DetectBox
Powered by UNSNARL

#DECENTRALISED PROACTIVE AUDIT WAR-ROOMS

Token Contract Security Assessment

NFTUBE

AUGUST 29TH, 2023

TABLE OF CONTENTS

SUMMARY

This report has been prepared for NFTube to discover issues and vulnerabilities in the source code of the token project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing 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 critical 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:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live

CERTIFICATION

This is to Certify that the **NFTube Security Report** was prepared using gold standard web3 security standards with all standard and advanced checks deployed.

The **Main Auditor** Involved in the Audit was -
0xnirlin (@0xnirlin)

The Auditor mentioned hereby was chosen after self due-dilligence from the project's end.

All reported issues were either acknowledged or fixed by the project.

DetectBox's audit mitigation & residual vulnerability check did not find any major vulnerability to report within the Scope and requirements of the audit.

From
Team DetectBox

NFTube
Security Assessment Report

ABOUT DETECTBOX

DetectBox by **UNSNARL** brings to you world's first Decentralised Audit War-Rooms. An intense collaborative Environment where chosen auditors/audit team, project's decision makers, Projects developers and Detect-Wardens join hands to conduct rigorous security audits with absolute transparency followed by a **Proof of Audit (POA).**

We at DetectBox not only find bugs in smart-contracts but also take a deep look into the project's **tokenomics**, **white-paper**, **business logic**, **functional flows** and overall development health.

At DetectBox, you get to choose from a pool of Independent Security researchers which is curated by strong due-diligence. List your project requirements, verify the auditor's past work from their profiles and choose the auditor that rightly fits your project's needs and your budget.

Our auditors have successfully completed **200+ audits**, found over **2200+ vulnerabilities** and secured over **$102M+ worth of TVL.**

To know more about us visit - detectbox.io
To see our docs - https://unsnarl.gitbook.io/detectbox/

Yours Securely
UNSNARL

OVERVIEW

Project Summary

NFTube is a groundbreaking streaming platform that combines the power of NFTs (non-fungible tokens) with video content. It aims to revolutionize the way content creators share their work and monetize their creations.

Audit Summary

A time-boxed independent security assessment of the NFTUBE contract was done by 0xnirlin(@0xnirlin) and Team DetectBox with a focus on the security aspects of the application's implementation.
We performed the security assessment based on the agreed scope, following our approach and methodology. Based on our scope and our performed activities, our security assessment revealed 1 Critical and 3 Low severity issues. Additionally, 1 Gas suggestion was also made which, if resolved appropriately, may improve the quality of the Project's Smart contract.

Audit Timeline: 22nd August'23 - 25th August'23

Docs:
https://nftube.gitbook.io/nftube/

Website:
https://nftube.cam

OVERVIEW

Audit Scope

The code under review is composed of a single smart contract and interfaces written in the Solidity programming language and includes 31 nSLOC- normalized source lines of code (only source-code lines; no comments, no blank lines).

Auditors Involved :

Main Auditor -

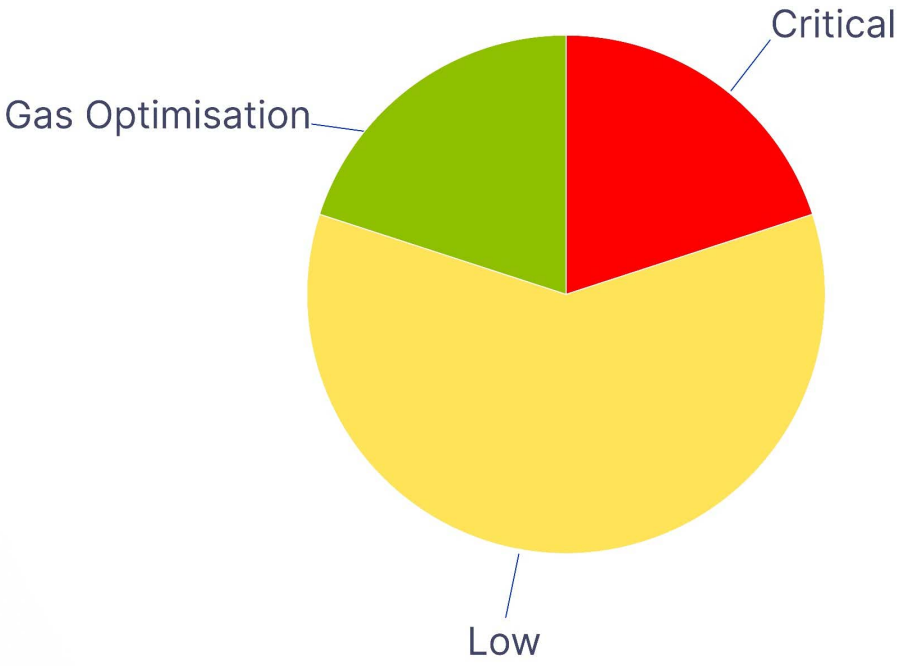
0xnirlin - https://app.detectbox.io/profile/0xnirlin

FINDINGS

Detailed Summary of Findings

Sl. No.	Name	Severity
C-01	Contract is not compliant with EIP-20 and hence completely unusable for any ERC20 use case	Critical
L-01	Missing Natspec	Low
L-02	New Solidity versions support named mappings, use that for more readibility of code	Low
L-03	Avoid Using Floating Pragma	Low
G-01	Constructors can be marked payable	Gas Optimization

FINDINGS



Static Analysis

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Manual Review

Critical Severity Issues

C-01. Contract is not compliant with EIP-20 and hence completely unusable for any ERC20 use case

FINDINGS

The token implemented is not compliant with the eip standard and miss the important functions that are necessary for the working of the token. All the tokens must comply with some standard eip, in this case EIP 20, which states that it must have following functions and events :

```
totalSupply()

balanceOf(account)

transfer(recipient, amount)

allowance(owner, spender)

approve(spender, amount)

transferFrom(sender, recipient, amount)
```

```
Transfer(from, to, value)

Approval(owner, spender, value)
```

Failing to comply with the eip20 will have the following impacts:

1. Token cannot be listed on any major CEX or DEX as they all expect existence of those functions and require the allowance specifically.
2. Not detectable by any wallet as no standard being followed.
3. Cannot be used for DAO purposes.
4. Cannot be used for any use case as not detectable by wallets, so it is essentially useless.

FINDINGS

Code Snippets:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract NFTube {
string public name = "NFTube";
string public symbol = "NFTT";
uint8 public decimals = 18;
uint256 public totalSupply = 100000000 * 10**uint256(decimals); // Total Supply: 100,00
0,000 NFTT (100 million)
mapping(address => uint256) public balanceOf;

event Transfer(address indexed from, address indexed to, uint256 value);

constructor() {
    balanceOf[msg.sender] = totalSupply;
}

function transfer(address to, uint256 value) external returns (bool) {
    require(balanceOf[msg.sender] >= value, "Insufficient balance");

    balanceOf[msg.sender] -= value;
    balanceOf[to] += value;

    emit Transfer(msg.sender, to, value);
    return true;
}
}
```

Recommended Mitigation Steps:

Use the openzeppelin ERC20 contracts by inheriting for them, they have all the necessary functions and are security tested.

FINDINGS

Low Severity Issues

L-01. Missing Natspec

Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).
There are no natspec functions in the whole contract for the functions, consider adding them to better explain the functions and make them more readable and audit-able.

Code Snippets:

```
pragma solidity ^0.8.0;

contract NFTube {
string public name = "NFTube";
string public symbol = "NFTT";
uint8 public decimals = 18;
uint256 public totalSupply = 100000000 * 10**uint256(decimals); // Total Supply: 100,00
0,000 NFTT (100 million)
mapping(address => uint256) public balanceOf;

event Transfer(address indexed from, address indexed to, uint256 value);

constructor() {
    balanceOf[msg.sender] = totalSupply;
}

function transfer(address to, uint256 value) external returns (bool) {
    require(balanceOf[msg.sender] >= value, "Insufficient balance");

    balanceOf[msg.sender] -= value;
    balanceOf[to] += value;

    emit Transfer(msg.sender, to, value);
    return true;
}
}
```

FINDINGS

Recommended Mitigation Steps:

Add the natspec comments in proper format on each function and constructor.

L-02. New Solidity versions support named mappings, use that for more readibility of code.

Newer versions of solidity like 0.8.20 allow the named mappings that make the code more readable.
Mapping are generally hard to read, specifically the nested ones. Adding names for them make them more readable and audit-able.

Code Snippets:

The following lines can be modified :

```
mapping(address => uint256) public balanceOf;
```

as

```
mapping(address user => uint256 balance) public balanceOf;
```

Recommended Mitigation Steps:

Use named mapping instead of unnamed mapping for more readability and auditibility of the codebase.

FINDINGS

L-03. Avoid Using Floating Pragma

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma.
An outdated pragma version might introduce bugs that affect the contract system negatively or recently released pragma versions may have unknown security vulnerabilities.

Code Snippets:

```
pragma solidity ^0.8.0;
```

Recommended Mitigation Steps:

Consider locking the pragma.

Gas Optimisation Issues

G-01. Constructors can be marked payable

You can cut out 10 opcodes in the creation-time EVM bytecode if you declare a constructor. Consider making the constructor payable to save gas.

FINDINGS

Code Snippets:

```
constructor() {
        balanceOf[msg.sender] = totalSupply;
    }
```

Recommended Mitigation Steps:

Make the constructor payable

POST AUDIT CONCLUSION

Fixing the Findings

Sl. No.	Name	Status
C-01	Contract is not compliant with EIP-20 and hence completely unusable for any ERC20 use case	Fixed
L-01	Missing Natspec	Fixed
L-02	New Solidity versions support named mappings, use that for more readibility of code	Fixed
L-03	Avoid Using Floating Pragma	Fixed
G-01	Constructors can be marked payable	Fixed

APPENDIX

Finding Categories

Finding Categories

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

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM 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 how block.timestamp works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

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

DISCLAIMER

DetectBox (by UNSNARL) has conducted an independent security review to verify the integrity of and highlight any vulnerabilities, bugs or errors, intentional or unintentional, that may be present in the codebase that were provided for the scope of this Engagement.

Blockchain technology and the cryptographic assets it supports are nascent technologies. This makes them extremely volatile assets. Any assessment report obtained on such volatile and nascent assets may include unpredictable results which may lead to positive or negative outcomes.

In some cases, services provided may be reliant on a variety of third parties. This security review does not constitute endorsement, agreement or acceptance for the Project and technology that was reviewed. Users relying on this security review should not consider this as having any merit for financial advice or technological due diligence in any shape, form or nature.

The veracity and accuracy of the findings presented in this report relate solely to the proficiency, competence, aptitude and discretion of our auditors. UNSNARL and its employees make no guarantees, nor assurance that the contracts are free of exploits, bugs, vulnerabilities, deprecation of technologies or any system / economical / mathematical malfunction.

DISCLAIMER

All the information/opinions/suggestions provided in this report does not constitute financial or investment advice, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report.

Information in this report is provided 'as is'. UNSNARL is under no covenant to the completeness, accuracy or solidity of the contracts reviewed. UNSNARL's goal is to help reduce the attack vectors/surface and the high level of variance associated with utilizing new and consistently changing technologies.

UNSNARL in no way claims any guarantee, warranty or assurance of security or functionality of the technology that was in scope for this security review.

In no event will UNSNARL, its partners, employees, agents or any parties related to the design/creation of this security review be ever liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this security review.

Cryptocurrencies and all other technologies directly or indirectly related to cryptocurrencies are not standardized, highly prone to malfunction and extremely speculative by nature. No due diligence and/or safeguards may be insufficient and users should exercise maximum caution when participating and/or investing in this nascent industry.

DISCLAIMER

The preparation of this security review has made all reasonable attempts to provide clear and actionable recommendations to the Project team (the “client”) with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts in scope for this engagement.

It is the sole responsibility of the Project team to provide adequate levels of test and perform the necessary checks to ensure that the contracts are functioning as intended, and more specifically to ensure that the functions contained within the contracts in scope have the desired intended effects, functionalities and outcomes, as documented by the Project team.

All services, the security reports, discussions, work product, attack vectors description or any other materials, products or results of this security review engagement is provided "as is" and "as available" and with all faults, uncertainty and defects without warranty or guarantee of any kind.

UNSNARL will assume no liability or responsibility for delays, errors, mistakes, or any inaccuracies of content, suggestions, materials or for any loss, delay, damage of any kind which arose as a result of this engagement/security review.

UNSNARL will assume no liability or responsibility for any personal injury, property damage, of any kind whatsoever that resulted in this engagement and the customer having access to or use of the products, engineers, services, security report, or any other other materials.

For avoidance of doubt, this report, its content, access, and/or usage thereof, including any associated services or materials, shall not be considered or relied upon as any form of financial, investment, tax, legal, regulatory, or any other type of advice.

GLORY OF AUDITORS AT DETECTBOX

DetectBox has created a pool of best auditors across the globe with significant experience in the web3 security industry auditing multiple protocols across multiple chains. Making audits better through **Proof of Talent**.

217+
Projects Audited

$100M+
Amount Secured

2200+
Vulnerabilities Detected

Follow Our Journey on

@unsnarl_secure

UNSNARL

founders@unsnarl.io

www.detectbox.io