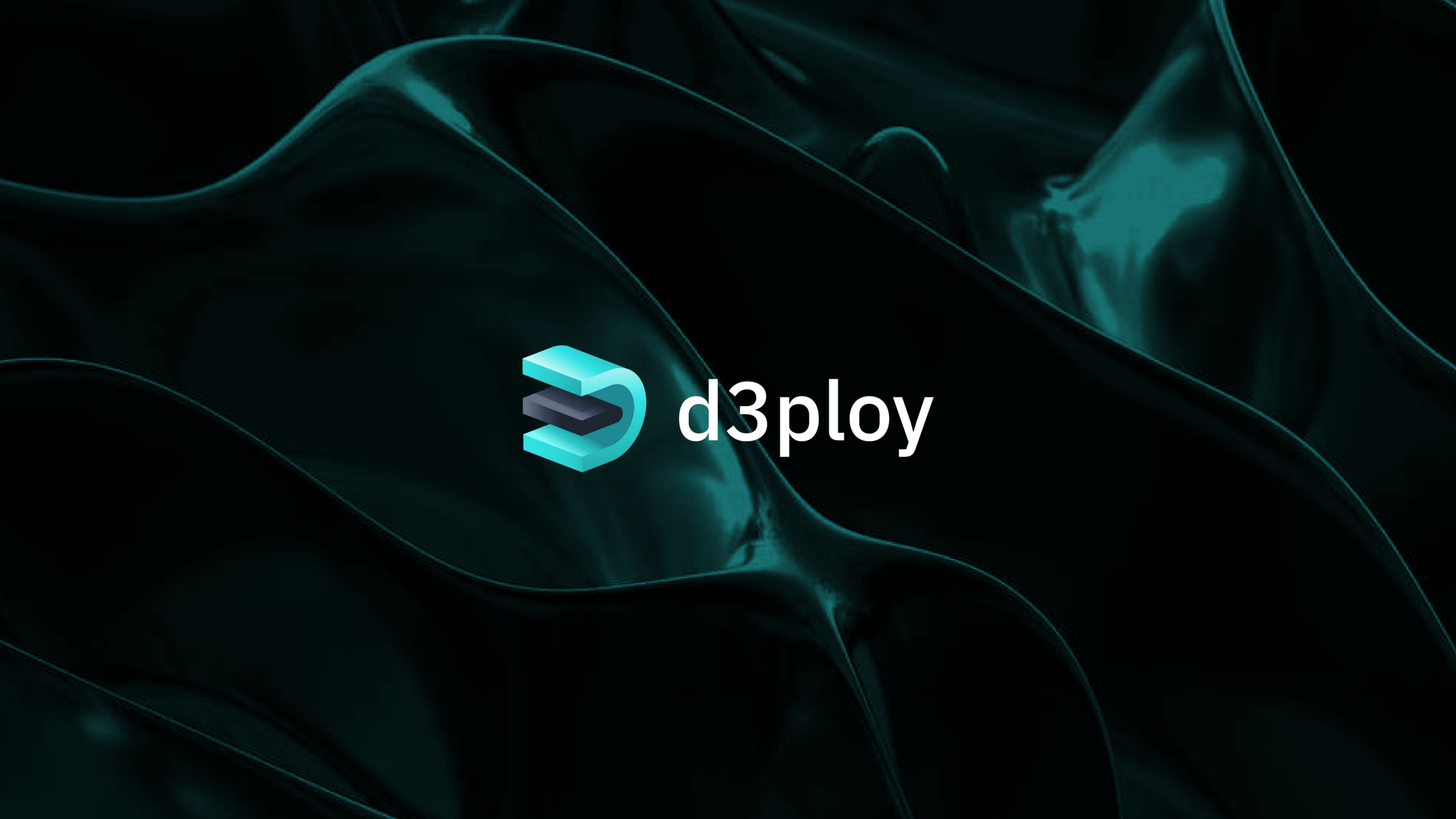
d3ploy

d3ploy

SECURITY ASSESSMENT

Tyche Finance

April 24th 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

Tyche Finance: ecosystem-centric multichain DEX

Tyche Finance is the ultimate tool for portfolio management with a dex. We help users assemble a portfolio or take a ready one from their favorite influencers. And then buy these tokens with one click of a button. And it is all decentralized

Project Name Tyche Finance
Contract Name TCH Token
Contract Address -
Contract Chain Not Yet Deployed on Mainnet
Contract Type Smart Contract
Platform EVM
Language Solidity
Network Arbitrum (ERC20), Polygon (MATIC)
Codebase Arbitrum One Explorer
Total Token Supply 100,000,000

INFO

Social

https://tyche.finance/

https://twitter.com/tyche_finance

-

-

https://medium.com/@tychefinance

-

-

95

PASS

AUDIT

Score

Issues	9
Critical	0
Major	0
Medium	1
Low	2
Informational	6
Discussion	0

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

AUDIT Scope

CODEBASE FILES	LOCATION
address/0x19Dc1EF9cc8122eCed725d774a8fa042aF7e1161#code	✦ Arbitrum One Explorer
address/0x76e26d419f8bb3857a63a89bb3b64c7dd79283ee#code	✦ Arbitrum One Explorer

REVIEW Methodology

TECHNIQUES

This report has been prepared for Tyche Finance to discover issues and vulnerabilities in the source code of the Tyche Finance 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	*2023/04/19*
Descrption	*Layout project*
	Architecture / Manual review / Static & dynamic security testing
	Summary
Version	*v1.1*
Date	*2023/04/24*
Descrption	*Reaudit addressed issues*
	Final Summary

KEY Finding

TITLE	SEVERITY	STATUS
INCORRECT INPUT VALIDATION IN SETSALEDEADLINE FUNCTION	✦ Medium	Fixed
MISSING ZERO ADDRESS VALIDATION	✦ Low	Fixed
CHEAPER INEQUALITIES IN IF()	✦ Gas	Fixed
CHEAPER INEQUALITIES IN REQUIRE()	✦ Gas	Fixed
GAS OPTIMIZATION FOR STATE VARIABLES	✦ Gas	Fixed
OUTDATED COMPILER VERSION	✦ Low	Fixed
PUBLIC CONSTANTS CAN BE PRIVATE	✦ Gas	Fixed
USE OF SAFEMATH LIBRARY	✦ Gas	Fixed
UNNECESSARY DEFAULT VALUE INITIALIZATION	✦ Gas	Fixed

IN - DEPTH Vulnerabilities

DESCRIPTION

The contract is using a function setSaleDeadline() to set the deadline for the sale. Inside the function it is validating if the deadline is greater than the current block's timestamp. However, the parameter used in the validation is not the one that is getting passed as the function argument. It is the one defined as a storage variable (_saleDeadline). This could allow the owner to set a sale deadline in the past which will also make the contract unusable.

LOCATION

- contracts/TCHSale.sol L61

Issue : INCORRECT INPUT VALIDATION IN SETSALEDEADLINE FUNCTION

Level : Medium

Remediation : Use the correct value saleDeadline during input validation.

Alleviation / Retest : A boundry check has been added and correct variable is used.

Vulnerabilities

DESCRIPTION

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

LOCATION

- contracts/TCHSale.sol L45

Issue : MISSING ZERO ADDRESS VALIDATION

Level : Low

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

Alleviation / Retest : Fixed, as per suggested remediation.

Vulnerabilities

DESCRIPTION

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the if statements, non-strict inequalities (>=, <=) are usually cheaper than the strict equalities (>, <).

LOCATION

- contracts/TCHSale.sol L79, L192, L198, L223, L241

Issue : CHEAPER INEQUALITIES IN IF()

Level : Gas

Remediation : It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save ~3 gas as long as the logic of the code is not affected.

Alleviation / Retest : Fixed, as per suggested remediation.

Vulnerabilities

DESCRIPTION

The contract was found to be performing comparisons using inequalities inside the require statement.
When inside the require statements, non-strict inequalities (>=, <=) are usually costlier than strict equalities (>, <).

LOCATION

- contracts/TCHSale.sol L96, L100, L127, L161, L181, L280

Issue : CHEAPER INEQUALITIES IN REQUIRE()

Level : Gas

Remediation : It is recommended to go through the code logic, and, if possible, modify the non-strict inequalities with the strict ones to save ~3 gas as long as the logic of the code is not affected.

Alleviation / Retest : Fixed, as per suggested remediation.

Vulnerabilities

DESCRIPTION

Plus equals (+=) costs more gas than addition operator. The same thing happens with minus equals (-=). Therefore, x +=y costs more gas than x = x + y.

LOCATION

- contracts/TCHSale.sol L138

Issue : GAS OPTIMIZATION FOR STATE VARIABLES

Level : Gas

Remediation : Consider

- addition operator over plus equals
- subtraction operator over minus equals
- division operator over divide equals
- multiplication operator over multiply equals

Alleviation / Retest : Fixed, as per suggested remediation.

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:
contracts/TCHSale.sol - =0.8.7
contracts/TycheFinanceToken.sol - =0.8.7

LOCATION

- contracts/TCHSale.sol L02
- contracts/TycheFinanceToken.sol L02

Issue : OUTDATED COMPILER VERSION

Level : Low

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 newer security checks. Consider using the solidity version 0.8.18, which patches most solidity vulnerabilities.

Alleviation / Retest : Fixed, as per suggested remediation.

Vulnerabilities

DESCRIPTION

Public constant variables cost more gas because the EVM automatically creates getter functions for them and adds entries to the method ID table. The values can be read from the source code instead.

LOCATION

- contracts/TCHSale.sol L14-L16; L18-L20; L22-L24

The following variable is affected: TWENTY
HUNDRED
MIN_MONTH_FOR_FULL_REWARD
ONE_MONTH_IN_SEC
MAX_DIFF_NEXT_DEADLINE
BASE
SOFT_CAP
HARD_CAP
TCH_PRICE

Issue : PUBLIC CONSTANTS CAN BE PRIVATE

Level : Gas

Remediation : If reading the values for the constants are not necessary, consider changing the public visibility to private.

Alleviation / Retest : Fixed, as per suggested remediation.

Vulnerabilities

DESCRIPTION

SafeMath library is found to be used in the contract. This increases gas consumption than traditional methods and validations if done manually.
Also, Solidity 0.8.0 includes checked arithmetic operations by default, and this renders SafeMath unnecessary.

LOCATION

- contracts/TCHSale.sol L10

Issue : USE OF SAFEMATH LIBRARY

Level : Gas

Remediation : We do not recommend using SafeMath library for all arithmetic operations. It is good practice to use explicit checks where it is really needed and to avoid extra checks where overflow/underflow is impossible.
The compiler should be upgraded to Solidity version 0.8.0+ which automatically checks for overflows and underflows.

Alleviation / Retest : Fixed, as per suggested remediation.

Vulnerabilities

DESCRIPTION

The contract was found to be initializing the value of uint256 to it's default value, i.e., 0. This is redundant and not required.

The contract was found to be initializing the value of bool to it's default value, i.e., false. This is redundant and not required.

LOCATION

- contracts/TCHSale.sol L28-L31

Issue : UNNECESSARY DEFAULT VALUE INITIALIZATION

Level : Gas

Remediation : It's not recommended to initialize the data types to their default values unless there's a use-case because it's unnecessary and costs around ~3 gas.

Alleviation / Retest : Fixed, as per suggested remediation.

SOURCE Code

Arbitrum One Explorer

address/0x19Dc1EF9cc8122eCed725d774a8fa042aF7e1161#code
address/0x76e26d419f8bb3857a63a89bb3b64c7dd79283ee#code

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 Tyche Finance project using the above techniques to examine and discover vulnerabilities and safe coding practices in Tyche Finance 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