# d3ploy

d3ploy

SECURITY ASSESSMENT

# Fluid

February 20th 2024

TABLE OF

# Contents

LEGAL

## Disclaimer

D3ploy audits 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 or project that contracts d3ploy to perform a security review. D3ploy does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

D3ploy 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. The report is provided only for the contract(s) mentioned in the report and does not include any other potential additions and/or contracts deployed by Owner. The report does not provide a review for contract(s), applications and/or operations, that are out of this report scope.

D3ploy’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

D3ploy represents 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. D3ploy’s position is that each company and individual are responsible for their own due diligence and continuous security. The security audit is not meant to replace functional testing done before a software release. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits and a public bug bounty program to ensure the security of the smart contracts.

D3PLOY

# Introduction

D3ploy is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

## Secure your project with d3ploy

We offer field-proven audits with in-depth reporting and a range of suggestions to improve and avoid contract vulnerabilities. Industry-leading comprehensive and transparent smart contract auditing on all public and private blockchains.

### Vunerability checking

A crucial manual inspection carried out to eliminate any code flaws and security loopholes. This is vital to avoid vulnerabilities and exposures incurring costly errors at a later stage.

### Contract verification

A thorough and comprehensive review in order to verify the safety of a smart contract and ensure it is ready for launch and built to protect the end-user

### Risk assessment

Analyse the architecture of the blockchain system to evaluate, assess and eliminate probable security breaches. This includes a full assessment of risk and a list of expert suggestions.

### In-depth reporting

A truly custom exhaustive report that is transparent and depicts details of any identified threats and vulnerabilities and classifies those by severity.

### Fast turnaround

We know that your time is valuable and therefore provide you with the fastest turnaround times in the industry to ensure that both your project and community are at ease.

### Best-of-class blockchain engineers

Our engineers combine both experience and knowledge stemming from a large pool of developers at our disposal. We work with some of the brightest minds that have audited countless smart contracts over the last 4 years.

PROJECT

## Introduction

Fluid is a fintech innovator offering advanced crypto trading software that integrates with platforms like Telegram and Discord.

With a focus on scalability and a strong development pipeline, we're set for rapid growth. Our edge comes from industry expertise, strategic partnerships, and a commitment to transparency. We're not just following the market, we're leading it.

Imagine executing a market order in under 15 seconds within Telegram, bypassing the tedious steps of logging in, connecting wallets, and manually inputting order details. With Fluid, you can swiftly trade top cryptocurrencies like BTC, ETH, and AVAX with up to 50x leverage, view real-time PnL stats, bridge assets, and even swap tokens seamlessly.

| | |
|---|---|
| *Project Name* | *Fluid* |
| *Contract Name* | *FLUID Token* |
| *Contract Address* | *0x4E47951508Fd4A4126F8ff9CF5E6Fa3b7cC8E073* |
| *Contract Chain* | *Mainnet* |
| *Contract Type* | *Smart Contract* |
| *Platform* | *EVM* |
| *Language* | *Solidity* |
| *Network* | *Ethereum (ERC20), Arbitrum* |
| *Codebase* | *Private GitHub Repository* |
| *Max Supply* | *10,000,000* |

INFO

## Social

https://fluid.trade/

https://twitter.com/FluidToken

https://t.me/FluidTradingPortal

https://fluidtrade.gitbook.io/docs/

https://medium.com/@fluidtrade

https://github.com/FluidTrade

support@fluid.trade

93

PASS

AUDIT

# Score

| | |
|---|---|
| ✦ Issues | 10 |
| ✦ Critical | 0 |
| ✦ Major | 0 |
| ✦ Medium | 0 |
| ✦ Minor | 6 |
| ✦ Informational | 4 |
| ✦ Discussion | 0 |

All issues are described in further detail on the following pages.

# AUDIT Scope

| RAW SOLIDITY FILES | LOCATION |
|---|---|
| V3AdaptedForCamelot/FluidOFT.sol | ✦ Solidity Files |
| V3AdaptedForCamelot/LiquidityManager.sol | ✦ Solidity Files |
| V3AdaptedForCamelot/RevShareStaking.sol | ✦ Solidity Files |

## REVIEW Methodology

### TECHNIQUES

This report has been prepared for Fluid to discover issues and vulnerabilities in the source code of the Fluid project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic, Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts producedby industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from major to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective in the comments below.

### TIMESTAMP

*Version v1.0*

*Date 2024/02/15*

*Descrption Layout project*

*Architecture / Manual review / Static & dynamic security testing*

*Summary*

*Version v1.1*

*Date 2024/02/20*

*Descrption Reaudit addressed vulnerabilities*

*Final Summary*

## KEY Finding

| TITLE | SEVERITY | STATUS |
|---|---|---|
| MISSING ZERO ADDRESS VALIDATION | ✦ Minor | Partially Fixed |
| APPROVING MAXIMUM VALUE | ✦ Minor | Acknowledged |
| LONG NUMBER LITERALS | ✦ Minor | Acknowledged |
| MISSING EVENTS | ✦ Minor | Fixed |
| OUTDATED COMPILER VERSION | ✦ Minor | Acknowledged |
| USE OWNABLE2STEP | ✦ Minor | Acknowledged |
| BOOLEAN EQUALITY | ✦ Informational | Fixed |
| MISSING UNDERSCORE IN NAMING VARIABLES | ✦ Informational | Acknowledged |
| UNUSED RECEIVE FALLBACK | ✦ Informational | Acknowledged |
| VARIABLES SHOULD BE IMMUTABLE | ✦ Informational | Fixed |

# IN - DEPTH Vulnerabilities

## DESCRIPTION

The contracts were found to be setting new addresses without proper validations for zero addresses. Address typeparameters should include a zero-address check otherwise contract functionality may become inaccessible or tokensburned forever. Depending on the logic of the contract, this could prove fatal and the users or the contracts could losetheir funds, or the ownership of the contract could be lost forever

## AFFECTED CODE

- FluidOFT.sol L397 - L399
- LiquidityManager.sol L305 - L307
- RevShareStaking.sol L315 - L317

**Issue** : MISSING ZERO ADDRESS VALIDATION

**Level** : Minor

**Remediation** : Add a zero address validation to all the functions where addresses are being set.

**Alleviation / Retest** : The bug has been partially fixed.

## IN - DEPTH Vulnerabilities

### DESCRIPTION

The function createThePool was detected to be using the maximum value for the approval amount. This is a malicious behavior and should be discouraged.

### AFFECTED CODE

- FluidOFT.sol L345; L355

**Issue** : APPROVING MAXIMUM VALUE

**Level** : Minor

**Remediation** : Modify the function call to approve only the required amount or use safeIncreaseAllowance and safeDecreaseAllowance.

**Alleviation / Retest** : The Fluid team has acknowledged the issue.

## IN - DEPTH Vulnerabilities

### DESCRIPTION

Solidity supports multiple rational and integer literals, including decimal fractions and scientific notations. The use of very large numbers with too many digits was detected in the code that could have been optimized using a different notation also supported by Solidity.
The value 100000 was detected on the below mentioned lines.

### AFFECTED CODE

- FluidOFT.sol L43; L367

**Issue** : LONG NUMBER LITERALS

**Level** : Minor

**Remediation** : Scientific notation in the form of 2e10 is also supported, where the mantissa can be fractional but the exponent has tobe an integer. The literal MeE is equivalent to M * 10**E . Examples include 2e10 , 2e10 , 2e-10 , 2.5e1 , as suggested in official solidity documentation *https://docs.soliditylang.org/en/latest/types.html#rationaland-integer-literals*

**Alleviation / Retest** : The Fluid team has acknowledged the issue and will not apply changes as it is not exploitable.

# IN - DEPTH Vulnerabilities

## DESCRIPTION

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contracts were found to be missing these events on the function setBaseBonusPercent which would make it difficult or impossible to track these transactions off-chain.

## AFFECTED CODE

- FluidOFT.sol L300 - L415
- LiquidityManager.sol L164 - L175
- RevShareStaking.sol L254 - L257; L301 - L348

**Issue** : MISSING EVENTS

**Level** : Minor

**Remediation** : Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

**Alleviation / Retest** : Fixed.

# IN - DEPTH Vulnerabilities

## DESCRIPTION

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.The following outdated versions were detected:

## AFFECTED CODE

- FluidOFT.sol L03
- LiquidityManager.sol L03
- RevShareStaking.sol L03

**Issue** : OUTDATED COMPILER VERSION

**Level** : Minor

**Remediation** : It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newersecurity checks. Consider using the solidity version v0.8.23 , which patches most solidity vulnerabilities.

**Alleviation / Retest** : The Fluid team has acknowledged the issue and will not apply changes as it is not exploitable.

# IN - DEPTH Vulnerabilities

## DESCRIPTION

Ownable2Step is safer than Ownable for smart contracts because the owner cannot accidentally transfer the ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.

## AFFECTED CODE

- FluidOFT.sol L39 - L477
- LiquidityManager.sol L16 - L176
- RevShareStaking.sol L29 - L364

**Issue** : USE OWNABLE2STEP

**Level** : Minor

**Remediation** : It is recommended to use either Ownable2Step or Ownable2StepUpgradeable depending on the smart contract.

**Alleviation / Retest** : The Fluid team has acknowledged the issue and will not apply changes as it is not exploitable.

## IN - DEPTH Vulnerabilities

DESCRIPTION

In Solidity, and many other languages, boolean constants can be used directly in conditionals like if and else statements.
The contract was found to be equating constants in conditionals which is unnecessary.

AFFECTED CODE

- RevShareStaking.sol L260; L335

**Issue** : BOOLEAN EQUALITY

**Level** : Informational

**Remediation** : It is recommended to directly use boolean constants. It is not required to equate them to true or false.

**Alleviation / Retest** : Fixed

# IN - DEPTH Vulnerabilities

## DESCRIPTION

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same

## AFFECTED CODE

- FluidOFT.sol L18 - L21; L43; L53; L62; L69; L417 - L443
- LiquidityManager.sol L164 - L166
- RevShareStaking.sol L66; L245 - L252; L278 - L299

**Issue** : MISSING UNDERSCORE IN NAMING VARIABLES

**Level** : Informational

**Remediation** : It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

**Alleviation / Retest** : The Fluid team has acknowledged the issue.

## *IN - DEPTH* Vulnerabilities

### DESCRIPTION

The contract was found to be defining an empty receive function.
It is not recommended to leave them empty unless there's a specific use case such as to receive Ether via an empty receive() function.

### VULNERABLE CODE

- FluidOFT.sol L103
- LiquidityManager.sol L162

**Issue** : UNUSED RECEIVE FALLBACK

**Level** : Informational

**Remediation** : It is recommended to go through the code to make sure these functions are properly implemented and are not missing any validations in the definition.

**Alleviation / Retest** : The Fluid team has acknowledged the issue and will not apply changes as it is not exploitable.

## IN - DEPTH Vulnerabilities

DESCRIPTION

Constants and Immutables should be used in their appropriate contexts.
constant should only be used for literal values written into the code. immutable variables should be used for expressions, or values calculated in, or passed into the constructor.

AFFECTED CODE

- FluidOFT.sol L53
- LiquidityManager.sol L19; L20
- RevShareStaking.sol L32; L63

Issue : VARIABLES SHOULD BE IMMUTABLE

**Level** : Informational

**Remediation** : It is recommended to use immutable instead of constant.

**Alleviation / Retest** : Fixed

## SOURCE Code

Raw Solidity Files

- FluidOFT.sol
- LiquidityManager.sol
- RevShareStaking.sol

# REPORT Appendix

## FINDING CATEGORIES

The assessment process will utilize a mixture of static analysis, dynamic analysis, in-depth manual review and/or other security techniques.

This report has been prepared for Fluid project using the above techniques to examine and discover vulnerabilities and safe coding practices in Fluid's smart contract including the libraries used by the contract that are not officially recognized.

A comprehensive static and dynamic analysis has been performed on the solidity code in order to find vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds.

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The testing methods find and flag issues related to gas optimizations that help in reducing the overall gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date.

## AUDIT SCORES

D3ploy Audit Score is not a live dynamic score. It is a fixed value determined at the time of the report issuance date.

D3ploy Audit Score is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports and scores are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts d3ploy to perform a security review.

d3ploy

WEBSITE d3ploy.co

@d3ploy_ TWITTER