



**Zeitls**

**Protocol**

**SMART CONTRACT AUDIT**

**31.01.2023**

**Made in Germany by Chainsulting.de**



## Table of contents

1. Disclaimer.....	4
2. About the Project and Company .....	5
2.1 Project Overview.....	6
3. Vulnerability & Risk Level .....	7
4. Auditing Strategy and Techniques Applied.....	8
4.1 Methodology .....	8
5. Metrics .....	9
5.1 Tested Contract Files .....	9
5.2 Used Code from other Frameworks/Smart Contracts .....	10
5.3 CallGraph .....	12
5.4 Inheritance Graph .....	13
5.5 Source Lines & Risk .....	14
5.6 Capabilities .....	15
5.7 Source Unites in Scope .....	16
6. Scope of Work.....	17
6.1 Findings Overview .....	18
6.2 Manual and Automated Vulnerability Test.....	19
6.2.1 Overpowered Owner rights.....	19
6.2.2 Missing Zero Address Checks .....	21
6.2.3 Misleading In-Line Comment.....	22
6.2.4 Floating Pragma Version Identified .....	23
6.2.5 Storing Metadata via tokenURI .....	23

6.3 SWC Attacks .....	25
6.4. Verify Claims .....	29
6.5 Unit Tests.....	30
7. Executive Summary.....	42
8. Deployed Contracts .....	42
9. About the Auditor .....	43

## 1. Disclaimer

The audit makes no statements or warranties 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 Zeitls AG. 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 (16.12.2022)	Layout
0.4 (19.12.2022)	Automated Security Testing Manual Security Testing
0.5 (20.12.2022)	Verify Claims and Test Deployment
0.6 (21.12.2022)	Testing SWC Checks
0.9 (22.12.2022)	Summary and Recommendation
1.0 (22.12.2022)	Final document
1.1 (02.01.2023)	Re-check commit 911026a171f9f80a97a773d4ce095b514f0f36bc
1.2 (31.01.2023)	Added deployed contracts

## 2. About the Project and Company

**Company address:**

Zeitls AG  
Hungerstrasse 52  
8832 Wilen b. Wollerau  
Switzerland



**Website:** <https://zeitls.ch>

**Twitter:** <https://twitter.com/ZeitlsAG>

**Facebook:** <https://www.facebook.com/Zeitls-115685684451739>

**Instagram:** <https://www.instagram.com/zeitlsag>

**LinkedIn:** <https://www.linkedin.com/company/zeitls>

**Medium:** <https://zeitls.medium.com>

**Discord:** <https://discord.gg/NUVGzpFnhk>

## 2.1 Project Overview

In the age of digitalization and rapid technological development, Zeitls recognized the importance of preserving historically important pieces of art for years to come by transferring them into the digital world. More practically, converting rare and historically valuable pieces of art into interactive, non-degradable crypto assets that can be owned or traded by claiming their non-physical rights. Zeitls aim to preserve historical pieces in their digital form for all eternity, while accumulating funding and liquidity for their preservation in the real world.

Zeitls are striving to provide museums and art collectors with the means to keep their collections pristine, or even expand their collection. For that reason, Zeitls, are an agent that bridges not only museums and collectors, but also the real world with the digital one, and the past with the future. They believe art and history should be accessible to everyone, in a decentralized environment.

A marker of human achievement, a timeless testament to the power of imagination, and the expression of the highest order of skills. Zeitls, is giving everyone the possibility of owning unique historical objects as authentic digital collectibles, because they believe pieces of human history should be available to everyone.

### 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 Chainsulting 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 Chainsulting 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)
./contracts/opensea/IProxyRegistry.sol	b459aa8a1cafa4e2541bda8020081fbc
./contracts/weth/IWETH.sol	838a63817a55eeb814236b0051f6ff51
./contracts/IZtDevilsTreasury.sol	d517ec06b1548d2fcd26e7fd120e83ef
./contracts/IZtDevils.sol	ffa0fda278bcec83cef0908009f8b089
./contracts/ZtDevilsWhitelist.sol	7c8d062be4940c279171b1e7bbb44246
./contracts/ZtDevilsTreasury.sol	68b1f9a1a06cde9cd7bffe550fba22cd
./contracts/ZtDevilsAuctionHouse.sol	b748ecac94e8857bc2098b3b377b8505
./contracts/ZtDevils.sol	9a15a8c75ccd4dda5870c2708b2a18dc

Update 02.01.2023 (commit 911026a171f9f80a97a773d4ce095b514f0f36bc)

File	Fingerprint (MD5)
./contracts/opensea/IProxyRegistry.sol	4e24af9c60d832afe3bc7d23d141b79a
./contracts/weth/IWETH.sol	5d429d0157581c31de763a3cfbb15011
./contracts/IZtDevilsTreasury.sol	e827d4480e483807e531c784fb7a15d9
./contracts/IZtDevils.sol	d0dfe08233c06d9bec32fa13b031c55e
./contracts/ZtDevilsWhitelist.sol	9e7bb73694fb9ea74ff508a08eb4ebf2
./contracts/ZtDevilsTreasury.sol	e827d4480e483807e531c784fb7a15d9

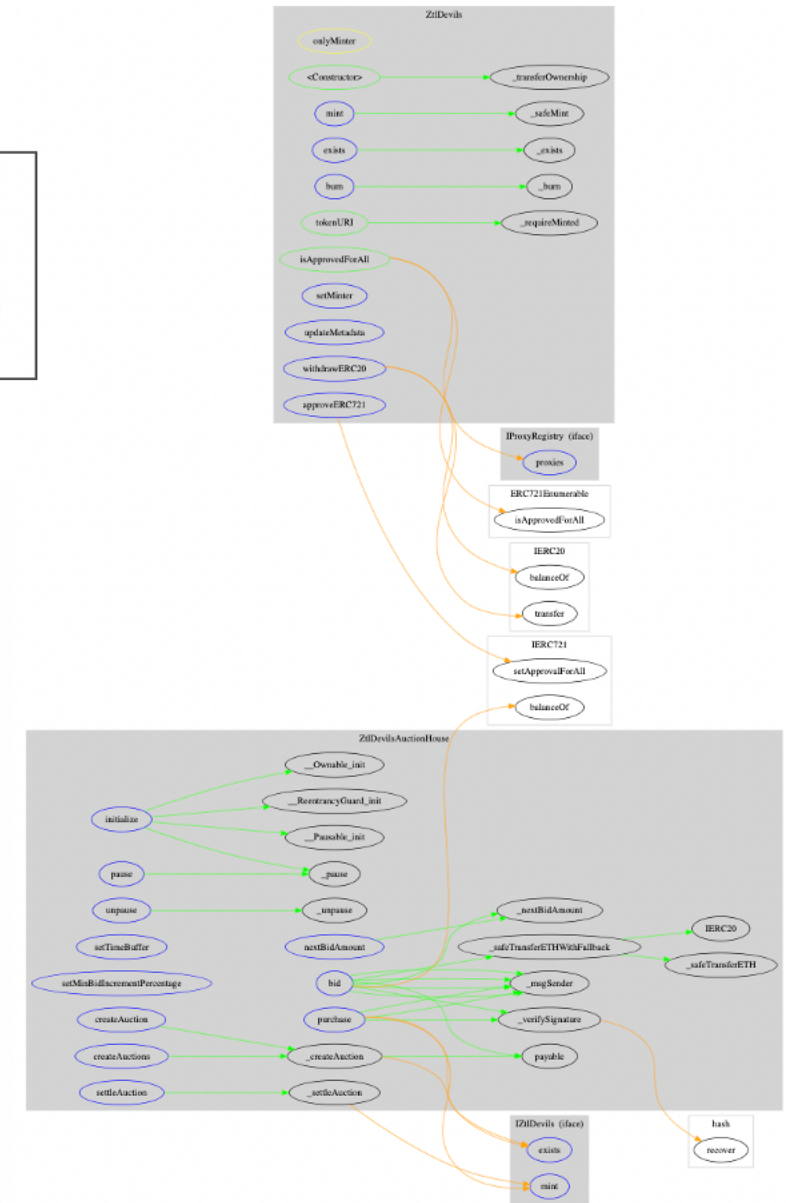
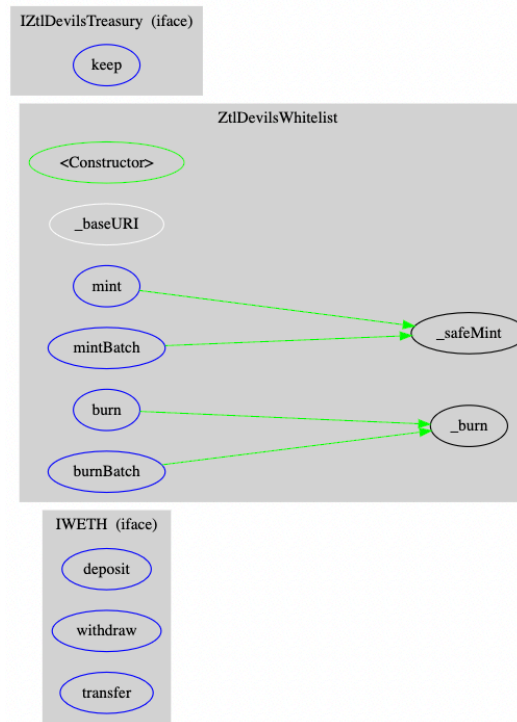
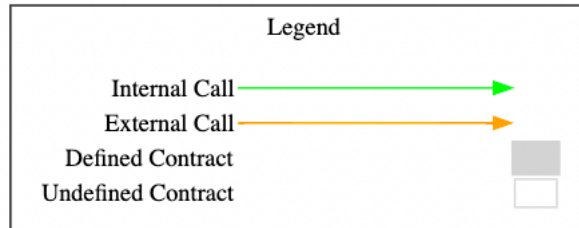
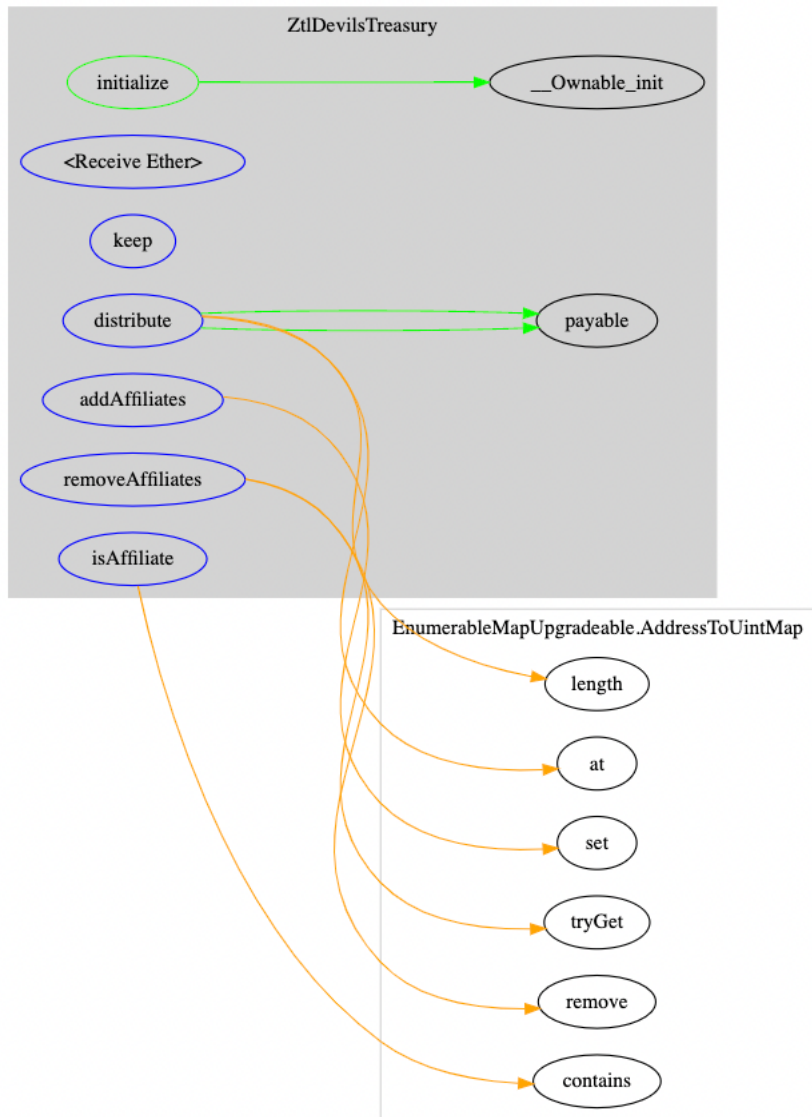
./contracts/ZtlDevilsAuctionHouse.sol	812d9ff423db6436190c711b142e42b1
./contracts/ZtlDevils.sol	73a7044f70e63e7eccfdbbe2f1fe2b2c

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

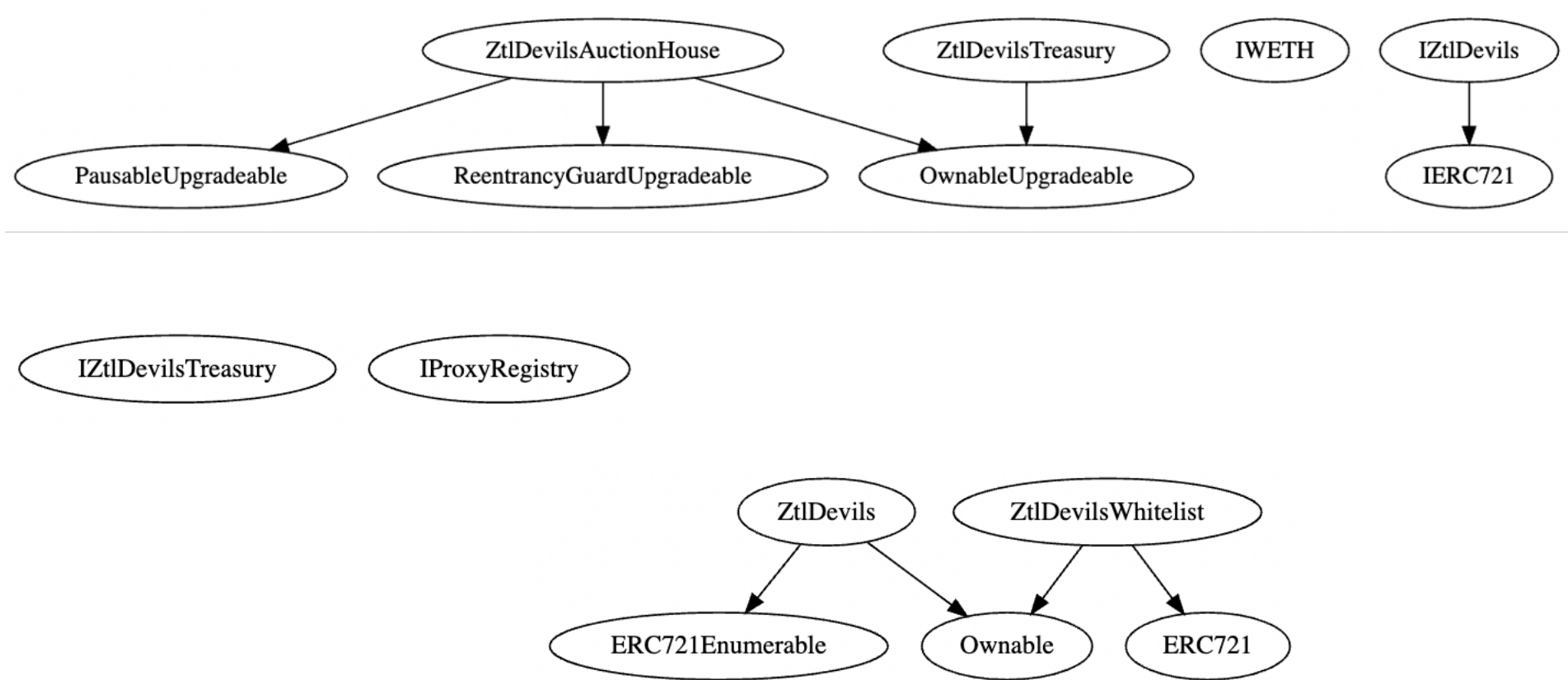
Dependency / Import Path	Source
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/access/OwnableUpgradeable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/access/OwnableUpgradeable.sol</a>
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/security/PausableUpgradeable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/security/PausableUpgradeable.sol</a>
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/security/ReentrancyGuardUpgradeable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/security/ReentrancyGuardUpgradeable.sol</a>
@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/utils/cryptography/ECDSAUpgradeable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/utils/cryptography/ECDSAUpgradeable.sol</a>
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableMapUpgradeable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/utils/structs/EnumerableMapUpgradeable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.7.3/contracts/utils/structs/EnumerableMapUpgradeable.sol</a>
@openzeppelin/contracts/access/Ownable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/access/Ownable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/access/Ownable.sol</a>

Dependency / Import Path	Source
@openzeppelin/contracts/token/ERC20/IERC20.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC20/IERC20.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC20/IERC20.sol</a>
@openzeppelin/contracts/token/ERC721/ERC721.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC721/ERC721.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC721/ERC721.sol</a>
@openzeppelin/contracts/token/ERC721/IERC721.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC721/IERC721.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC721/IERC721.sol</a>
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC721/extensions/ERC721Enumerable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC721/extensions/ERC721Enumerable.sol</a>

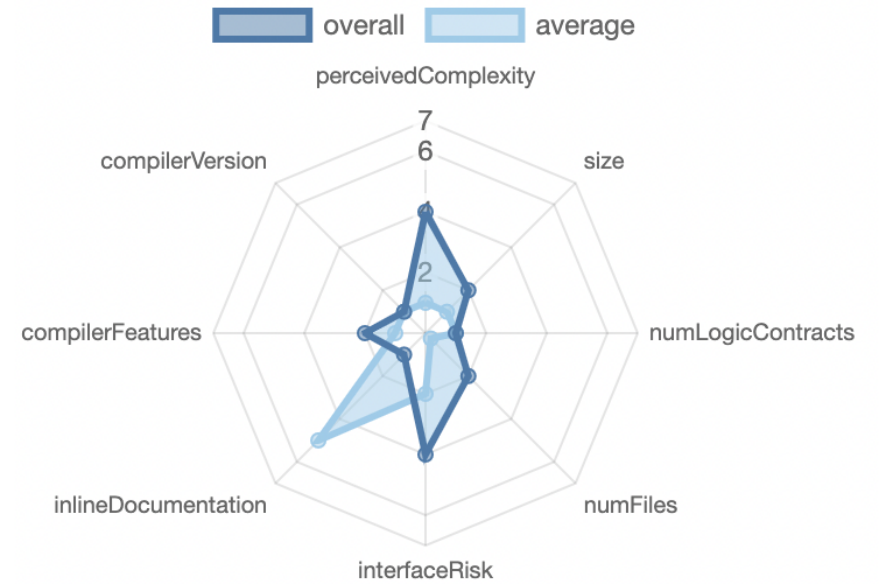
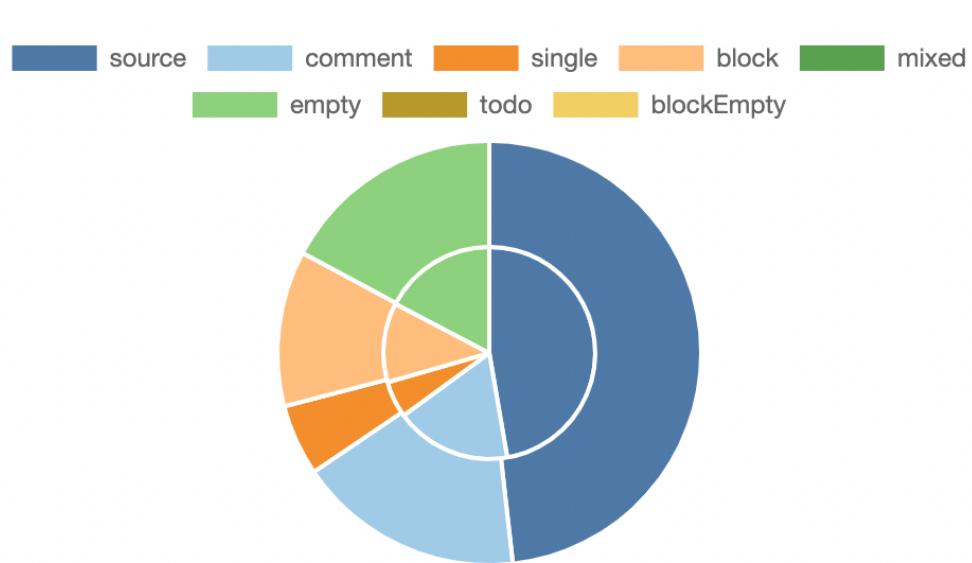
## 5.3 CallGraph





## 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
<input type="text" value="^0.8.16"/>			<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRecover	 New/Create/Create2
<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>

### Exposed Functions

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

 <b>Public</b>	 <b>Payable</b>				
<input type="text" value="38"/>	<input type="text" value="6"/>				
<b>External</b>	<b>Internal</b>	<b>Private</b>	<b>Pure</b>	<b>View</b>	
<input type="text" value="35"/>	<input type="text" value="38"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="10"/>	

### StateVariables

















<b>Total</b>	 <b>Public</b>
<input type="text" value="20"/>	<input type="text" value="17"/>

## 5.7 Source Unites in Scope

Source: <https://github.com/hooy/zeitls-devils-museum-kaunas>

Commit: a355deb099f82992efef421e161bad148e1fb79c

Branch: main

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/ZtlDevils.sol	1	_____	142	142	77	40	72	
	contracts/ZtlDevilsAuctionHouse.sol	1	_____	317	308	172	75	151	
	contracts/ZtlDevilsTreasury.sol	1	_____	113	113	70	17	68	
	contracts/weth/IWETH.sol	_____	1	11	6	3	1	10	
	contracts/ZtlDevilsWhitelist.sol	1	_____	46	46	33	1	34	_____
	contracts/IZtlDevils.sol	_____	1	10	8	4	1	7	_____
	contracts/IZtlDevilsTreasury.sol	_____	1	9	8	4	1	6	
	contracts/opensea/IProxyRegistry.sol	_____	1	7	6	3	1	3	_____
 	<b>Totals</b>	<b>4</b>	<b>4</b>	<b>655</b>	<b>637</b>	<b>366</b>	<b>137</b>	<b>351</b>	



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

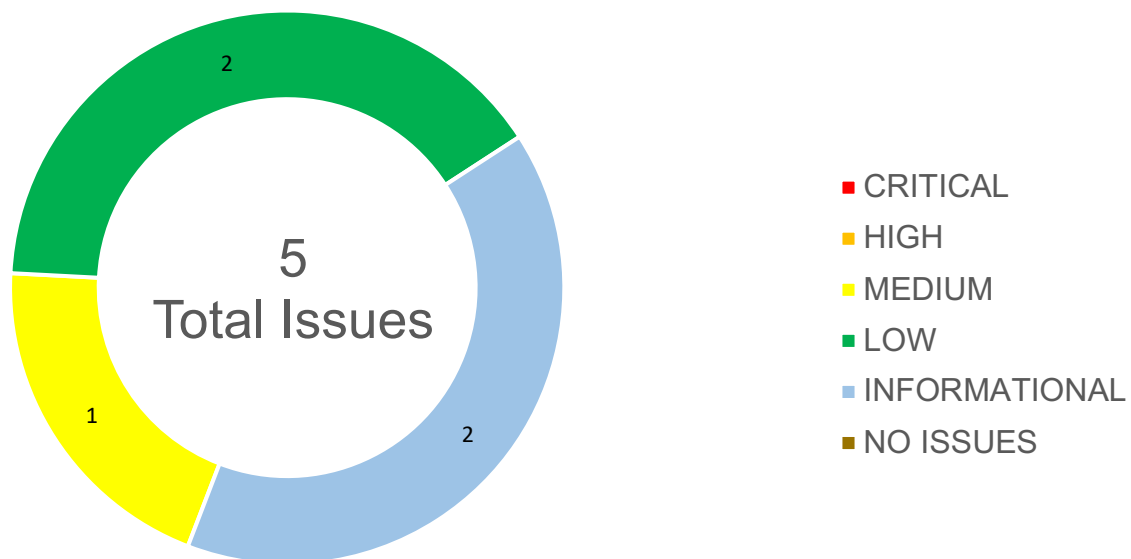
The Zeitls Team provided us with the files that needs to be tested. The scope of the audit is the Zeitls Protocol contract.

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

- The ERC-721 token standard is correctly implemented
- The Whitelist is working as expected
- The Treasury is working as expected
- Auctions are working as expected
- 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	Overpowered Owner rights	MEDIUM	ACKNOWLEDGED
6.2.2	Missing Zero Address Checks	LOW	FIXED
6.2.3	Misleading In-Line Comment	LOW	FIXED
6.2.4	Floating Pragma Version Identified	INFORMATIONAL	FIXED
6.2.5	Storing Metadata via tokenURI	INFORMATIONAL	ACKNOWLEDGED

## 6.2 Manual and Automated Vulnerability Test

### CRITICAL ISSUES

During the audit, Chainsulting's experts found **0 Critical issues** in the code of the smart contract.

### HIGH ISSUES

During the audit, Chainsulting's experts found **0 High issues** in the code of the smart contract.

### MEDIUM ISSUES

During the audit, Chainsulting's experts found **1 Medium issue** in the code of the smart contract.

#### 6.2.1 Overpowered Owner rights

Severity: MEDIUM

Status: ACKNOWLEDGED

Code: NA

File(s) affected: ZtlDevils.sol, ZtlDevilsAuctionHouse.sol, ZtlDevilsTreasury.sol, ZtlDevilsWhitelist.sol

Update: Zeitle Team is aware of the potential risks of losing a single key, therefore their intention is to leverage Gnosis Safe (Multisig)

Attack / Description	Code Snippet	Result/Recommendation
The owner has extensive rights. The auditor has not recognized any roles, governance or multi-sig structure.	Line 104 (ZtlDevils.sol) <pre>function setMinter(address _minter) external onlyOwner {</pre> Line 114 (ZtlDevils.sol)	The owner has rights to distribute royalties, set the Minter, Pause the auction, and burn token.  If the owner wallet/private key gets into the wrong hands, caused by a leak or hack, then it's easily

	<pre>function updateMetadata(uint[] calldata ids, string[] calldata uris) external onlyOwner</pre> <p>Line 132 (ZtlDevils.sol)</p> <pre>function withdrawERC20(IERC20 _tokenContract) external onlyOwner</pre> <p>Line 139 (ZtlDevils.sol)</p> <pre>function approveERC721(IERC721 _tokenContract) external onlyOwner {</pre> <p>Line 113 (ZtlDevilsAuctionHouse.sol)</p> <pre>function pause() external onlyOwner</pre> <p>Line 121 (ZtlDevilsAuctionHouse.sol)</p> <pre>function unpause() external onlyOwner {</pre> <p>Line 129 (ZtlDevilsAuctionHouse.sol)</p> <pre>function setTimeBuffer(uint40 _timeBuffer) external onlyOwner {</pre> <p>Line 138 (ZtlDevilsAuctionHouse.sol)</p> <pre>function setMinBidIncrementPercentage(uint8 _minBidIncrementPercentage) external onlyOwner {</pre> <p>And within ZtlDevilsTreasury.sol, ZtlDevilsWhitelist.sol</p>	<p>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.</p>
--	--	--

## LOW ISSUES

During the audit, Chainsulting's experts found **2 Low issues** in the code of the smart contract.

### 6.2.2 Missing Zero Address Checks

Severity: LOW

Status: FIXED

Code: NA

File(s) affected: ZtlDeviIs.sol, ZtlDeviIsAuctionHouse.sol

Update: <https://github.com/hooy/zeiIs-deviIs-museum-kaunas/commit/911026a171f9f80a97a773d4ce095b514f0f36bc>

<b>Attack / Description</b>	In the current implementation, there are several addresses set without checking for the zero address. This can lead to unintended behaviour.
<b>Code</b>	<div>Line 104 - 107 (ZtlDeviIs.sol)</div> <pre>function setMinter(address _minter) external onlyOwner {     minter = _minter;     emit MinterUpdated(_minter); }</pre> <div>Line 55 - 62 (ZtlDeviIsAuctionHouse.sol)</div> <pre>// The ERC721 whitelist token contract for community members IERC721 public whitelist;  // The address of the WETH contract address public weth;  // The address of the signer for auction participation address public signer;</pre>
<b>Result/Recommendation</b>	It is recommended to check address values for correctness. This can be done by checking for zero address or exclude in a require.

### 6.2.3 Misleading In-Line Comment

Severity: LOW

Status: FIXED

Code: NA

File(s) affected: ZtlDevilsAuctionHouse.sol

Update: <https://github.com/hooy/zeitls-devils-museum-kaunas/commit/911026a171f9f80a97a773d4ce095b514f0f36bc>

<b>Attack / Description</b>	In the current codebase are some comments not matching with the current project. This could lead to misleading understanding and usage of the code.
<b>Code</b>	<p>Line 108 (ZtlDevilsAuctionHouse.sol) * @notice Pause the Nouns auction house.</p> <p>Line 118 (ZtlDevilsAuctionHouse.sol) * @notice Unpause the Nouns auction house.</p> <p>Line 271 (ZtlDevilsAuctionHouse.sol) * @dev If there are no bids, the Noun is burned.</p>
<b>Result/Recommendation</b>	It is recommended to correct the misleading comments and replace Nouns with Devils or Zeitls

## INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **2 Informational issues** in the code of the smart contract.

#### 6.2.4 Floating Pragma Version Identified

Severity: INFORMATIONAL

Status: FIXED

Code: SWC-103

File(s) affected: ALL

Update: <https://github.com/hooy/zeitls-devils-museum-kaunas/commit/911026a171f9f80a97a773d4ce095b514f0f36bc>

<b>Attack / Description</b>	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.
<b>Code</b>	e.g. Line 3 <code>pragma solidity ^0.8.16;</code>
<b>Result/Recommendation</b>	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. <code>pragma solidity 0.8.16</code>

#### 6.2.5 Storing Metadata via tokenURI

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: ZtlDevils.sol

Update: Functionality exists only because business expectations are to share the token contract across different physical collections, thus metadata update feature is mandatory.

<b>Attack / Description</b>	In the current implementation the tokenURI is not hardcoded, means the owner/creator is free to choose the way how the metadata file is stored.
<b>Code</b>	Line 86 - 95 (ZtlDevils.sol) <pre> function tokenURI(uint256 tokenId) public view override returns (string memory) {     _requireMinted(tokenId);      uint ipfsIndex = metadataIds.length - 1;     for (uint i = 1; i &lt; metadataIds.length; i++) {         if (tokenId &lt; metadataIds[i]) {             ipfsIndex = i - 1;             break;         }     } } </pre>
<b>Result/Recommendation</b>	<p>We recommend using IPFS and pinning services to make the metadata behind the tokenURI permanently stored.</p> <p>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.</p> <p>Check more information here:  <a href="https://docs.ipfs.io/concepts/persistence/#persistence-versus-permanence">https://docs.ipfs.io/concepts/persistence/#persistence-versus-permanence</a></p> <p>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.</p> <p>Please investigate SVG generated on-chain visuals and on-chain stored metadata, for persistent storage.</p>







## 6.3 SWC Attacks

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


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

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


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

## 6.4. Verify Claims


6.4.1 The ERC-721 token standard is correctly implemented

**Status:** tested and verified 


6.4.2 The Whitelist is working as expected

**Status:** tested and verified 


6.4.3 The Treasury is working as expected

**Status:** tested and verified 

6.4.4 Auctions are working as expected

**Status:** tested and verified 

6.4.5 The smart contract is coded according to the newest standards and in a secure way.

**Status:** tested and verified 

## 6.5 Unit Tests

### ZtlDevilsAuctionHouse

#### deployment

- ✓ should set contract owner address contract from constructor args (539ms)
- ✓ should set time buffer
- ✓ should set min increment percentage

#### listing

- ✓ list single auction
- ✓ list many auctions

#### forbid listing

- ✓ should forbid not permitted listing
- ✓ should forbid with zero reserve price
- ✓ should forbid secondary listing
- ✓ should forbid listing on existing token

#### bidding

- ✓ should activate auction timer with the first bid
- ✓ should calculate next bid amount (98ms)
- ✓ should refund the previous bidder when the following user creates a bid
- ✓ should extend auction when bid made in buffer time

#### settle

- ✓ should allow auction settle and token claim after release time (45ms)

#### direct purchase

- ✓ should permit purchase
- ✓ should forbid expired signature
- ✓ should forbid underpaid transaction
- ✓ should forbid purchase for existing auction
- ✓ should forbid purchase for existing token
- ✓ should forbid invalid signature

#### whitelist

- ✓ should allow bid when sender is whitelisted
- ✓ should forbid bid on limited auction without whitelist

### ZtlDevils

#### deployment

- ✓ should set contract owner address contract from constructor args

#### management

- ✓ should change minter
- ✓ should change owner

#### token

- ✓ should mint token
- ✓ should burn token

#### metadata

- ✓ should update metadata and return correct link according to token id (41ms)
- ✓ should expand metadata links (87ms)

#### marketplace

- ✓ should bypass approved for all for a marketplace (45ms)
- ✓ should forbid approved for all for a stranger

#### ZtlDevilsTreasury

- ✓ auction sale (59ms)
- ✓ royalty (48ms)

#### affiliates

- ✓ add affiliates
- ✓ remove affiliates (38ms)
- ✓ forbid share limit exceeding (42ms)
- ✓ forbid share overwrite

#### distribution

- ✓ distribute auction income (72ms)

#### ZtlDevilsWhitelist

##### whitelist management

- ✓ add (42ms)
- ✓ add batch

#### ERC165

- ✓ supportsInterface uses less than 30k gas
- ✓ all interfaces are reported as supported
- ✓ all interface functions are in ABI

```
with minted tokens
  balanceOf
    when the given address owns some tokens
      ✓ returns the amount of tokens owned by the given address
    when the given address does not own any tokens
      ✓ returns 0
    when querying the zero address
      ✓ throws
  ownerOf
    when the given token ID was tracked by this token
      ✓ returns the owner of the given token ID
    when the given token ID was not tracked by this token
      ✓ reverts
  transfers
    via transferFrom
      when called by the owner
        ✓ transfers the ownership of the given token ID to the given address
        ✓ emits a Transfer event
        ✓ clears the approval for the token ID
        ✓ adjusts owners balances
        ✓ adjusts owners tokens by index
      when called by the approved individual
        ✓ transfers the ownership of the given token ID to the given address
        ✓ emits a Transfer event
        ✓ clears the approval for the token ID
        ✓ adjusts owners balances
        ✓ adjusts owners tokens by index
      when called by the operator
        ✓ transfers the ownership of the given token ID to the given address
        ✓ emits a Transfer event
        ✓ clears the approval for the token ID
        ✓ adjusts owners balances
        ✓ adjusts owners tokens by index
      when called by the owner without an approved user
```



- ✓ transfers the ownership of the given token ID to the given address
- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index
- when sent to the owner
  - ✓ keeps ownership of the token
  - ✓ clears the approval for the token ID
  - ✓ emits only a transfer event
  - ✓ keeps the owner balance
  - ✓ keeps same tokens by index
- when the address of the previous owner is incorrect
  - ✓ reverts
- when the sender is not authorized for the token id
  - ✓ reverts
- when the given token ID does not exist
  - ✓ reverts
- when the address to transfer the token to is the zero address
  - ✓ reverts
- via safeTransferFrom
  - with data
    - to a user account
      - when called by the owner
        - ✓ transfers the ownership of the given token ID to the given address
        - ✓ emits a Transfer event
        - ✓ clears the approval for the token ID
        - ✓ adjusts owners balances
        - ✓ adjusts owners tokens by index
      - when called by the approved individual
        - ✓ transfers the ownership of the given token ID to the given address
        - ✓ emits a Transfer event
        - ✓ clears the approval for the token ID
        - ✓ adjusts owners balances
        - ✓ adjusts owners tokens by index
      - when called by the operator

- ✓ transfers the ownership of the given token ID to the given address
- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index
- when called by the owner without an approved user
  - ✓ transfers the ownership of the given token ID to the given address
  - ✓ emits a Transfer event
  - ✓ clears the approval for the token ID
  - ✓ adjusts owners balances
  - ✓ adjusts owners tokens by index
- when sent to the owner
  - ✓ keeps ownership of the token
  - ✓ clears the approval for the token ID
  - ✓ emits only a transfer event
  - ✓ keeps the owner balance
  - ✓ keeps same tokens by index
- when the address of the previous owner is incorrect
  - ✓ reverts
- when the sender is not authorized for the token id
  - ✓ reverts
- when the given token ID does not exist
  - ✓ reverts
- when the address to transfer the token to is the zero address
  - ✓ reverts
- to a valid receiver contract
  - ✓ calls onERC721Received
  - ✓ calls onERC721Received from approved
- when called by the owner
  - ✓ transfers the ownership of the given token ID to the given address
  - ✓ emits a Transfer event
  - ✓ clears the approval for the token ID
  - ✓ adjusts owners balances
  - ✓ adjusts owners tokens by index
- when called by the approved individual

- ✓ transfers the ownership of the given token ID to the given address
- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index
- when called by the operator
  - ✓ transfers the ownership of the given token ID to the given address
  - ✓ emits a Transfer event
  - ✓ clears the approval for the token ID
  - ✓ adjusts owners balances
  - ✓ adjusts owners tokens by index
- when called by the owner without an approved user
  - ✓ transfers the ownership of the given token ID to the given address
  - ✓ emits a Transfer event
  - ✓ clears the approval for the token ID
  - ✓ adjusts owners balances
  - ✓ adjusts owners tokens by index
- when sent to the owner
  - ✓ keeps ownership of the token
  - ✓ clears the approval for the token ID
  - ✓ emits only a transfer event
  - ✓ keeps the owner balance
  - ✓ keeps same tokens by index
- when the address of the previous owner is incorrect
  - ✓ reverts
- when the sender is not authorized for the token id
  - ✓ reverts
- when the given token ID does not exist
  - ✓ reverts
- when the address to transfer the token to is the zero address
  - ✓ reverts
- with an invalid token id
  - ✓ reverts
- without data
- to a user account

when called by the owner

- ✓ transfers the ownership of the given token ID to the given address
- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the approved individual

- ✓ transfers the ownership of the given token ID to the given address
- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the operator

- ✓ transfers the ownership of the given token ID to the given address
- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when called by the owner without an approved user

- ✓ transfers the ownership of the given token ID to the given address
- ✓ emits a Transfer event
- ✓ clears the approval for the token ID
- ✓ adjusts owners balances
- ✓ adjusts owners tokens by index

when sent to the owner

- ✓ keeps ownership of the token
- ✓ clears the approval for the token ID
- ✓ emits only a transfer event
- ✓ keeps the owner balance
- ✓ keeps same tokens by index

when the address of the previous owner is incorrect

- ✓ reverts

when the sender is not authorized for the token id

- ✓ reverts

when the given token ID does not exist

- ✓ reverts
- when the address to transfer the token to is the zero address
  - ✓ reverts
- to a valid receiver contract
  - ✓ calls onERC721Received
  - ✓ calls onERC721Received from approved
- when called by the owner
  - ✓ transfers the ownership of the given token ID to the given address
  - ✓ emits a Transfer event
  - ✓ clears the approval for the token ID
  - ✓ adjusts owners balances
  - ✓ adjusts owners tokens by index
- when called by the approved individual
  - ✓ transfers the ownership of the given token ID to the given address
  - ✓ emits a Transfer event
  - ✓ clears the approval for the token ID
  - ✓ adjusts owners balances
  - ✓ adjusts owners tokens by index
- when called by the operator
  - ✓ transfers the ownership of the given token ID to the given address
  - ✓ emits a Transfer event
  - ✓ clears the approval for the token ID
  - ✓ adjusts owners balances
  - ✓ adjusts owners tokens by index
- when called by the owner without an approved user
  - ✓ transfers the ownership of the given token ID to the given address
  - ✓ emits a Transfer event
  - ✓ clears the approval for the token ID
  - ✓ adjusts owners balances
  - ✓ adjusts owners tokens by index
- when sent to the owner
  - ✓ keeps ownership of the token
  - ✓ clears the approval for the token ID
  - ✓ emits only a transfer event
  - ✓ keeps the owner balance

- ✓ keeps same tokens by index
- when the address of the previous owner is incorrect
  - ✓ reverts
- when the sender is not authorized for the token id
  - ✓ reverts
- when the given token ID does not exist
  - ✓ reverts
- when the address to transfer the token to is the zero address
  - ✓ reverts
- with an invalid token id
  - ✓ reverts
- to a receiver contract returning unexpected value
  - ✓ reverts (41ms)
- to a receiver contract that reverts with message
  - ✓ reverts (53ms)
- to a receiver contract that reverts without message
  - ✓ reverts (41ms)
- to a receiver contract that panics
  - ✓ reverts
- to a contract that does not implement the required function
  - ✓ reverts

safe mint

via safeMint

- calls onERC721Received – with data
- calls onERC721Received – without data
- to a receiver contract returning unexpected value
  - reverts
- to a receiver contract that reverts with message
  - reverts
- to a receiver contract that reverts without message
  - reverts
- to a receiver contract that panics
  - reverts
- to a contract that does not implement the required function
  - reverts

```
approve
  when clearing approval
    when there was no prior approval
      ✓ clears approval for the token
      ✓ emits an approval event
    when there was a prior approval
      ✓ clears approval for the token
      ✓ emits an approval event
  when approving a non-zero address
    when there was no prior approval
      ✓ sets the approval for the target address
      ✓ emits an approval event
    when there was a prior approval to the same address
      ✓ sets the approval for the target address
      ✓ emits an approval event
    when there was a prior approval to a different address
      ✓ sets the approval for the target address
      ✓ emits an approval event
  when the address that receives the approval is the owner
    ✓ reverts
  when the sender does not own the given token ID
    ✓ reverts
  when the sender is approved for the given token ID
    ✓ reverts
  when the sender is an operator
    ✓ sets the approval for the target address
    ✓ emits an approval event
  when the given token ID does not exist
    ✓ reverts
setApprovalForAll
  when the operator willing to approve is not the owner
    when there is no operator approval set by the sender
      ✓ approves the operator
      ✓ emits an approval event
    when the operator was set as not approved
```

- ✓ approves the operator
- ✓ emits an approval event
- ✓ can unset the operator approval
- when the operator was already approved
  - ✓ keeps the approval to the given address
  - ✓ emits an approval event
- when the operator is the owner
  - ✓ reverts
- getApproved
  - when token is not minted
    - ✓ reverts
  - when token has been minted
    - ✓ should return the zero address
  - when account has been approved
    - ✓ returns approved account
- \_mint(address, uint256)
  - ✓ reverts with a null destination address (64ms)
  - with minted token
    - ✓ emits a Transfer event
    - ✓ creates the token
    - ✓ reverts when adding a token id that already exists
- \_burn
  - ✓ reverts when burning a non-existent token id (59ms)
  - with minted tokens
    - with burnt token
      - ✓ emits a Transfer event
      - ✓ deletes the token
      - ✓ reverts when burning a token id that has been deleted
- ERC165
  - ✓ supportsInterface uses less than 30k gas
  - ✓ all interfaces are reported as supported
  - ✓ all interface functions are in ABI



with minted tokens  
totalSupply  
✓ returns total token supply  
tokenOfOwnerByIndex  
when the given index is lower than the amount of tokens owned by the given address  
✓ returns the token ID placed at the given index  
when the index is greater than or equal to the total tokens owned by the given address  
✓ reverts  
when the given address does not own any token  
✓ reverts  
after transferring all tokens to another user  
✓ returns correct token IDs for target  
✓ returns empty collection for original owner  
tokenByIndex  
✓ returns all tokens  
✓ reverts if index is greater than supply  
✓ returns all tokens after burning token 5042 and minting new tokens  
✓ returns all tokens after burning token 79217 and minting new tokens  
  
\_mint(address, uint256)  
✓ reverts with a null destination address  
with minted token  
✓ adjusts owner tokens by index  
✓ adjusts all tokens list  
  
\_burn  
✓ reverts when burning a non-existent token id  
with minted tokens  
with burnt token  
✓ removes that token from the token list of the owner  
✓ adjusts all tokens list  
✓ burns all tokens

259 passing (27s)  
7 pending

## 7. Executive Summary

Two (2) independent Chainsulting 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, no high, one medium, two low and two informational issues have been found, after the manual and automated security testing.

We advise the Zeitls team to implement the recommendations to further enhance the code's security and readability.

Update (02.01.2023): The Zeitls Team addressed all issues and fixed them.

## 8. Deployed Contracts

Zeitls Devils (DEVILS): <https://etherscan.io/address/0xf0b8b1F6685a5a2E95E922D0d45D89efFa7a0cc6#code>

Zeitls Key (ZTL-KEY): <https://etherscan.io/address/0x8EB199D3620759ada6C3377AC6e5d71B3521eA21#code>

ZtlDevilsTreasury: <https://etherscan.io/address/0x907583c591A85A3C8dc3E29294836D9034CF82F4#code>

ZtlDevilsAuctionHouse: <https://etherscan.io/address/0x56cE7EFFa21c995C5fB2E29FFf18Cdf6FC2fe8d2#code>



## 9. About the Auditor

Chainsulting is a professional software development firm, founded in 2017 and based in Germany. They show ways, opportunities, risks and offer comprehensive Web3 solutions. Their services include web3 development, security and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Solana, Tezos, Ethereum, Binance Smart Chain, and Polygon to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of many top DeFi projects.

Chainsulting 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 web3 sector to deliver top-notch smart contract audit solutions, tailored to the clients' evolving business needs.

Check our website for further information: <https://chainsulting.de>

### 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.