d3ploy

d3ploy

SECURITY ASSESSMENT

Magpie Protocol

June 03rd 2023

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

Magpie protocol is a cross-chain liquidity aggregator that enables seamless cross-chain swaps with near-instant finality and cost efficiency on many of the top blockchains, all without the need to bridge any assets, making for an extremely fast, secure, easy, and gas efficient solution.

Magpie protocol incorporates a unique technical implementation that allows execution of cross-chain swaps without the need for the user to bridge assets from any of the top bridges. This saves time and cost by reducing the complexity and risks involved in using any of the bridging solutions to move assets across chains.

Project Name	*Magpie Protocol*
Contract Name	*FLY Token*
Contract Address	-
Contract Chain	*Not Yet Deployed on Mainnet*
Contract Type	*Smart Contract*
Platform	*EVM*
Language	*Solidity*
Network	*Ethereum (ERC20)*
Codebase	*Private GitHub Repository*
Total Token Supply	-

INFO

Social

https://www.magpiefi.xyz/

https://twitter.com/magpieprotocol

https://t.me/magpieprotocol

https://discord.gg/CwJuFeHp6f

https://medium.com/@Magpieprotocol

https://github.com/magpieprotocol/

contact@magpiefi.xyz

95

PASS

AUDIT

Score

Issues	9
Critical	0
Major	0
Medium	0
Minor	0
Informational	1
Discussion	8

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

AUDIT Scope

CODEBASE FILES	LOCATION
magpieprotocol/magpie-contracts/contracts/aggregator/	✦ Private Repository
magpieprotocol/magpie-contracts/contracts/bridge/	✦ Private Repository
magpieprotocol/magpie-contracts/contracts/data-transfer/	✦ Private Repository
magpieprotocol/magpie-contracts/contracts/interfaces/	✦ Private Repository
magpieprotocol/magpie-contracts/contracts/libraries/	✦ Private Repository
magpieprotocol/magpie-contracts/contracts/multicall/	✦ Private Repository
magpieprotocol/magpie-contracts/contracts/pauser/	✦ Private Repository
magpieprotocol/magpie-contracts/contracts/router/	✦ Private Repository

REVIEW Methodology

TECHNIQUES

This report has been prepared for Magpie Protocol to discover issues and vulnerabilities in the source code of the Magpie Protocol 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 produced by 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	*2023/06/03*
Descrption	*Layout project*
	Architecture / Manual review / Static & dynamic security testing
	Summary

KEY Finding

TITLE	SEVERITY	STATUS
SAME CONTRACT/LIBRARY NAMES	✦ Informational	Pending
ARRAY LENGTH CACHING	✦ Gas	Pending
CODE OPTIMIZATION BY USING MAX AND MIN	✦ Gas	Pending
CONSTANT STATE VARIABLES	✦ Gas	Pending
GAS OPTIMIZATION IN INCREMENTS	✦ Gas	Pending
INTERNAL FUNCTIONS NEVER USED	✦ Gas	Pending
MAKE PUBLIC LIBRARY FUNCTIONS INTERNAL	✦ Gas	Pending
UNUSED IMPORTS	✦ Gas	Pending
UNUSED NAMED RETURNS	✦ Gas	Pending

IN - DEPTH Vulnerabilities

DESCRIPTION

The libraries were found to be having same names in the same project. The library LibWormhole's name was reused in Bridge and Data Transfer. This may introduce errors or confusion during development and audit.

FILE LOCATION

- contracts/bridge/LibWormhole.sol L17
- contracts/data-transfer/LibWormhole.sol L17

Issue : SAME CONTRACT/LIBRARY NAMES

Level : Informational

Remediation : It is recommended to have different names for all the contracts and libraries in the same project.

Alleviation / Retest :

IN - DEPTH Vulnerabilities

DESCRIPTION

During each iteration of the loop, reading the length of the array uses more gas than is necessary. In the most favorable scenario, in which the length is read from a memory variable, storing the array length in the stack can save about 3 gas per iteration. In the least favorable scenario, in which external calls are made during each iteration, the amount of gas wasted can be significant.

FILE LOCATION

- /contracts/router/LibRouter.sol L157-L226
- /contracts/multicall/LibMulticall.sol L17-L23

Issue : ARRAY LENGTH CACHING

Level : Gas

Remediation : Consider storing the array length of the variable before the loop and use the stored length instead of fetching it in each iteration.

Alleviation / Retest :

IN - DEPTH Vulnerabilities

DESCRIPTION

The contract was using the expression 2**32 to calculate the maximum storage capacity of the data type. This both lacks readability and costs more gas.

FILE LOCATION

- /contracts/data-transfer/LibWormhole.sol L41
- /contracts/bridge/LibWormhole.sol L77

Issue : CODE OPTIMIZATION BY USING MAX AND MIN

Level : Gas

Remediation : 2**32 can be replaced by type(uint32).max; or type(uint32).min; which is more readable and will also save some gas.
However, keep in mind that type(uint32).max; is equal to 2**32 - 1

Alleviation / Retest :

IN - DEPTH Vulnerabilities

DESCRIPTION

The contract has defined state variables whose values are never modified throughout the contract.
The variables whose values never change should be marked as constant to save gas.

FILE LOCATION

- /contracts/pauser/facets/PauserFacet.sol L10
- /contracts/data-transfer/facets/DataTransferFacet.sol L13
- /contracts/bridge/facets/BridgeFacet.sol L13
- /contracts/router/facets/RouterManagerFacet.sol L10
- /contracts/multicall/facets/MulticallFacet.sol L10
- /contracts/aggregator/facets/AggregatorFacet.sol L14
- /contracts/router/facets/Router1Facet.sol L23

Issue : CONSTANT STATE VARIABLES

Level : Gas

Remediation : Make sure that the values stored in the variables flagged above do not change throughout the contract. If this is the case, then consider setting these variables as constant.

Alleviation / Retest :

IN - DEPTH Vulnerabilities

DESCRIPTION

++i costs less gas compared to i++ or i += 1 for unsigned integers. In i++, the compiler has to create a temporary variable to store the initial value. This is not the case with ++i in which the value is directly incremented and returned, thus, making it a cheaper alternative.

FILE LOCATION

- /contracts/multicall/LibMulticall.sol L17

Issue : GAS OPTIMIZATION IN INCREMENTS

Level : Gas

Remediation : Consider changing the post-increments (i++) to pre-increments (++i) as long as the value is not used in any calculations or inside returns. Make sure that the logic of the code is not changed.

Alleviation / Retest :

IN - DEPTH Vulnerabilities

DESCRIPTION

The contract declared internal functions but was not using them in any of the functions or contracts.
Since internal functions can only be called from inside the contracts, it makes no sense to have them if they are not used. This uses up gas and causes issues for auditors when understanding the contract logic.

FILE LOCATION

- /contracts/libraries/LibAsset.sol L57-63

Issue : INTERNAL FUNCTIONS NEVER USED

Level : Gas

Remediation : Having dead code in the contracts uses up unnecessary gas and increases the complexity of the overall smart contract.
It is recommended to remove the internal functions from the contracts if they are never used.

Alleviation / Retest :

IN - DEPTH Vulnerabilities

DESCRIPTION

The contracts have a library LibRouter which has defined its function as public. This can be optimized by changing the function visibility. Changing the visibility from public will remove the compiler-introduced checks for msg.value and decrease the contract's method ID table size

FILE LOCATION

- /contracts/router/LibRouter.sol L46-117

Issue : MAKE PUBLIC LIBRARY FUNCTIONS INTERNAL

Level : Gas

Remediation : The public functions can be changed to private/internal to save some gas.

Alleviation / Retest :

IN - DEPTH Vulnerabilities

DESCRIPTION

Solidity is a Gas-constrained language. Having unused code or import statements incurs extra gas usage when deploying the contract.
The contract was found to be importing the files listed below which is not used anywhere in the code..
openzeppelin-solidity/contracts/token/ERC20/IERC20.sol
../interfaces/curve/IAddressProvider.sol
hardhat-deploy/solc_0.8/diamond/libraries/LibDiamond.sol
../libraries/LibBytes.sol

FILE LOCATION

- /contracts/interfaces/balancer-v2/IVault.sol L04
- /contracts/router/LibRouter.sol L05
- /contracts/router/LibRouterManager.sol L04
- /contracts/bridge/LibCommon.sol L04

Issue : UNUSED IMPORTS

Level : Gas

Remediation : It is recommended to remove the import statement if it’s not supposed to be used.

Alleviation / Retest :

IN - DEPTH Vulnerabilities

DESCRIPTION

Using both named returns and a return statement isn't necessary. Removing unused named return variables can reduce gas usage and improve code clarity.

FILE LOCATION

- /contracts/router/LibTrident.sol L43-55

Issue : UNUSED NAMED RETURNS

Level : Gas

Remediation : To save gas and improve code quality, consider using either the named returns or a return statement.

Alleviation / Retest :

SOURCE Code

Private GitHub Repository

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 Magpie Protocol project using the above techniques to examine and discover vulnerabilities and safe coding practices in Magpie Protocol'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