DeHacker

Code Security Assessment

# REAL SMURF CAT

August 14th, 2024

# Contents

## Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

# Issue Categories

Every issue in this report was assigned a severity level from the following:

## Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

## Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

## Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

## Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

## Informational

A vulnerability that has informational character but is not affecting any of the code.

# Overview

## Project Summary

| | |
|---|---|
| **Project Name** | REAL SMURF CAT |
| **Platform** | Ethereum |
| **Website** | smurfcat.org |
| **Type** | DeFi |
| **Language** | Solidity |
| **Codebase** | https://etherscan.io/token/0xfF836A5821E69066c87E268bC51b849FaB94240C |

## Vulnerability Summary

| **Vulnerability Level** | **Total** | **Mitigated** | **Declined** | **Acknowledged** | **Partially Resolved** | **Resolved** |
|---|---|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 | 0 | 0 |
| Major | 2 | 0 | 1 | 2 | 0 | 1 |
| Medium | 1 | 0 | 0 | 0 | 0 | 1 |
| Minor | 1 | 0 | 0 | 1 | 0 | 0 |
| Informational | 1 | 0 | 0 | 1 | 0 | 0 |
| Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

## Audit scope

| ID | File | SHA256 Checksum |
|---|---|---|
| SCH | mainnet | 0e28077e937bb3c9bd7b8fc7e6352a337bca89d08a20eeb292f976f9e9c1c814 |

## Findings

| ID | Title | Severity | Status |
|---|---|---|---|
| SCH-01 | **Initial Token Distribution** | **Major** | Mitigated |
| SCH-02 | **Centralization Risks In SmurfCat.Sol** | **Major** | Resolved |
| SCH-03 | Blacklisted Users Can Transfer Tokens BeforeSetting Up The Blacklist Via A Front Run | Medium | Resolved |
| SCH-04 | Missing Zero Address Validation | Minor | Acknowledged |
| SCH-05 | Function Fails On Valid Inputs | Informational | Acknowledged |

# MAJOR

## SCH-01|Initial Token Distribution

| Issue | Severity | Location | Status |
|---|---|---|---|
| Centralization | Major | SmurfCat.sol: 602 | Mitigated |

### Description

**Important Note**: Certain identification and KYC procedures were attempted to be applied to the project team in order tobetter understand the centralization situation and potential risks of the project. Based on the collected information and furtherconducted investigation, the project failed to pass the criteria with a certain amount of negative signals. Thus the caseanalysis concluded that there is potential high risk to the project. We strongly advise end users to conduct further researchand exercise due diligence before engaging with the project. It is crucial for end users to independently verify and assess allavailable information to make informed decisions.

All of the Real Smurf Cat tokens are sent to the contract deployer or one or several externally-owned account (EOA)addresses. This is a centralization risk because the deployer or the owner(s) of the EOAs can distribute tokens withoutobtaining the consensus of the community. Any compromise to these addresses may allow a hacker to steal and sell tokenson the market, resulting in severe damage to the project.

## Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution planshould be published in a public location that the community can access. The team should make efforts to restrict access tothe private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point offailure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vestingschedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greateraccountability.

## MAJOR

### SCH-02|Centralization Risks In Smurfcat.Sol

| Issue | Severity | Location | Status |
|---|---|---|---|
| Centralization | Major | SmurfCat.sol: 88, 96, 605, 609 | Resolved |

### Description

**Important Note**: Certain identification and KYC procedures were attempted to be applied to the project team in order tobetter understand the centralization situation and potential risks of the project. Based on the collected information and furtherconducted investigation, the project failed to pass the criteria with a certain amount of negative signals. Thus the caseanalysis concluded that there is potential high risk to the project. We strongly advise end users to conduct further researchand exercise due diligence before engaging with the project. It is crucial for end users to independently verify and assess allavailable information to make informed decisions.

In the contract Ownable the role _owner has authority over the functions.

- renounceOwnership() : This function is used by the current owner to renounce their ownership of the contract,effectively leaving the contract without an owner and unable to call onlyOwner functions. Only the owner can callthis function.
- transferOwnership(address newOwner) : Allows the current owner to transfer ownership of the contract to a newaccount ( newOwner ). Only the owner can call this function.

In the contract SmurfCat the role _owner has authority over the functions.

## Description

- blacklist(address _address, bool _isBlacklisting) : This function allows the owner to add or removeaddresses from the blacklist, which prevents them from participating in token transfers. Only the owner of thecontract can call this function.
- setRule(bool _limited, address _uniswapV2Pair, uint256 _maxHoldingAmount, uint256_minHoldingAmount) : Sets trading rules including enabling/disabling trading limits, setting a Uniswap pair address,and defining maximum and minimum holding amounts. Only the owner can call this function.

Any compromise to the _owner account may allow the hacker to take advantage of this authority and update the sensitivesettings and execute sensitive functionalities of the project.

## Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level ofdecentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefullymanage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommendcentralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accountswith enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that wouldalso mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

**Short Term:**

Timelock and Multi sign (⅔, ⅗) combination mitigate by delaying the sensitive operation and avoiding a single point of keymanagement failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

**Long Term:**

Timelock and DAO, the combination, mitigate by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

## Recommendation

**Permanent:**

Renouncing the ownership or removing the function can be considered fully resolved.

- Renounce the ownership and never claim back the privileged roles.

  OR

- Remove the risky functionality.

## MEDIUM

### SCH-03 | Blacklisted Users Can Transfer Tokens Before Setting Up The Blacklist Viaa Front Run

| Issue | Severity | Location | Status |
|---|---|---|---|
| Design Issue | Medium | SmurfCat.sol: 605~607 | Resolved |

### Description

The protocol has implemented a blacklist feature for security requirements. A blacklisted user will not be able to send orreceive tokens. Therefore, before the token is transferred, the protocol checks if the sender or the receiver is blacklisted.However, a malicious user can montior the mempool and if he detects the admin is setting up a blacklist, he can front-run thetransaction and transfer the token to another address to avoid being blacklisted.

### Recommendation

We recommend setting up a blacklist before trading starts. If there is a need to set up a blacklist after trading starts, werecommend using some private relay service such as flashbot to avoid frontrunning.

## MINOR

### SCH-04 Missing Zero Address Validation

| Issue | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Minor | SmurfCat.sol: 611 | Acknowledged |

### Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leadingto unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent lossof those tokens.

```
611 uniswapV2Pair = _uniswapV2Pair;
```

### Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

# INFORMATIONAL

## SCH-05|Function Fails On Valid Inputs

| Issue | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Informational | SmurfCat.sol: 341, 378 | Acknowledged |

### Description

It is expected that invocations of transferFrom(from, dest, amount) succeed and return true if

- the value of amount does not exceed the balance of address from ,
- the value of amount does not exceed the allowance of msg.sender for address from ,
- transferring a value of amount to the address in dest does not lead to an overflow of the recipient's balance, and
- the supplied gas suffices to complete the call.

Nevertheless, the transferFrom function of SmurfCat fails in some of those valid cases.

It is expected that invocations of the form transfer(recipient, amount) succeed and return true if

- The address in recipient is not the zero address,
- amount does not exceed the balance of address msg.sender ,
- transferring amount to the address recipient does not lead to an overflow of the recipient's balance, and
- the supplied gas suffices to complete the call.

Nevertheless, the transfer function of SmurfCat fails in some of those valid cases.

## Recommendation

Ensure that the transferFrom function does not fail on any transfer with valid call parameters as stated above. Thisincludes, but is not limited to, invocations of transferFrom with an amount of zero.

Ensure that the function performs the desired transfer whenever invoked with valid call parameters as listed above. Thisincludes, but is not limited to, invocations of transfer with an amount of zero.

## Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

# Appendix

## Finding Categories

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

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

**Volatile Code**
Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

**Logical Issue**
Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

## About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

### BLOCKCHAIINS

| | |
|---|---|
| Ethereum | Cosmos |
| Eos | Substrate |

### TECH STACK

| | |
|---|---|
| Python | Solidity |
| Rust | c++ |

### CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

August 14th 2024