

Smart Contract Security Audit V1

Degen Santa NFT

22/12/2021



<https://saferico.com/>

business@saferico.com

https://t.me/SFI_ANN

—

Table of Contents

Table of Contents

Background

Project Information

NFT Information

Executive Summary

File and Function Level Report

File in Scope:

Issues Checking Status

Severity Definitions

Audit Findings

Automatic testing

Testing proves

Inheritance graph

Call graph

Unified Modeling Language (UML)

Functions signature

Automatic general report

Conclusion

Disclaimer

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Website:**
- **Telegram group:**
- **Twitter**
- **WhitePaper:**
- **Platform:**
- **Contract Address:**

NFT Information

- Name: DS
- Total Supply:
- Holders:
- Total transactions:

Contracts address deployed to test net (ETH)

DS NFT contract on ETH test net.

<https://goerli.etherscan.io/address/0x993a2fe2fE1636608FeA3F71264719e9cdeF8b53>

<https://rinkeby.etherscan.io/address/0xdf4b45a68c2a8604decad245d5211d7fab3185d7>

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.

Well Secured	
Secured	✓
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 0 low, 2 very low-level issues and 2 notes in all solidity files of the contract

The files:

NFT.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
NFT.sol	d6a47c84c08ff72514f5bdb71b6e60c6d1102a0147c4aa12e3cbaeb5b25168dd	0x993a2fe2fE1636608FeA3F71264719e9cdeF8b53

- Contract: DegenSanta
- Inherit: ERC721URIStorage
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	Passed
symbol	✓	Read / public	Passed
whitelistPS	✓	Read / public	Passed
salesOpen	✓	Read / public	Passed
whitelistPS	✓	Read / public	Passed
balanceOf	✓	Read / public	Passed
Owner	✓	Read / public	Passed
baseUri	✓	Read / public	Passed
maxMint	✓	Read / public	Passed
getApprovedForAll	✓	Read / public	Passed
normalPrice	✓	Read / public	Passed
getApproved	✓	Read / public	Passed

ownerOf	✓	Read / public	Passed
tokenURI	✓	Read / private	Passed
totalFreeSale	✓	Read / private	Passed
totalMint	✓	Read / public	Passed
totalPrivateSale	✓	Read / private	Passed
whitelist1F	✓	Read / private	Passed
setSales	✓	Write / public	Passed
approve	✓	Write / public	Passed
safeTransferFrom	✓	Write / public	Passed
safeTransferFrom	✓	Write / public	Passed
addListFree	✓	Write / public	Passed
addListPrivate	✓	Write / public	Passed
awardItem	✓	Write / payable	Passed
claimBalance	✓	Write / public	Passed
changeOwner	✓	Write / public	Passed
setApprovalForAll	✓	Write / public	Passed
includeInReward	✓	Write / public	Passed

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical:

No critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found

Medium:

No Medium severity vulnerabilities were found.

Low:

No Low severity vulnerabilities were found.

Very Low:

#Pragm version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.0 instead of ^0.8.0). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Remediation

Remove the ^ sign to lock the pragma version

Status: **Acknowledged**.

#Missing zero address validation

Description

When setting the new owner address, it should be checked for zero address. Otherwise, they may lose the ability to use the privileged functions.

```
function changeOwner(address adres) external onlyOwner{
    owner=adres;
}
```

Remediation

Use the require statement to check for zero addresses.

Status: **Acknowledged**

Notes:

#Missing SPDX-License-Identifier:

Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see <https://spdx.org> for more information .

Remediation

Add License Identifier

```
// SPDX-License-Identifier: UNLICENSE
```

Status: **Acknowledged**

#Compiler version is old

Description

The compiler being used was released a year ago. It's recommended to use more recent compiler version, there can be benefits like reduction in bytecode size etc.

Status: **Acknowledged**

Automatic Testing

1- Check for security

d6a47c84c08ff72514f5bdb71b6e60c6d1102a0147c4aa12e3cbaeb5b25168dd

File: NFT.sol | Language: solidity | Size: 3452 bytes | Date: 2021-12-22T14:54:12.882Z

Critical	High	Medium	Low	Note
0	0	0	0	2



2- SOLIDITY STATIC ANALYSIS

SOLIDITY STATIC ANALYSIS

☒ Select all ☒ Autorun Run

▼ Security

☒ Select Security

- ☒ **Transaction origin:**
'tx.origin' used
- ☒ **Check-effects-interaction:**
Potential reentrancy bugs
- ☒ **Inline assembly:**
Inline assembly used
- ☒ **Block timestamp:**
Can be influenced by miners
- ☒ **Low level calls:**
Should only be used by experienced devs
- ☒ **Block hash:**
Can be influenced by miners
- ☒ **Selfdestruct:**
Contracts using destructed contract can be broken

▼ Gas & Economy

☒ Select Gas & Economy

- ☒ **Gas costs:**
Too high gas requirement of functions
- ☒ **This on local calls:**
Invocation of local functions via 'this'
- ☒ **Delete dynamic array:**
Use require/assert to ensure complete deletion
- ☒ **For loop over dynamic array:**
Iterations depend on dynamic array's size
- ☒ **Ether transfer in loop:**
Transferring Ether in a for/while/do-while loop

SOLIDITY STATIC ANALYSIS

▼ ERC

☒ Select ERC

- ☒ **ERC20:**
'decimals' should be 'uint8'

▼ Miscellaneous

☒ Select Miscellaneous

- ☒ **Constant/View/Pure functions:**
Potentially constant/view/pure functions
- ☒ **Similar variable names:**
Variable names are too similar
- ☒ **No return:**
Function with 'returns' not returning
- ☒ **Guard conditions:**
Ensure appropriate use of require/assert
- ☒ **Result not used:**
The result of an operation not used
- ☒ **String length:**
Bytes length != String length
- ☒ **Delete from dynamic array:**
'delete' leaves a gap in array
- ☒ **Data truncated:**
Division on int/uint values truncates the result

3- SOLIDITY UNIT TESTING

SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

☒ Select all

☒ tests/NFT_test.sol

Progress: 1 finished (of 1)

PASS testSuite (tests/NFT_test.sol)

✓ Before all

✖

✓ Check success

✖

✓ Check success2

✖

✓ Check failure

✖

✓ Check sender and value

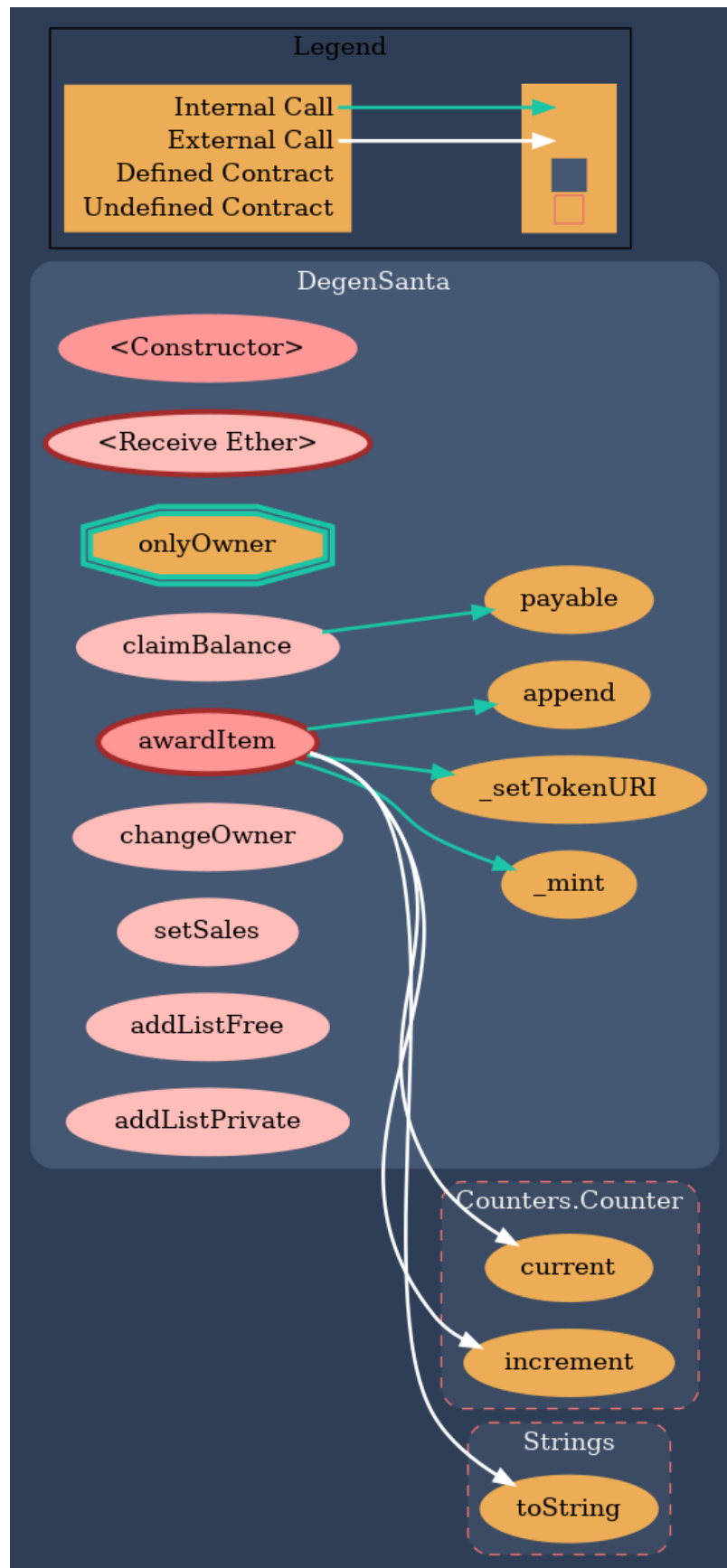
✖

Result for tests/NFT_test.sol

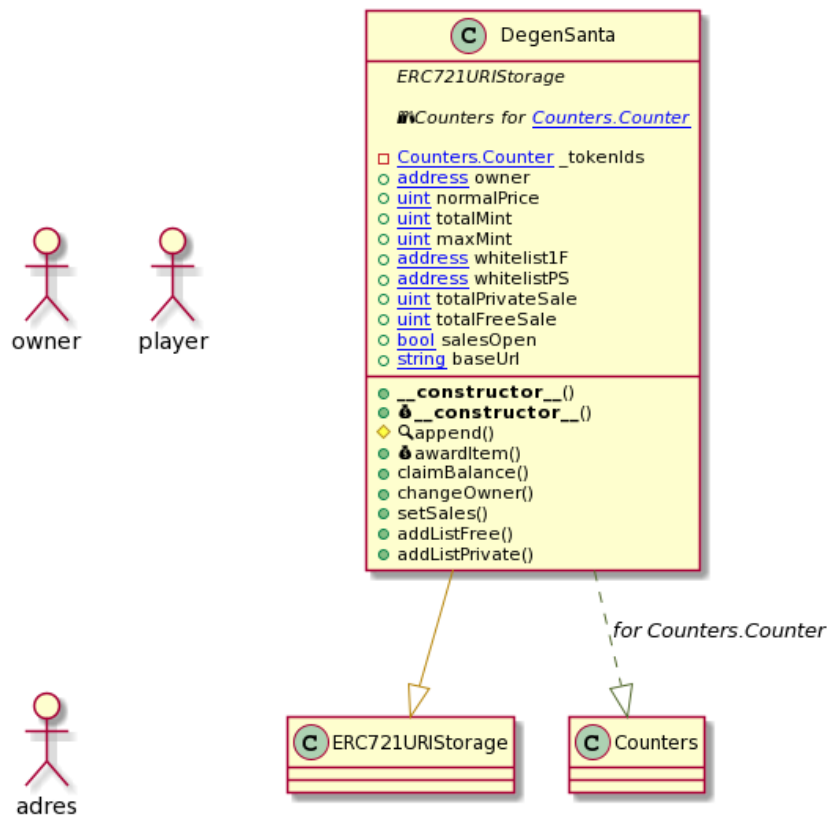
Passing: 5

Total time: 0.46s

4- Call graph



Unified Modeling Language (UML)



Functions signature

Sighash		Function Signature
=====		
47395ced	=>	append(string,string,string)
144c9d4c	=>	awardItem(address,uint256)
e9fad396	=>	claimBalance(uint256)
a6f9dae1	=>	changeOwner(address)
56addaf8	=>	setSales(bool)
734cbcc7	=>	addListFree(address)
c54b305c	=>	addListPrivate(address)

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/NFT.sol	ef4749c3e6b29f55932fcc3f046d639a00638ed3

Contracts Description Table

Contract	Type	Bases	
:	:	:	:
:	:	:	:
:	:	:	:
L	**Function Name**	**Visibility**	**Mutability**
Modifiers			
DegenSanta	Implementation	ERC721URIStorage	
L	<Constructor>	Public !	ERC721
L	<Receive Ether>	External !	NO !
L	append	Internal	
L	awardItem	Public	NO !
L	claimBalance	External !	onlyOwner
L	changeOwner	External !	onlyOwner
L	setSales	External !	onlyOwner
L	addListFree	External !	onlyOwner
L	addListPrivate	External !	onlyOwner

Legend

Symbol	Meaning
:	:
!	Function can modify state
!	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “secured”.

- ✓ No volatile code.
- ✓ Not many high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.