auditor Confidence	High

Overall Audit Result: PASSED

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 First Digital USD 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 First Digital USD 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 First Digital USD 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.

USE OF DEPENDENCIES

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

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

INSPECTOR LOVELY

PROJECT WEBSITE PERFORMANCE AUDIT

Performance Metrics

Browser Timings

Redirect Duration **0ms**	Connection Duration **27ms**	Backend Duration **418ms**
Time to First Byte **445ms**	First Paint **1.5s**	DOM Interactive Time **1.5s**
DOM Content Loaded **1.5s**	Onload Time **1.5s**	Fully Loaded Time **1.9s**

Grade

A	Performance 89%	Structure 94%

Web Vitals

LCP	TBT	CLS
1.5s	0ms	0

Top Issues

IMPECT	AUDIT
High	Use explicit width and height on image elements (LCP) — 21 Images found

FAILING ELEMENTS

div.container > div.row > div.col-md-8 > img.img-fluid

```
<img src="https://firstdigitallabs.com/workspace/img/svg/hero.png" class="img-fluid">
```

Pancake Swap

```
<img alt="Pancake Swap" src="https://firstdigitallabs.com/workspace/uploads/pancake-swap-logo-653a3a347…" class="logo">
```

FDUSD September 2023 Attestation Report

```
<img src="https://firstdigitallabs.com/workspace/img/png/fdusd-card-thumb.png" class="card-img-top" alt="FDUSD September 2023 Attestation Report">
```

FDUSD August 2023 Attestation Report

```
<img src="https://firstdigitallabs.com/workspace/img/png/fdusd-card-thumb.png" class="card-img-top" alt="FDUSD August 2023 Attestation Report">
```

FDUSD July 2023 Attestation Report

```
<img src="https://firstdigitallabs.com/workspace/img/png/fdusd-card-thumb.png" class="card-img-top" alt="FDUSD July 2023 Attestation Report">
```

Gate.io

```
<img alt="Gate.io" src="https://firstdigitallabs.com/workspace/uploads/gate-io-logo-654605603dc20.…" class="logo">
```

Uniswap

```
<img alt="Uniswap" src="https://firstdigitallabs.com/workspace/uploads/uniswap-logo-653a3aba0ae2e.…" class="logo">
```

BTSE

```
<img alt="BTSE" src="https://firstdigitallabs.com/workspace/uploads/btse-logo-653a309460363.svg" class="logo">
```

INSPECTOR LOVELY

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.

INSPECTOR LOVELY

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 First Digital USD (FDUSD) Token - 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.

INSPECTOR LOVELY

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.

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. The following are Admin functions:

Stablecoin.sol

- pause: The owner can trigger a stop.
- unpause: The owner can return to a normal state.
- mint: Mint a new amount by the owner.
- burn: Burn amount by the owner.
- freeze: Adds account to the frozen state by the owner.
- unfreeze: Removes account from frozen state by the owner.

Ownable2StepUpgradeable.sol

- transferOwnership: The current owner can transfer ownership of the contract to a new account.
- acceptOwnership: The new owner accepts the ownership transfer by the current owner.

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.

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

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 the 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.

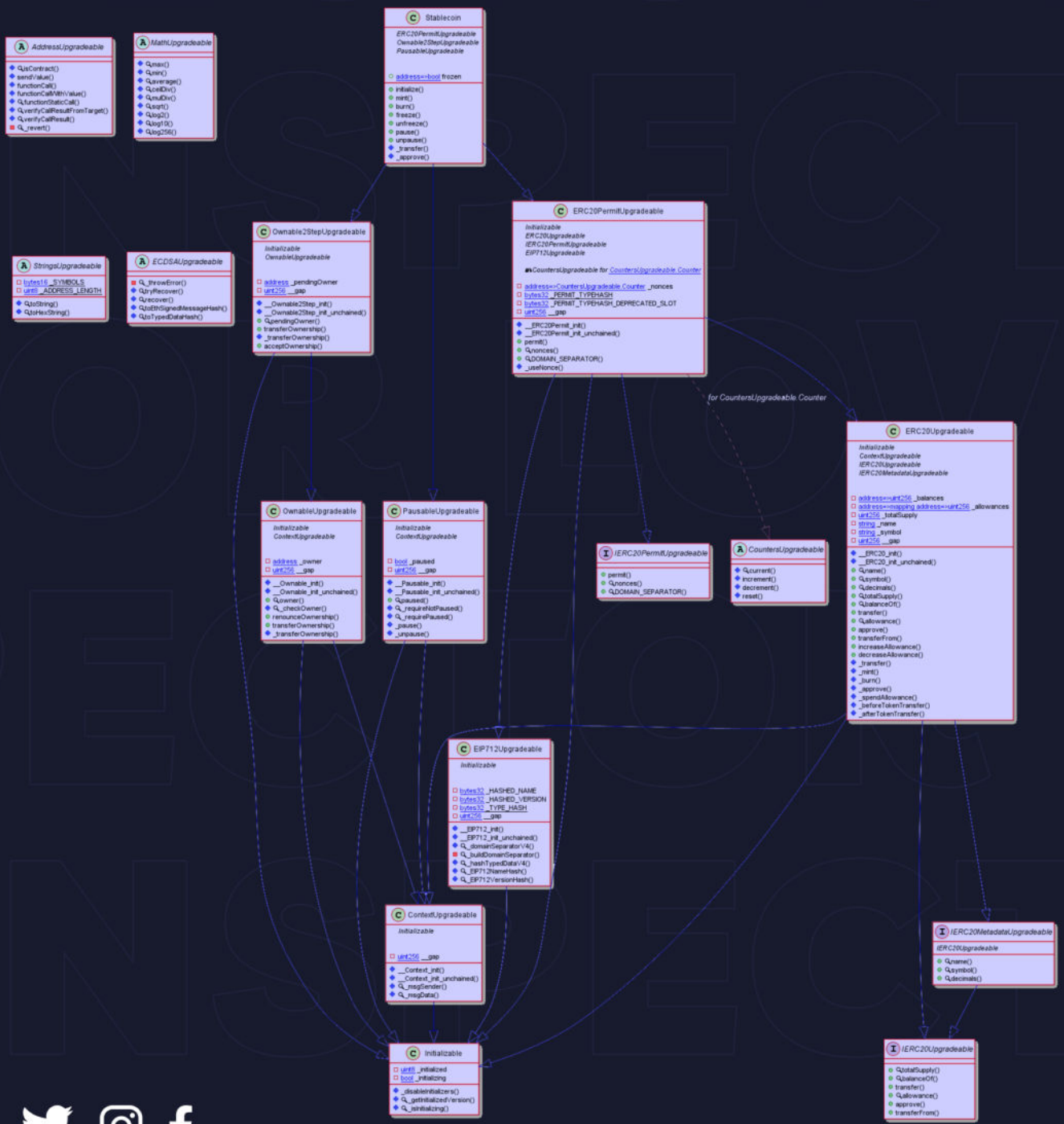
Security state of the reviewed smart contract, based on standard audit procedure scope, is **"Secured"**.

INSPECTOR LOVELY

ADDENDUM

Code Flow Diagram

First Digital USD Token



inspector.lovely.finance

Audited by INSPECTOR LOVELY

INSPECTOR LOVELY

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 >> Stablecoin.sol

```
ERC20PermitUpgradeable.__ERC20Permit_init(string).name (Stablecoin.sol#997) shadows:
        - ERC20Upgradeable.name() (Stablecoin.sol#785-787) (function)
        - IERC20MetadataUpgradeable.name() (Stablecoin.sol#406) (function)
Stablecoin.initialize(string,string)._name (Stablecoin.sol#1051) shadows:
        - ERC20Upgradeable._name (Stablecoin.sol#773) (state variable)
Stablecoin.initialize(string,string)._symbol (Stablecoin.sol#1051) shadows:
        - ERC20Upgradeable._symbol (Stablecoin.sol#774) (state variable)
Stablecoin._approve(address,address,uint256).owner (Stablecoin.sol#1098) shadows:
        - OwnableUpgradeable.owner() (Stablecoin.sol#656-658) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Ownable2StepUpgradeable.transferOwnership(address).newOwner (Stablecoin.sol#697) lacks a zero-check on :
                - _pendingOwner = newOwner (Stablecoin.sol#698)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

ERC20PermitUpgradeable.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (Stablecoin.sol#1003-1022) uses timestamp f
or comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,ERC20Permit: expired deadline) (Stablecoin.sol#1012)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable._revert(bytes,string) (Stablecoin.sol#105-114) uses assembly
        - INLINE ASM (Stablecoin.sol#107-110)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (Stablecoin.sol#169-222) uses assembly
        - INLINE ASM (Stablecoin.sol#177-181)
        - INLINE ASM (Stablecoin.sol#191-196)
        - INLINE ASM (Stablecoin.sol#200-206)
StringsUpgradeable.toString(uint256) (Stablecoin.sol#417-435) uses assembly
        - INLINE ASM (Stablecoin.sol#422-424)
        - INLINE ASM (Stablecoin.sol#427-429)
ECDSAUpgradeable.tryRecover(bytes32,bytes) (Stablecoin.sol#481-495) uses assembly
        - INLINE ASM (Stablecoin.sol#486-490)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
AddressUpgradeable._revert(bytes,string) (Stablecoin.sol#105-114) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (Stablecoin.sol#33-35) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (Stablecoin.sol#37-43) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (Stablecoin.sol#45-51) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (Stablecoin.sol#53-62) is never used and should be remove
d
AddressUpgradeable.functionStaticCall(address,bytes) (Stablecoin.sol#64-66) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (Stablecoin.sol#68-75) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (Stablecoin.sol#26-31) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (Stablecoin.sol#93-103) is never used and should be removed
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (Stablecoin.sol#77-91) is never used and should be remo
ved
ContextUpgradeable.__Context_init_unchained() (Stablecoin.sol#625-626) is never used and should be removed
ContextUpgradeable._msgData() (Stablecoin.sol#631-633) is never used and should be removed
CountersUpgradeable.decrement(CountersUpgradeable.Counter) (Stablecoin.sol#133-139) is never used and should be removed
CountersUpgradeable.reset(CountersUpgradeable.Counter) (Stablecoin.sol#141-143) is never used and should be removed
ECDSAUpgradeable.recover(bytes32,bytes) (Stablecoin.sol#497-501) is never used and should be removed
ECDSAUpgradeable.recover(bytes32,bytes32,bytes32) (Stablecoin.sol#513-521) is never used and should be removed
ECDSAUpgradeable.toEthSignedMessageHash(bytes) (Stablecoin.sol#556-558) is never used and should be removed
ECDSAUpgradeable.toEthSignedMessageHash(bytes32) (Stablecoin.sol#552-554) is never used and should be removed
ECDSAUpgradeable.tryRecover(bytes32,bytes) (Stablecoin.sol#481-495) is never used and should be removed
ECDSAUpgradeable.tryRecover(bytes32,bytes32,bytes32) (Stablecoin.sol#503-511) is never used and should be removed
EIP712Upgradeable.__EIP712_init(string,string) (Stablecoin.sol#950-952) is never used and should be removed
ERC20PermitUpgradeable.__ERC20Permit_init_unchained(string) (Stablecoin.sol#1001) is never used and should be removed
Initializable._disableInitializers() (Stablecoin.sol#604-610) is never used and should be removed
Initializable._getInitializedVersion() (Stablecoin.sol#612-614) is never used and should be removed
Initializable._isInitializing() (Stablecoin.sol#616-618) is never used and should be removed
MathUpgradeable.average(uint256,uint256) (Stablecoin.sol#161-163) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (Stablecoin.sol#165-167) is never used and should be removed
MathUpgradeable.log10(uint256) (Stablecoin.sol#308-340) is never used and should be removed
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (Stablecoin.sol#342-347) is never used and should be removed
MathUpgradeable.log2(uint256) (Stablecoin.sol#263-299) is never used and should be removed
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (Stablecoin.sol#301-306) is never used and should be removed
MathUpgradeable.log256(uint256) (Stablecoin.sol#349-373) is never used and should be removed
MathUpgradeable.log256(uint256,MathUpgradeable.Rounding) (Stablecoin.sol#375-380) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (Stablecoin.sol#153-155) is never used and should be removed
```

```
MathUpgradeable.min(uint256,uint256) (Stablecoin.sol#157-159) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256) (Stablecoin.sol#169-222) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (Stablecoin.sol#224-235) is never used and should be re
moved
MathUpgradeable.sqrt(uint256) (Stablecoin.sol#237-254) is never used and should be removed
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (Stablecoin.sol#256-261) is never used and should be removed
Ownable2StepUpgradeable.__Ownable2Step_init_unchained() (Stablecoin.sol#687-688) is never used and should be removed
OwnableUpgradeable.__Ownable_init() (Stablecoin.sol#643-645) is never used and should be removed
StringsUpgradeable.toHexString(address) (Stablecoin.sol#455-457) is never used and should be removed
StringsUpgradeable.toHexString(uint256) (Stablecoin.sol#437-441) is never used and should be removed
StringsUpgradeable.toHexString(uint256,uint256) (Stablecoin.sol#443-453) is never used and should be removed
StringsUpgradeable.toString(uint256) (Stablecoin.sol#417-435) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (Stablecoin.sol#2) 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 AddressUpgradeable.sendValue(address,uint256) (Stablecoin.sol#26-31):
        - (success) = recipient.call{value: amount}() (Stablecoin.sol#29)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (Stablecoin.sol#53-62):
        - (success,returndata) = target.call{value: value}(data) (Stablecoin.sol#60)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (Stablecoin.sol#68-75):
        - (success,returndata) = target.staticcall(data) (Stablecoin.sol#73)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function IERC20PermitUpgradeable.DOMAIN_SEPARATOR() (Stablecoin.sol#17) is not in mixedCase
Function ContextUpgradeable.__Context_init() (Stablecoin.sol#622-623) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (Stablecoin.sol#625-626) is not in mixedCase
Variable ContextUpgradeable.__gap (Stablecoin.sol#635) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (Stablecoin.sol#643-645) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (Stablecoin.sol#647-649) is not in mixedCase
Variable OwnableUpgradeable.__gap (Stablecoin.sol#679) is not in mixedCase
Function Ownable2StepUpgradeable.__Ownable2Step_init() (Stablecoin.sol#683-685) is not in mixedCase
Function Ownable2StepUpgradeable.__Ownable2Step_init_unchained() (Stablecoin.sol#687-688) is not in mixedCase
Variable Ownable2StepUpgradeable.__gap (Stablecoin.sol#713) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (Stablecoin.sol#723-725) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (Stablecoin.sol#727-729) is not in mixedCase
Variable PausableUpgradeable.__gap (Stablecoin.sol#763) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (Stablecoin.sol#776-778) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (Stablecoin.sol#780-783) is not in mixedCase
Variable ERC20Upgradeable.__gap (Stablecoin.sol#940) is not in mixedCase
Function EIP712Upgradeable.__EIP712_init(string,string) (Stablecoin.sol#950-952) is not in mixedCase
Function EIP712Upgradeable.__EIP712_init_unchained(string,string) (Stablecoin.sol#954-959) is not in mixedCase
Function EIP712Upgradeable._EIP712NameHash() (Stablecoin.sol#977-979) is not in mixedCase
Function EIP712Upgradeable._EIP712VersionHash() (Stablecoin.sol#981-983) is not in mixedCase
Variable EIP712Upgradeable._HASHED_NAME (Stablecoin.sol#945) is not in mixedCase
Variable EIP712Upgradeable._HASHED_VERSION (Stablecoin.sol#946) is not in mixedCase
Variable EIP712Upgradeable.__gap (Stablecoin.sol#985) is not in mixedCase
Function ERC20PermitUpgradeable.__ERC20Permit_init(string) (Stablecoin.sol#997-999) is not in mixedCase
Function ERC20PermitUpgradeable.__ERC20Permit_init_unchained(string) (Stablecoin.sol#1001) is not in mixedCase
Function ERC20PermitUpgradeable.DOMAIN_SEPARATOR() (Stablecoin.sol#1028-1030) is not in mixedCase
Variable ERC20PermitUpgradeable._PERMIT_TYPEHASH_DEPRECATED_SLOT (Stablecoin.sol#995) is not in mixedCase
Variable ERC20PermitUpgradeable.__gap (Stablecoin.sol#1038) is not in mixedCase
Parameter Stablecoin.initialize(string,string)._name (Stablecoin.sol#1051) is not in mixedCase
Parameter Stablecoin.initialize(string,string)._symbol (Stablecoin.sol#1051) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
Stablecoin.sol analyzed (17 contracts with 84 detectors), 98 result(s) found
```

INSPECTOR LOVELY

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.

Stablecoin.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 173:4:

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: 838:16:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 1854:23:

INSPECTOR LOVELY

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 180:50:

Gas costs:

Gas requirement of function Stablecoin.unpause 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: 1980:11:

Constant/View/Pure functions:

Stablecoin._approve(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2000:11:

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: 1922:15:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1964:15:

INSPECTOR LOVELY

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.

Stablecoin.sol

```
Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement
Pos: 1:3
Error message for require is too long
Pos: 9:108
Error message for require is too long
Pos: 9:178
Error message for require is too long
Pos: 9:1018
Error message for require is too long
Pos: 9:1052
Error message for require is too long
Pos: 9:1065
Error message for require is too long
Pos: 9:1078
Function name must be in mixedCase
Pos: 5:1101
Code contains empty blocks
Pos: 57:1101
Function name must be in mixedCase
Pos: 5:1104
Code contains empty blocks
Pos: 67:1104
Function name must be in mixedCase
Pos: 5:1130
Function name must be in mixedCase
Pos: 5:1134
Error message for require is too long
Pos: 9:1176
Function name must be in mixedCase
Pos: 5:1199
Function name must be in mixedCase
Pos: 5:1203
Code contains empty blocks
Pos: 72:1203
Error message for require is too long
Pos: 9:1239
Function name must be in mixedCase
Pos: 5:1267
Function name must be in mixedCase
Pos: 5:1271
Function name must be in mixedCase
Pos: 5:1371
```

INSPECTOR LOVELY

```
Function name must be in mixedCase
Pos: 5:1375
Error message for require is too long
Pos: 9:1525
Error message for require is too long
Pos: 9:1552
Error message for require is too long
Pos: 9:1553
Error message for require is too long
Pos: 9:1558
Error message for require is too long
Pos: 9:1607
Error message for require is too long
Pos: 9:1612
Error message for require is too long
Pos: 9:1642
Error message for require is too long
Pos: 9:1643
Code contains empty blocks
Pos: 24:1689
Code contains empty blocks
Pos: 24:1709
Function name must be in mixedCase
Pos: 5:1740
Function name must be in mixedCase
Pos: 5:1744
Function name must be in mixedCase
Pos: 5:1791
Function name must be in mixedCase
Pos: 5:1801
Function name must be in mixedCase
Pos: 5:1835
Function name must be in mixedCase
Pos: 5:1839
Code contains empty blocks
Pos: 84:1839
Avoid making time-based decisions in your business logic
Pos: 17:1853
```

SOFTWARE ANALYSIS RESULT

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

INSPECTOR
LOVELY

INSPECTOR LOVELY

INFO

Website: Inspector.lovely.finance

Telegram community: t.me/inspectorlovely

Twitter: twitter.com/InspectorLovely

inspector.lovely.finance