d3ploy

d3ploy

SECURITY ASSESSMENT

NF3x

May 27th 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

NF3x is an NFT/FT Swaps Market

A multi-asset swaps and options platform to barter, reserve & exchange your favourite NFT collections, entirely peer-to-peer.

Project Name	*NF3 Exchange*
Contract Name	*-*
Contract Address	*-*
Contract Chain	*-*
Contract Type	*Smart Contract*
Platform	*EVM*
Language	*Solidity*
Network	*Ethereum (ERC20), Polygon (MATIC)*
Codebase	*Private GitHub Repository*
Total Token Supply	*-*

INFO

Social

https://nf3.exchange/

https://twitter.com/nf3exchange

-

https://discord.gg/XYKrCHMufu

-

https://github.com/NF3Labs/

hello@nf3x.io

96

PASS

AUDIT

Score

Issues	6
Critical	0
Major	0
Medium	0
Low	4
Informational	2
Discussion	0

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

AUDIT Scope

CODEBASE FILES	LOCATION
NF3Labs/contracts-V2/blob/main/contracts/Vault.sol	✦ Private GitHub Repository
NF3Labs/contracts-V2/blob/main/contracts/Swap.sol	✦ Private GitHub Repository
NF3Labs/contracts-V2/blob/main/contracts/Reserve.sol	✦ Private GitHub Repository
NF3Labs/contracts-V2/blob/main/contracts/NF3Market.sol	✦ Private GitHub Repository

REVIEW Methodology

TECHNIQUES

This report has been prepared for NF3 Exchange to discover issues and vulnerabilities in the source code of the NF3 Exchange 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	*2023/05/18*
Descrption	*Layout project*
	Architecture / Manual review / Static & dynamic security testing
	Summary
Version	*v1.1*
Date	*2023/05/27*
Descrption	*Reaudit addressed issues*
	Final Summary

KEY Finding

TITLE	SEVERITY	STATUS
MISSING ZERO ADDRESS VALIDATION	✦ Low	Fixed
NFT REENTRANCY	✦ Low	Acknowledged
BOOLEAN EQUALITY	✦ Informational	Fixed
CHEAPER INEQUALITIES IN IF()	✦ Gas	Fixed
USE OF FLOATING PRAGMA	✦ Low	Fixed
OUTDATED COMPILER VERSION	✦ Low	Fixed

IN - DEPTH 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

- NF3Market.sol L53
- Vault.sol L75, L218

Issue : MISSING ZERO ADDRESS VALIDATION

Level : Low

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

Alleviation / Retest : Zero address validation has been added to the address type parameters.

IN - DEPTH Vulnerabilities

DESCRIPTION

In a Re-entrancy 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, especially in cases where the function updates state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.
The function transferAssets() is making a safeTransferFrom() which in turn calls _checkOnERC721Received() that makes a function call onERC721Received() to the receiver's address which can re-enter back into the contract causing unexpected behavior, token loss or malicious state update.

LOCATION

- Vault.sol L87-91, L93-99

Issue : NFT REENTRANCY

Level : Low

Remediation : It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.
https://docs.openzeppelin.com/contracts/4.x/api/security#ReentrancyGuard

Alleviation / Retest : NF3x team acknowledged and commented that they don't think the issue is applicable to their vaults as only certain functions are allowed to create trx.

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.

LOCATION

- Vault.sol L141, L275, L522

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, as per suggested remediation.

IN - DEPTH 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

- NF3Market.sol L142, L179, L241, L282, L318, L380, L415, L549
- Swap.sol L497
- Reserve.sol L540-544, L774, L813

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.

IN - DEPTH Vulnerabilities

DESCRIPTION

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
['/Swap.sol'] - ^0.8.9
['/NF3Market.sol'] - ^0.8.9
['/Vault.sol'] - ^0.8.9
['/Reserve.sol'] - ^0.8.9

LOCATION

- NF3Market.sol L02
- Swap.sol L02
- Reserve.sol L02
- Vault.sol L02

Issue : USE OF FLOATING PRAGMA

Level : Gas

Remediation : It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of ^0.8.9 use pragma solidity 0.8.18, which is a stable and recommended version right now.

Alleviation / Retest : Fixed, as per suggested remediation.

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:
['/Swap.sol'] - ^0.8.9
['/NF3Market.sol'] - ^0.8.9
['/Vault.sol'] - ^0.8.9
['/Reserve.sol'] - ^0.8.9

LOCATION

- NF3Market.sol L02
- Swap.sol L02
- Reserve.sol L02
- Vault.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 : Compiler version has been updated.

SOURCE Code

Private GitHub Repository

https://github.com/NF3Labs/contracts-V2/tree/main/contracts

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 NF3 Exchange project using the above techniques to examine and discover vulnerabilities and safe coding practices in NF3 Exchange 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