hexens × Fungify

FEB.24

SECURITY REVIEW REPORT FOR FUNGIFY

CONTENTS

ABOUT HEXENS

Hexens is a cybersecurity company that strives to elevate the standards of security in Web 3.0, create a safer environment for users, and ensure mass Web 3.0 adoption.

Hexens has multiple top-notch auditing teams specialized in different fields of information security, showing extreme performance in the most challenging and technically complex tasks, including but not limited to: Infrastructure Audits, Zero Knowledge Proofs / Novel Cryptography, DeFi and NFTs. Hexens not only uses widely known methodologies and flows, but focuses on discovering and introducing new ones on a day-to-day basis.

In 2022, our team announced the closure of a $4.2 million seed round led by IOSG Ventures, the leading Web 3.0 venture capital. Other investors include Delta Blockchain Fund, Chapter One, Hash Capital, ImToken Ventures, Tenzor Capital, and angels from Polygon and other blockchain projects.

Since Hexens was founded in 2021, it has had an impressive track record and recognition in the industry: Mudit Gupta - CISO of Polygon Technology - the biggest EVM Ecosystem, joined the company advisory board after completing just a single cooperation iteration. Polygon Technology, 1inch, Lido, Hats Finance, Quickswap, Layerswap, 4K, RociFi, as well as dozens of DeFi protocols and bridges, have already become our customers and taken proactive measures towards protecting their assets.

EXECUTIVE SUMMARY

OVERVIEW

This audit covered a single commit for Fungify, a new lending protocol that builds on Compound to support lending/borrow of NFTs and introduces a special interest market token that is linked to NFT markets. This commit introduced a new type of token, namely CERC721NoBorrow, which is very similar to CERC721, except that deposited NFTs stay linked to the original owner and they can only be used as collateral and not borrowed.

Our security assessment was a review of the smart contracts changes in this commit, spanning a total of 2 days

During our audit, we did not identify any major vulnerabilities. We did identify 2 optimisations.

Finally, all of our reported issues were fixed by the development team and consequently validated by us.

We can confidently say that the overall security and code quality have increased after completion of our audit.

SCOPE

The analyzed resources are located on:

https://github.com/fungify-dao/taki-contracts/commit/197e76662fb13e977fe8ffd92782fb8b4fec7386

The issues described in this report were fixed in the following commit:

https://github.com/fungify-dao/taki-contracts/commit/e545376cf95c4844aaa4b00cf6f7a7336b8d188f

AUDITING DETAILS

STARTED

12.02.2024

DELIVERED

13.02.2024

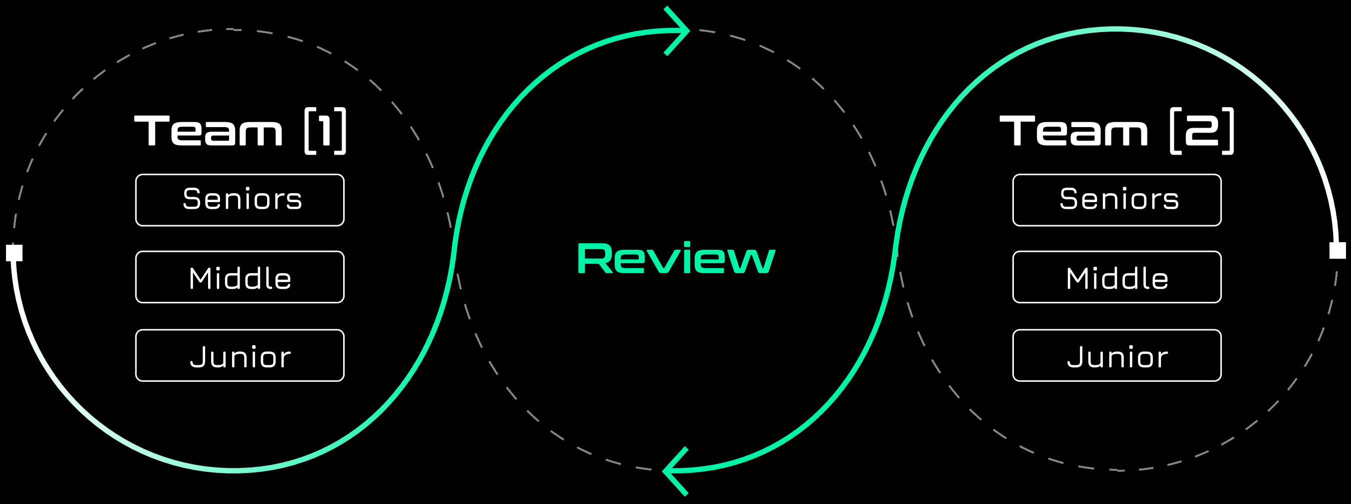
Review Led by

KASPER ZWIJSEN

Head of Smart Contract Audits | Hexens

HEXENS METHODOLOGY

Hexens methodology involves 2 teams, including multiple auditors of different seniority, with at least 5 security engineers. This unique cross-checking mechanism helps us provide the best quality in the market.



SEVERITY STRUCTURE

The vulnerability severity is calculated based on two components

- Impact of the vulnerability
- Probability of the vulnerability

Impact	Probability			
	rare	unlikely	likely	very likely
Low/Info	Low/Info	Low/Info	Medium	Medium
Medium	Low/Info	Medium	Medium	High
High	Medium	Medium	High	Critical
Critical	Medium	High	Critical	Critical

SEVERITY CHARACTERISTICS

Smart contract vulnerabilities can range in severity and impact, and it's important to understand their level of severity in order to prioritize their resolution. Here are the different types of severity levels of smart contract vulnerabilities:

Critical

Vulnerabilities with this level of severity can result in significant financial losses or reputational damage. They often allow an attacker to gain complete control of a contract, directly steal or freeze funds from the contract or users, or permanently block the functionality of a protocol. Examples include infinite mints and governance manipulation.

High

Vulnerabilities with this level of severity can result in some financial losses or reputational damage. They often allow an attacker to directly steal yield from the contract or users, or temporarily freeze funds. Examples include inadequate access control integer overflow/underflow, or logic bugs.

Medium

Vulnerabilities with this level of severity can result in some damage to the protocol or users, without profit for the attacker. They often allow an attacker to exploit a contract to cause harm, but the impact may be limited, such as temporarily blocking the functionality of the protocol. Examples include uninitialized storage pointers and failure to check external calls.

Low

Vulnerabilities with this level of severity may not result in financial losses or significant harm. They may, however, impact the usability or reliability of a contract. Examples include slippage and front-running, or minor logic bugs.

Informational

Vulnerabilities with this level of severity are regarding gas optimizations and code style. They often involve issues with documentation, incorrect usage of EIP standards, best practices for saving gas, or the overall design of a contract. Examples include not conforming to ERC20, or disagreement between documentation and code.

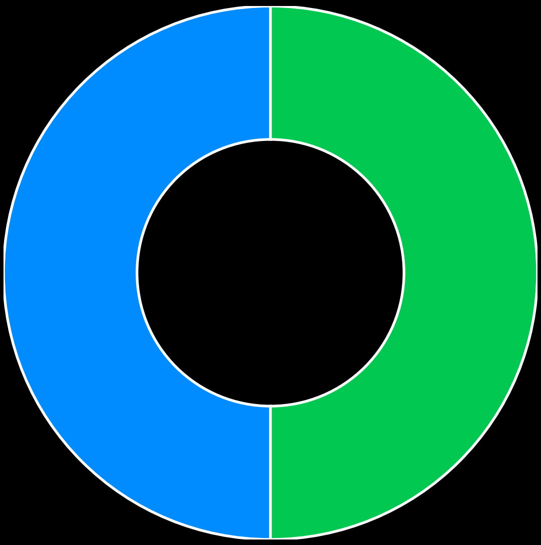
ISSUE SYMBOLIC CODES

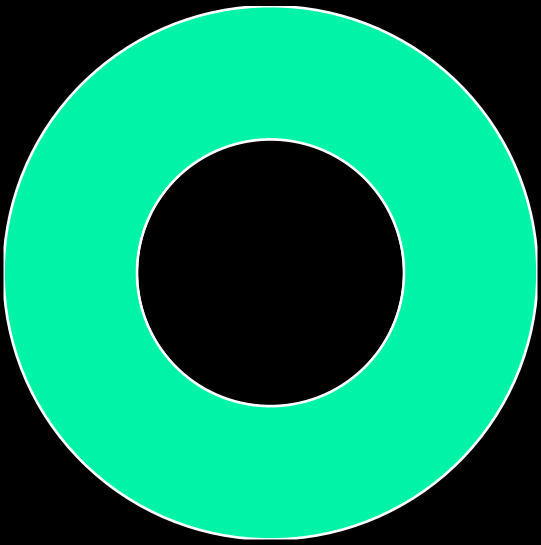
Every issue being identified and validated has its unique symbolic code assigned to the issue at the security research stage. Cause of the vulnerability reporting flow design, some of the rejected issues could be missing.

FINDINGS SUMMARY

Severity	Number of Findings
Critical	0
High	0
Medium	0
Low	1
Informational	1

Total: 2





WEAKNESSES

This section contains the list of discovered weaknesses.

FNG-22

OPTIMISATION OF NFT TRANSFER OUT BY ID

SEVERITY: Low

PATH:

CERC721NoBorrow.sol:doNFTTransferOutByIds:L179-214

REMEDIATION:

See description

STATUS: Fixed

DESCRIPTION:

CERC721NoBorrow implements a function **doNFTTransferOutByIds** to transfer NFTs out to the receiver using specific IDs.

In the function, it loops over each given NFT ID and the currently held NFTs of the user. Once found, the NFT is transferred to the receiver and the NFT is deleted from the **userHeldNFTs** array.

However, in the case where the given NFT ID is the last one in the array, it will do a redundant SSTORE of the NFT ID to the last element of the array on line 207 before popping the same value. This can be prevented by checking the index against the length.

```
    function doNFTTransferOutByIds(address to, uint[] memory nftIds) internal {
        uint[] storage userHeldNFTs_ = userHeldNFTs[to];
        uint len = userHeldNFTs_.length;
        uint nftCount = nftIds.length;
        if (len < nftCount) {
            revert NftAmountTooHigh();
        }

        IERC721 underlying_ = IERC721(underlying);
        for (uint i = 0; i < nftCount;) {
            uint nftID = nftIds[i];
            uint assetIndex = len;
            for (uint j = 0; j < len;) {
                if (userHeldNFTs_[j] == nftID) {
                    assetIndex = j;
                    break;
                }
                unchecked { j++; }
            }

            if (assetIndex == len) {
                revert NftNotFound(nftID);
            }

            underlying_.transferFrom(address(this), to, nftID);

             // copy last item in list to location of item to be removed, reduce
length by 1
            len = len - 1;
            userHeldNFTs_[assetIndex] = userHeldNFTs_[len];
            userHeldNFTs_.pop();

            unchecked { i++; }
        }
```

Check for the index to be equal to the length and prevent redundant storing a value before deleting it. For example, change line 207 to:

```
if (assetIndex != len) {
  userHeldNFTs_[assetIndex] = userHeldNFTs_[len];
}
```

FNG-21

CUSTOM ERROR

SEVERITY: Informational

PATH:

CErc721.sol:L318

REMEDIATION:

Consider replacing this with a custom error to save on byte code size.

STATUS: Fixed

DESCRIPTION:

In CERC721.sol on line 318 a validation check is performed using the **revert** function with a reason string.

```
if (redeemTokens == 0 && redeemAmount > 0) {
    revert("redeemTokens zero");
}
```

hexens × Fungify.