Assure DeFi™

The Verification **Gold Standard ™**

Security Assessment

Deluge.Cash Locker

December 16, 2023

Audit Status: Fail

Audit Edition: Advance

ASSURE DEFI™

THE VERIFICATION GOLD STANDARD

Project Overview

Token Summary

Parameter	Result
Address	0x2F1060944a890B318FBEEd0531E308F1419A7F69
Name	Deluge.Cash
Token Tracker	Deluge.Cash (Flash)
Decimals	0
Supply	0
Platform	Ethereum
compiler	v0.8.19+commit.7dd6d404
Contract Name	ETHUniswapV2Locker
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/address/0x2F1060944a890B318FBEEd0531E308F1419A7F69#code
Payment Tx	Corporate

Main Contract Assessed
Contract Name

Name	Contract	Live
Deluge.Cash	0x2F1060944a890B318FBEEd0531E308F1419A7F69	No

TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
Deluge	973ea649e6231d227d9aabb660bb9e4ac38a6969	ETHUniswapV2Locker.sol
Deluge		
Deluge		
Deluge		

Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	ETHUniswapV2Locker.sol	L: 8 C: 0
SWC-104	Pass	Unchecked Call Return Value.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-107	Low	Read of persistent state following external call.	ETHUniswapV2Locker.sol	L: 613 C: 39,L: 616 C: 17,L: 508 C: 39,L: 511 C: 16,L: 388 C: 17,L: 618 C: 16,L: 512 C: 17,L: 237 C: 17,L: 390 C: 16,L: 698 C: 17,L: 700 C: 16
SWC-108	Low	State variable visibility is not set..	ETHUniswapV2Locker.sol	L: 125 C: 12,L: 126 C: 20,L: 128 C: 12

ID	Severity	Name	File	location
SWC-109	Pass	Uninitialized Storage Pointer.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-113	Medium	Multiple calls are executed in the same transaction.	ETHUniswapV2Locker.sol	L: 238 C: 16,L: 515 C: 12,L: 392 C: 12,L: 702 C: 12,L: 620 C: 12,L: 396 C: 12
SWC-114	Pass	Transaction Order Dependence.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	ETHUniswapV2Locker.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-124	Pass	Write to Arbitrary Storage Location.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	ETHUniswapV2Locker.sol	L: 0 C: 0
SWC-135	Low	Code With No Effects (Irrelevant/Dead Code).	ETHUniswapV2Locker.sol	L: 173 C: 8
SWC-136	Pass	Unencrypted Private Data On-Chain.	ETHUniswapV2Locker.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to 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.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Smart Contract Vulnerability Details

SWC-107 - Reentrancy.

CWE-841: Improper Enforcement of Behavioral Workflow.

Description:

One of the major dangers of calling external contracts is that they can take over the control flow. In the reentrancy attack (a.k.a. recursive call attack), a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways.

Remediation:

The best practices to avoid Reentrancy weaknesses are: Make sure all internal state changes are performed before the call is executed. This is known as the Checks-Effects-Interactions pattern Use a reentrancy lock.

References:

Ethereum Smart Contract Best Practices - Reentrancy

Smart Contract Vulnerability Details

SWC-108 - State Variable Default Visibility

CWE-710: Improper Adherence to Coding Standards

Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Remediation:

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables

Smart Contract Vulnerability Details

SWC-113 - DoS with Failed Call

CWE-703: Improper Check or Handling of Exceptional Conditions

Description:

External calls can fail accidentally or deliberately, which can cause a DoS condition in the contract. To minimize the damage caused by such failures, it is better to isolate each external call into its own transaction that can be initiated by the recipient of the call. This is especially relevant for payments, where it is better to let users withdraw funds rather than push funds to them automatically (this also reduces the chance of problems with the gas limit).

Remediation:

It is recommended to follow call best practices:Avoid combining multiple calls in a single transaction, especially when calls are executed as part of a loop. Always assume that external calls can fail. Implement the contract logic to handle failed calls

References:

ConsenSys Smart Contract Best Practices

Smart Contract Vulnerability Details

SWC-135 - Code With No Effects

CWE-1164: Irrelevant Code

Description:

In Solidity, it's possible to write code that does not produce the intended effects. Currently, the solidity compiler will not return a warning for effect-free code. This can lead to the introduction of "dead" code that does not properly performing an intended action.

For example, it's easy to miss the trailing parentheses in msg.sender.call.value(address(this).balance)("");, which could lead to a function proceeding without transferring funds to msg.sender. Although, this should be avoided by checking the return value of the call.

Remediation:

It s important to carefully ensure that your contract works as intended. Write unit tests to verify correct behaviour of the code.

References:

Issue on Solidity's Github - raise an error when a statement can never have side-effects

Issue on Solidity's Github - msg.sender.call.value(address(this).balance); should produce a warning

Inheritance

The contract for Deluge.Cash has the following inheritance structure.

The Project has a Total Supply of 0

Flash-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	Low	ETHUniswapV2Locker.sol: L: 125 C: 13,L: 126 C: 13,L: 128 C: 13	Detected

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
owner		public
devaddr		public
_uniswapRouter		public

The functions that are never called internally within the contract should have external visibility

Remediation

We advise that the function's visibility specifiers are set to external, and the array-based arguments change their data location from memory to calldata, optimizing the gas cost of the function.

References:

external vs public best practices.

Flash-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Low	ETHUniswapV2Locker.sol: L: 205, C: 14,L: 201, C: 14,L: 197, C: 14,L: 193, C: 14,L: 188, C: 14,L: 184, C: 14,,L: 180, C: 14,L: 176, C: 14,L: 172, C: 14	Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the missing required function.

Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
 require(receiver != address(0), "Receiver is the zero address");
...
...
require(value X limitation, "Your not able to do this function");
...
```

We also recommend customer to review the following function that is missing a required validation. missing required function.

Flash-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Low	ETHUniswapV2Locker.sol: L: 205, C: 14,L: 201, C: 14,L: 197, C: 14,L: 193, C: 14,L: 188, C: 14,L: 184, C: 14,,L: 180, C: 14,L: 176, C: 14,L: 172, C: 14,L: 167, C: 14	Detected

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

Remediation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Flash-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	Low	ETHUniswapV2Locker.sol: L: 172, C: 14	Detected

Description

Solidity defines a naming convention that should be followed. Rule exceptions: Allow constant variable name/symbol/decimals to be lowercase. Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
changeaddress
```

Remediation

Follow the Solidity naming convention.

https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-convention

Flash-11 | MigrateLock function part of iMigrate can be used as bad actor..

Category	Severity	Location	Status
Optimization	Critical	ETHUniswapV2Locker.sol: L: 305, C: 14	Detected

Description

MigrateLock and move log to a new contract, this can be a function that can be compromised.

Remediation

Recommended to remove MigrateLock, if a new contract is needed it should be v1 and v2 and leave lock founds alone in first contrac.

Project Action

Flash-16 | Taxes can be up to 100%.

Category	Severity	Location	Status
Logical Issue	Critical	ETHUniswapV2Locker.sol: L: 225 C: 21	Detected

Description

The current definition of taxes can be set up to 100% for specific wallets, we suggest to modify the function not to be dynamic but to be a static resolution.

feeInTokens > senderBalance &&
(feeInTokens / 100) * 95 <= senderBalance

due to the logic written in here may results in loss of funds.

Remediation

We advise the team to review the following logic..

Project Action

Flash-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	Critical	ETHUniswapV2Locker.sol: L: 201 C:13,L: 197 C:13	Detected

Description

Enable Trade is presend on the following contract and when combined with Exclude from fees it can be considered a whitelist process, this will allow anyone to trade before others and can represent and issue for the holders.

Remediation

We recommend the project owner to carefully review this function and avoid problems when performing both actions.

Project Action

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
High	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Low	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	3	3	0
High	0	0	0
Medium	0	0	0
Low	3	3	0
Informational	1	1	0
Total	7	7	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/delugecash	Pass
Other	https://t.me/krabsdeluge	Pass
Website	https://deluge.cash/	Pass
Telegram	https://t.me/delugecash	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:

Audit Result

Final Audit Score

Review	Score
Security Score	50
Auditor Score	30

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 85 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

Audit Fail

Assessment Results

Important Notes:

- Several items were found.ı
- Function change It would be beneficial to set up a max ETH value to prevent the owner from setting it too high and drain wallets of users.ı
- some functions may allow re-entrancy, suggest implementing reentrancy guard. This is prone to reentrancy attacks, as the user can call this function and then call withdrawLP() before the transfer is completed.ı
- Additional details have been commented on code.

Auditor Score =30
Audit Fail

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided ‘as is, and Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.