Long Lost Friends

WhIsBe Vandalz

SMART CONTRACT AUDIT

Made by Auditor Labs

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of What is Beauty Inc. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (26.06.2022)	Layout
0.4 (01.07.2022)	Automated Security Testing
	Manual Security Testing
0.5 (04.07.2022)	Verify Claims and Test Deployment
0.6 (05.07.2022)	Testing SWC Checks
0.9 (07.07.2022)	Summary and Recommendation
1.0 (08.07.2022)	Final document
1.1 (18.07.2022)	Re-check 48b4efef6f2e396c3af21389eaa2e96f65a74741
1.2 (25.07.2022)	Added deployed contract

2. About the Project and Company

Company address:

What is Beauty Inc.
475 5th Ave, New York
NY 10017, United States

Website: https://whisbeverse.wtf

Twitter: https://twitter.com/whisbe

Instagram: https://instagram.com/whisbe

Discord: https://discord.gg/whisbeverse

NiftyGateway: https://niftygateway.com/marketplace/artist/5954

OpenSea: https://opensea.io/collection/whisbe

2.1 Project Overview

Vandalz is a collection of 11,111 unique 3D animated NFTs that live in the WhIsBeVerse. The WhIsBeVerse uniquely blends both physical and digital worlds together to truly connect communities. The collection shares the narrative of innocence lost, representing a turning point in WhIsBe's life. In this evolution of the series, the artist empowers his collectors to unlock their Vandalz' personal story, tying it to life changing events that have helped shape who they are today.

About WhIsBe

Contemporary artist WhIsBe has established a formidable reputation in both the renegade world of street art but also the mainstream art world encompassing the museum, gallery and arena of public installations. His moniker, shorthand for "What is Beauty", is at once innocuous and sweet and introduces more substantive themes of cultural examination and subversion that underline much of his body of work.

WhIsBe wanted to share his message with a broad spectrum of people, not only those who have access to museums and galleries and began putting up self-sanctioned work in the street in 2011. WhIsBe has partnered with charities and corporations ranging from the Art Works Charity Foundation to Charity Water to Red Bull to COACH and has exhibited work at notable events including Art Basel, Context & Art Show, Scope Art Show, and Art Southampton.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i. Review of the specifications, sources, and instructions provided to Auditor Labs to make sure we understand the size, scope, and functionality of the smart contract.
    ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditor Labs describe.
2. Testing and automated analysis that includes the following:
    i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
./ERC721RandomlyAssignVandalzTier.sol	7afaeccb2f674d60175a3a6472c2552a
./types/Errors.sol	649bc3d9caa38533ad0ed864683a2e7f
./types/DataTypes.sol	b14467acb2cb5883eb0b7a3ab6bec73b
./GoodKarmaToken.sol	3d33fecc845d7ce01c064162d7ff7e9b
./WhIsBeVandalzPeriphery.sol	58f2cbc5be7fc9e39427a7df451096ec
./ERC721Extensions/ERC721LimitedSupply.sol	76afb7251bda6b143fda43aec235e292
./ERC721Extensions/IERC721Burnable.sol	baf664680f5a67618525bc04a7c30f60
./WhisbeVandalz.sol	ad84b875c9711418b83f5152f5d1ef4b
./ERC721RandomlyAssignVandalzTierPeriphery.sol	94a39d70bfa1e922bd1c0420a7b603a6
./Constants.sol	6c1ab41dc6f4b546eae620dff25e8679

## 5.2 Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/access/Ownable.sol
@openzeppelin/contracts/interfaces/IERC2981.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/interfaces/IERC2981.sol
@openzeppelin/contracts/security/ReentrancyGuard.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/security/ReentrancyGuard.sol
@openzeppelin/contracts/token/ERC1155/ERC1155.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC1155/ERC1155.sol
@openzeppelin/contracts/token/ERC1155/IERC1155.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC1155/IERC1155.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/token/ERC721/ERC721.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC721/ERC721.sol
@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC721/IERC721Receiver.sol

Dependency / Import Path	Source
@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC721/extensions/ERC721Burnable.sol
@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC721/utils/ERC721Holder.sol
@openzeppelin/contracts/utils/Address.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/utils/Address.sol
@openzeppelin/contracts/utils/Counters.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/utils/Counters.sol
@openzeppelin/contracts/utils/Strings.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/utils/Strings.sol
@openzeppelin/contracts/utils/cryptography/MerkleProof.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/utils/cryptography/MerkleProof.sol
@openzeppelin/contracts/utils/introspection/ERC165Storage.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/utils/introspection/ERC165Storage.sol
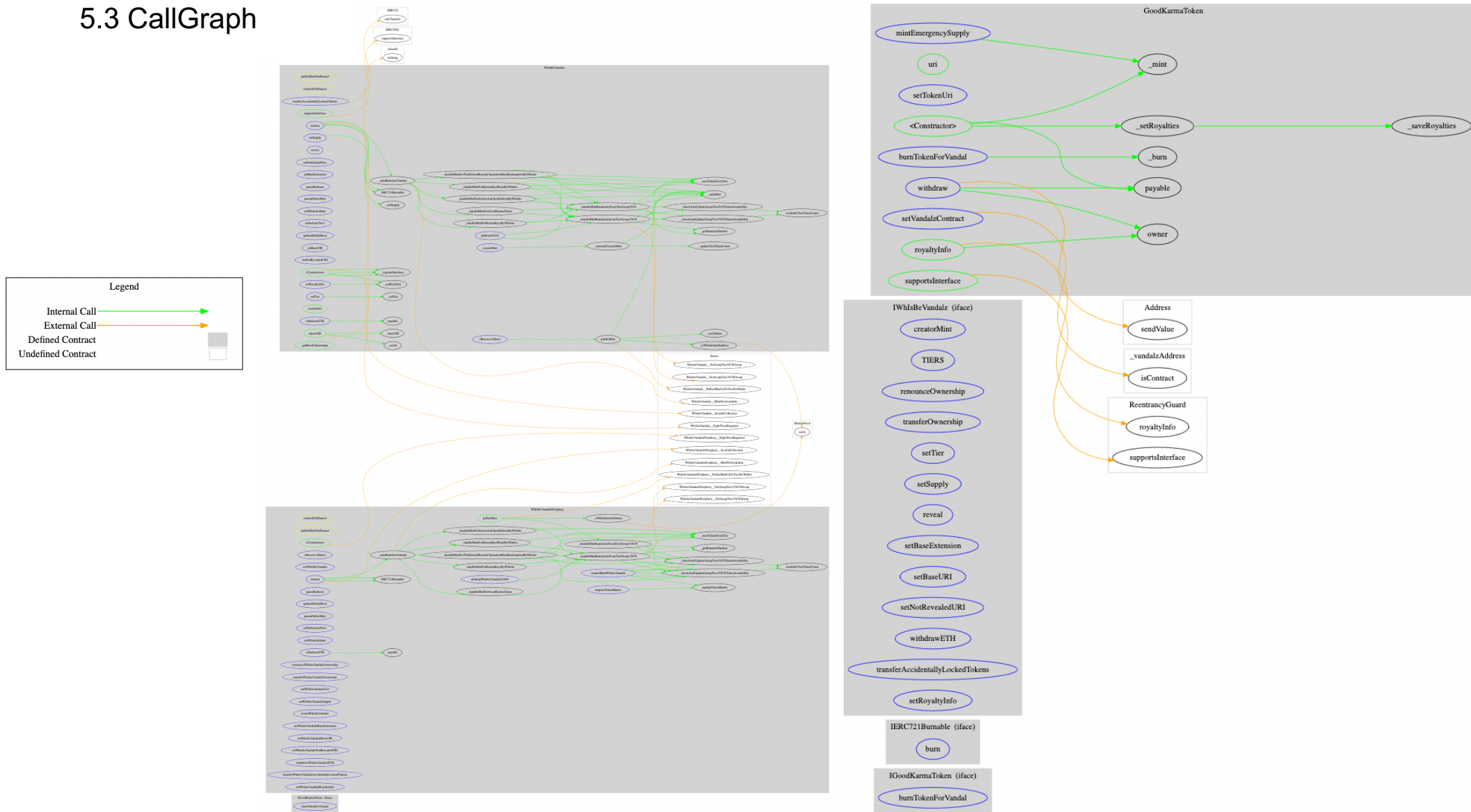
5.4 Inheritance Graph
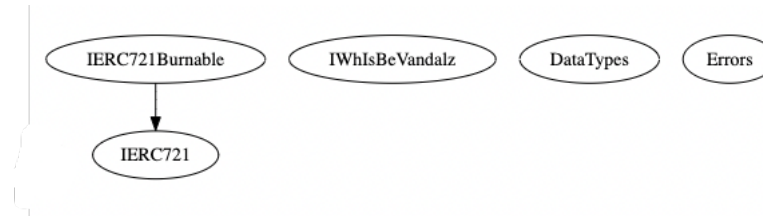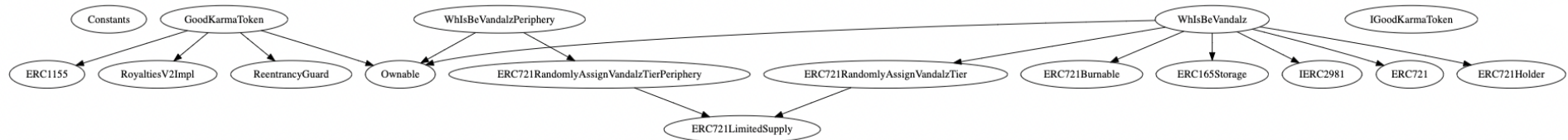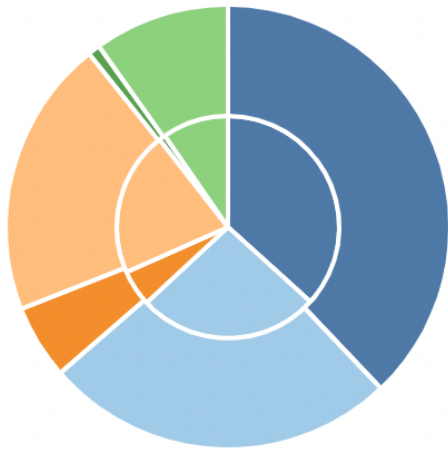
5.5 Source Lines & Risk
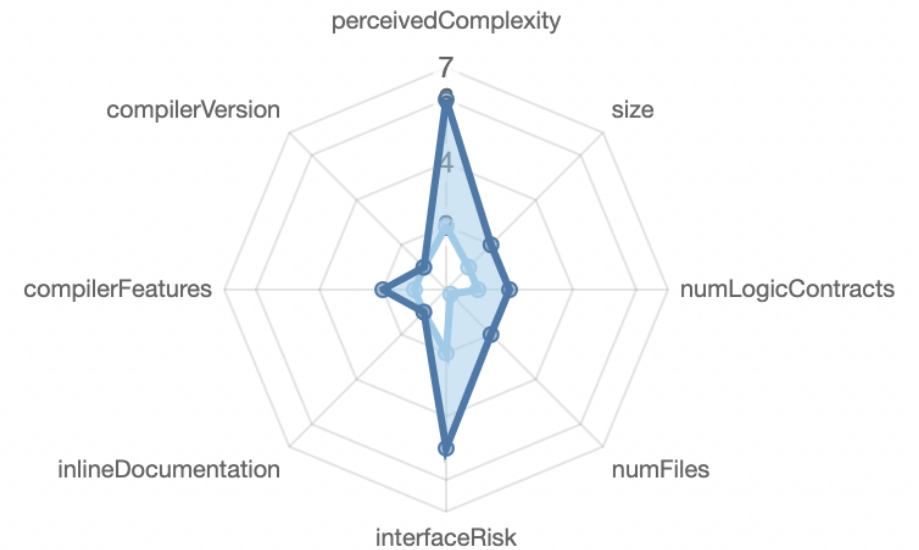
5.6 Capabilities

📝 Solidity Versions observed	🖊️ Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`^0.8.7` `^0.8.0`		`yes`		

📥 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🟥 Uses Hash Functions	🖊️ ECRecover	🌀 New/Create/Create2
`yes`			`yes`		

Exposed Functions
This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐 Public	💰 Payable
79	4

External	Internal	Private	Pure	View
63	123	18	0	20

StateVariables

Total	🌐 Public
52	43

5.7 Source Unites in Scope

Source: https://github.com/Aegis-Studio-Dev/whisbe-vandalz
Last commit: 2f3259b4949085babc1cf69ec7afd8f13234a7b6

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📚	contracts/Constants.sol	1		15	15	9	5	6	
🎨	contracts/ERC721RandomlyAssignVandalzTierPeriphery.sol	1		165	157	87	50	47	🎰
📝🔍	contracts/WhisbeVandalz.sol	1	1	723	681	324	284	313	💰📤🎰
🔍	contracts/ERC721Extensions/IERC721Burnable.sol		1	19	18	4	12	5	
🎨	contracts/ERC721Extensions/ERC721LimitedSupply.sol	1		100	100	38	46	22	
📝🔍	contracts/WhIsBeVandalzPeriphery.sol	1	2	574	526	274	195	357	💰📤🎰
📝	contracts/GoodKarmaToken.sol	1		100	94	68	14	72	
📚	contracts/types/DataTypes.sol	1		23	23	10	11	1	
📚	contracts/types/Errors.sol	1		19	19	17	1	1	
🎨	contracts/ERC721RandomlyAssignVandalzTier.sol	1		154	150	81	51	44	🎰

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	Totals	9	4	1892	1783	912	669	868	💰📤🧮

Legend: [▬]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

6. Scope of Work

The Long Lost Friends Team provided us with the files that needs to be tested. The scope of the audit is the WhIsBe Vandalz contract.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The ERC-721 / 1155 Token standard is correctly implemented
- Deployer/Owner cannot mint any new Token
- Deployer/Owner cannot burn or lock user funds
- Deployer/Owner cannot pause the contract
- The smart contract is coded according to the newest standards and in a secure way.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview
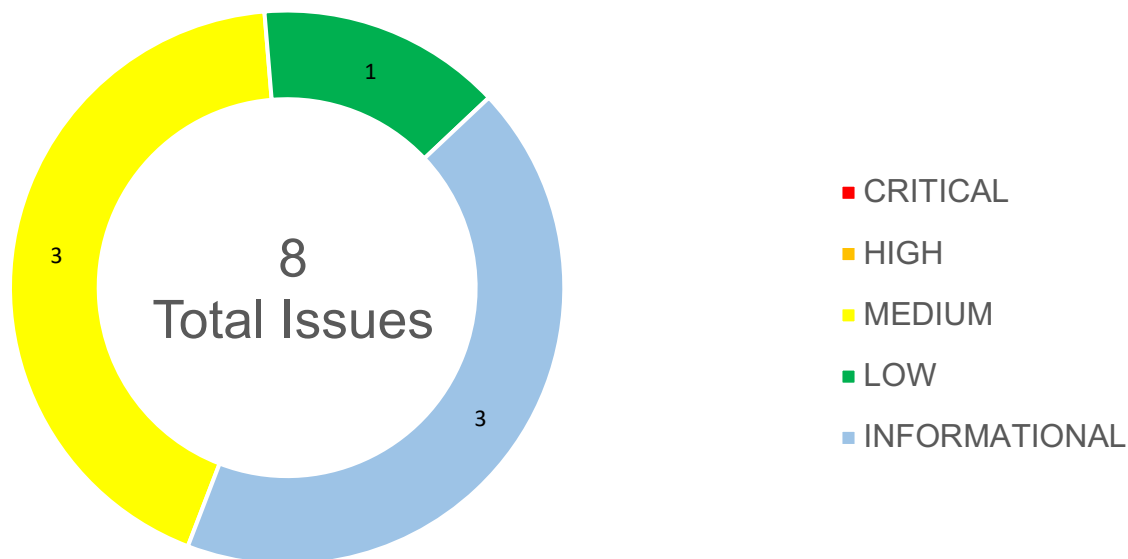
No	Title	Severity	Status
6.2.1	Predictable Random Number	MEDIUM	FIXED
6.2.2	Centralization Risk	MEDIUM	ACKNOWLEDGED
6.2.3	Unlimited Mint	MEDIUM	ACKNOWLEDGED
6.2.4	Missing Value Verification	LOW	ACKNOWLEDGED
6.2.5	Floating Pragma Version Identified	INFORMATIONAL	ACKNOWLEDGED
6.2.6	Missing Natspec Documentation	INFORMATIONAL	ACKNOWLEDGED
6.2.7	Missing Error Message	INFORMATIONAL	ACKNOWLEDGED
6.2.8	Storing Data Via baseURI	INFORMATIONAL	ACKNOWLEDGED

6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES
During the audit, experts found **0 Critical issues** in the code of the smart contract.

HIGH ISSUES
During the audit, experts found **0 High issues** in the code of the smart contract.

MEDIUM ISSUES
During the audit, experts found **3 Medium issues** in the code of the smart contract.

6.2.1 Predictable Random Number
Severity: MEDIUM
Status: FIXED
Code: CWE-330
File(s) affected: ERC721RandomlyAssignVandalzTier.sol, ERC721RandomlyAssignVandalzTierPeriphery.sol
Commit: b48127fbdd9985d5c70ebadef771c02433b31f54, 08ba5d083907e57790bb51907b2c67f3e062d285

Attack / Description	By using the blockhash and the block variable attributes, malicious users can predict the outcome of the random number generation.

Code	Line 137 - 153 (ERC721RandomlyAssignVandalzTier.sol)
	
	``solidity
	function _getRandomNumber() internal view virtual returns (uint256) {
	    return
	        uint256(
	            keccak256(
	                abi.encodePacked(
	``

```
                        msg.sender,
                        block.coinbase,
                        blockhash(block.number),
                        block.gaslimit,
                        block.timestamp,
                        tokenCount(),
                        availableTokenCount(),
                        totalSupply()
                    )
                )
            );
    }
}
```

Line 148 - 165 (ERC721RandomlyAssignVandalzTierPeriphery.sol)

```
function _getRandomNumber() internal view virtual returns (uint256) {
        return
            uint256(
                keccak256(
                    abi.encodePacked(
                        msg.sender,
                        block.coinbase,
                        blockhash(block.number),
                        block.gaslimit,
                        block.timestamp,
                        tokenCount(),
                        availableTokenCount(),
                        totalSupply()
                    )
                )
            );
    }
```

	``` } ```
Result/Recommendation	Consider using an oracle or Chainlink VRF to generate randomness. https://docs.chain.link/docs/chainlink-vrf/

6.2.2 Centralization Risk
Severity: MEDIUM
Status: ACKNOWLEDGED
Code: NA
File(s) affected: WhisbeVandalz.sol, GoodKarmaToken.sol

Attack / Description	The Owner has the rights to mint token, set baseURI and pause mint / redemption. This represents a significant centralization risk, where the Owner has too much control over the contract. The auditor has not recognized any multi-signature structure.

Code	Line 310 – 331 (WhisbeVandalz.sol)
	<pre>function creatorMint(address _to, uint256[] memory _tokenIds) external onlyOwner {
	    for (uint256 _i; _i < _tokenIds.length; _i++) {
	        _internalCreatorMint(_to, _tokenIds[_i]);
	    }
	}
	
	/**
	 * @notice
	 * @dev
	 */
	function _internalCreatorMint(address _to, uint256 _tokenId) internal ensureAvailability {
	    _updateTierTokenCount(_tokenId);
	    _safeMint(_to, _tokenId);
	}</pre>

```solidity
    /**
     * @notice
     * @dev
     */
    function setSupply(uint256 _supply) external onlyOwner {
        _setSupply(_supply);
    }
```

Line 74 (WhisbeVandalz.sol)

```solidity
  bool public publicMintPaused = true;
```

Line 92 (WhisbeVandalz.sol)

```solidity
  bool public redeemPaused = true;
```

Line 420 - 422 (WhisbeVandalz.sol)
```solidity
function setBaseURI(string memory _newBaseURI) external onlyOwner {
        baseURI = _newBaseURI;
    }
```

Line 47 - 50 (GoodKarmaToken.sol)

```solidity
  function mintEmergencySupply(uint256 _numNewTokens) external onlyOwner {
        MAX_SUPPLY += _numNewTokens;
        _mint(msg.sender, GoodKarmaTokenID, _numNewTokens, "");
    }
```

Result/Recommendation	If the Owner wallet/private key gets into the wrong hands, caused by a leak or hack, then it's easily possible to harm the project. We recommend protecting the Owner wallet with a multi-signature structure, such as gnosis safe or add on-chain governance for roles.

6.2.3 Unlimited Mint
Severity: MEDIUM
Status: ACKNOWLEDGED
Code: NA
File(s) affected: WhisbeVandalz.sol, GoodKarmaToken.sol

Attack / Description	The creatorMint and mintEmergencySupply function allows the Owner to mint new tokens to a specific address. If this function is not used carefully, it can cause inflation of the tokens.

Code	Line 310 - 314 (WhisbeVandalz.sol)
	
	``solidity
	function creatorMint(address _to, uint256[] memory _tokenIds) external onlyOwner {
	        for (uint256 _i; _i < _tokenIds.length; _i++) {
	            _internalCreatorMint(_to, _tokenIds[_i]);
	        }
	    }
	``
	
	Line 47 - 50 (GoodKarmaToken.sol)
	
	``solidity
	  function mintEmergencySupply(uint256 _numNewTokens) external onlyOwner {
	        MAX_SUPPLY += _numNewTokens;
	        _mint(msg.sender, GoodKarmaTokenID, _numNewTokens, "");
	    }
	``

Result/Recommendation	Consider adding some restrictions to the mint or add multi-signature to the Owner functions.

LOW ISSUES

During the audit, experts found **1 Low issue** in the code of the smart contract.

6.2.4 Missing Value Verification
Severity: LOW
Status: ACKNOWLEDGED
Code: CWE-354
File(s) affected: ERC721RandomlyAssignVandalzTier.sol, ERC721RandomlyAssignVandalzTierPeriphery.sol

Attack / Description	Certain functions lack a value safety check, the values of the arguments should be verified to allow only the ones that comply with the contract's logic. In the constructor, the _startFrom variable should not have a big value, otherwise, incrementing the token count will cause an overflow, therefore, no new tokens can be minted.

Code	Line 34 - 36 (ERC721RandomlyAssignVandalzTier.sol, ERC721RandomlyAssignVandalzTierPeriphery.sol)
	``solidity
	constructor(uint256 _totalSupply, uint256 _startFrom) ERC721LimitedSupply(_totalSupply) {
	        startFrom = _startFrom;
	    }
	``

Result/Recommendation	Certain functions lack a value safety check, the values of the arguments should be verified to allow only the ones that comply with the contract's logic.

INFORMATIONAL ISSUES

During the audit, experts found **4 Informational issues** in the code of the smart contract.

6.2.5 Floating Pragma Version Identified
Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: SWC-103
File(s) affected: ALL

Attack / Description	It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.
Code	Line 1 `pragma solidity ^0.8.7;`
Result/Recommendation	It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version. i.e. Pragma solidity 0.8.7

6.2.6 Missing Natspec Documentation
Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: ALL

Attack / Description	Solidity contracts can use a special form of comments to provide rich documentation for function, return variables, and more. This special form is named Ethereum Natural Language Specification Format (NatSpec).
Code	```/**
* @notice
* @dev
*/```
Result/Recommendation	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand the smart contract. Some comments are already added but increasing the comments will help to support the understanding of each function.

6.2.7 Missing Error Message
Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: WhisbeVandalz.sol

Attack / Description	Require statements check if a condition is true and allow code to flow only if the condition is true. If the condition is false, require will throw an error, the rest of the code will not be executed and the transaction will revert.
Code	Line 268 (WhisbeVandalz.sol) ```require(_collectionsLen == _tokenIds.length, "");```
Result/Recommendation	It is a best practice to add error messages to each require statement.

6.2.8 Storing Data Via baseURI
Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: WhisbeVandalz.sol

Attack / Description	In the current implementation the baseURI is not hardcoded or on-chain generated, means the owner/creator is free to choose the way how the metadata file is stored.

Code	Line 420 - 422 (WhisbeVandalz.sol)
	
	``solidity
	function setBaseURI(string memory _newBaseURI) external onlyOwner {
	        baseURI = _newBaseURI;
	    }
	``

Result/Recommendation	We recommend using IPFS and pinning services to make the metadata behind the baseURI permanently stored or implement into the event creation service. To ensure that data persists on IPFS, and is not deleted during garbage collection, data can be pinned to one or more IPFS nodes. Pinning gives you control over disk space and data retention. As such, you should use that control to pin any content you wish to keep on IPFS indefinitely. Check more information here: https://docs.ipfs.io/concepts/persistence/#persistence-versus-permanence Keep in mind even if you use an IPFS Service, the file will only exist as long if it is "pinned". And you still may need a dedicated gateway to serve your files with a decent speed, which may lead to your metadata requests timing out in the future. Please investigate SVG generated on-chain visuals and on-chain stored metadata, for persistent storage.

6.3 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✅
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✅
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✅
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✅
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✅
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✅
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✅
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✅

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅

ID	Title	Relationships	Test Result
[SWC-103](#)	Floating Pragma	[CWE-664: Improper Control of a Resource Through its Lifetime](#)	X
[SWC-102](#)	Outdated Compiler Version	[CWE-937: Using Components with Known Vulnerabilities](#)	✅
[SWC-101](#)	Integer Overflow and Underflow	[CWE-682: Incorrect Calculation](#)	✅
[SWC-100](#)	Function Default Visibility	[CWE-710: Improper Adherence to Coding Standards](#)	✅

6.4. Verify Claims

6.4.1 The ERC-721 / 1155 Token standard is correctly implemented
Status: tested and verified ✅

6.4.2 Deployer/Owner cannot mint any new Token
Status: tested and verified **X**
See: 6.2.3 Unlimited Mint

6.4.3 Deployer/Owner cannot burn or lock user funds
Status: tested and verified ✅

6.4.4 Deployer/Owner cannot pause the contract
Status: tested and verified **X**
See: 6.2.2 Centralization Risk

6.4.5 The smart contract is coded according to the newest standards and in a secure way.
Status: tested and verified ✅

7. Executive Summary

Two (2) independent experts performed an unbiased and isolated audit of the smart contract codebase.

The main goal of the audit was to verify the claims regarding the security and functions of the smart contract. During the audit, no critical, 3 medium, 1 low and 4 informational issues have been found, after the manual and automated security testing.

We advise Long Lost Friends team to implement the recommendations contained in all 8 of our findings, to further enhance the code's security and readability.

Update (27.07.2022): Long Lost Friends have addressed all issues and will secure the owner rights via multi-signature wallet structure.

8. Deployed Smart Contract

VERIFIED

https://etherscan.io/address/0x31719ea121e5e77b12efaf134693bb2633d0dc3f#code

9. About the Auditor

Auditor Labs is a professional software development firm, founded in 2017 and based in Germany. They show ways, opportunities, risks and offer comprehensive blockchain solutions. Some of their services include blockchain development, smart contract audits and consulting.

Auditor Labs conducts code audits on market-leading blockchains such as Hyperledger, Tezos, Ethereum, Binance Smart Chain, and Solana to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of 1Inch, POA Network, Unicrypt, Amun, Furucombo among numerous other top DeFi projects.

Auditor Labs currently secures $100 billion in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the blockchain sector to deliver top-notch smart contract audit solutions, tailored to the clients' evolving business needs.

Check our website for further information: https://auditorlabs.com

How We Work

1 --------

PREPARATION
Supply our team with audit ready code and additional materials

2 --------

COMMUNICATION
We setup a real-time communication tool of your choice or communicate via e-mails.

3 --------

AUDIT
We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.

4 --------

FIXES
Your development team applies fixes while consulting with our auditors on their safety.

5 --------

REPORT
We check the applied fixes and deliver a full report on all steps done.