DeHacker

Code Security Assessment

ICE COIN (ICE)

July 31th, 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	ICE COIN (ICE)
Platform	Ethereum
Website	ice-coin.xyz/
Type	DeFi
Language	Solidity
Codebase	https://etherscan.io/address/0x4b4c8abe2dc873f04068075908f4719554b2cb2c#code

Vulnerability Summary

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

Audit scope

ID	File	SHA256 Checksum
PUD	contracts/PUDGY.sol	411f3bef85010b1fe3eeb8426a7c6fdcf32ae92d0b011c42bccf326fdc3b721d

Findings

ID	Issue	Severity	Status
PUD-01	Initial Token Distribution	Major	Acknowledged
PUD-05	Miscalculation Of Max Holding	Minor	Resolved
PUD-07	Burning Tokens Without Updating The Total Supply In Circulation	Minor	Resolved
PUD-03	Out of Scope Dependency Usage	Informational	Resolved
PUD-08	Missing Zero Address Validation	Informational	Acknowledged

MAJOR

PUD-01 | Initial Token Distribution

Issue	Severity	Location	Status
Centralization	Major	contracts/PUDGY.sol: 1128, 1167	Mitigated

Description

5% of the ICE tokens are sent to the contract owner. 80% of the ICE tokens are used for pool liquidity, and LP tokens aresent to the contract owner after enableTrading() . This is a centralization risk because the owner can distribute tokens orremove liquidity without obtaining the consensus of the community. Any compromise to these addresses may allow a hackerto steal and sell tokens on 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 greater accountability.

MINOR

PUD-05 | Miscalculation Of Max Holding

Issue	Severity	Location	Status
Inconsistency	Minor	contracts/PUDGY.sol: 487, 1250~1253, 1295	Acknowledged

Description

The transaction may be charged fees, so the max holding of the receiver should be the sum of the tokens held and theamount received from the transfer. The fees should not be included in the max holding.

```
1250 require(
1251 amount + balanceOf(to) <= maxWallet,
1252 "Max wallet exceeded"
1253 );
```

- Max holding is checked.

```
1295 amount -= fees;
```

- Receiving amount is adjusted

```
487 _balances[recipient] = _balances[recipient].add(amount);
```

- Balance is increased by the adjusted amount.

Recommendation

We recommend the client check the max holding using the amount received after it has been calculated.

MINOR

PUD-07| Burning Tokens Without Updating The Total Supply In Circulation

Issue	Severity	Location	Status
Logical Issue	Minor	contracts/PUDGY.sol: 1158	Acknowledged

Description

The disableClaim() function transfers the entire balance of tokens held by the address(NFT) to a burn address(address(0xdead)):

```
1158 _transfer(address(NFT), address(0xdead), balanceOf(address(NFT)));
```

This action is intended to prevent further token claims by NFT holders. However, the implemented _transfer() functionused for this purpose does not reduce the total supply of the ICE tokens, as it merely transfers the tokens to the burnaddress without decrementing the total supply counter. This can lead to a discrepancy between the actual circulating supplyand the reported total supply.

Recommendation

Consider replacing it with the _burn() function to burn tokens.

INFORMATIONAL

PUD-03|Redundant And Unreasonable Approval

Issue	Severity	Location	Status
Logical Issue	Informational	contracts/PUDGY.sol: 1166, 1306	Acknowledged

Description

The allowance is approved to the router based on the token swapping amount in the swapTokensForEth function. Therefore,approving the maximum allowance to the router is redundant and unnecessary.

Recommendation

We suggest removing the redundant approval in the constructor and advise against approving the maximum allowance forthe router.

INFORMATIONAL

PUD-08 | The Dex Router Does Not Need To Be Excluded From The Max Wallet Limit

Issue	Severity	Location	Status
Coding Issue	Informational	contracts/PUDGY.sol: 1103	Acknowledged

Description

The contract constructor excludes the DEX router from the maximum wallet holding restriction:

```
1103 excludeFromMaxWallet(address(_uniswapV2Router), true);
```

This exclusion is unnecessary as the DEX router is responsible for transferring tokens from traders directly to the DEX pair,and does not hold tokens itself. The exclusion should be applied to the DEX pair instead, which is the entity that actuallyholds the tokens after trades

Recommendation

Consider excluding the DEX pair from the max wallet holding restriction, rather than the DEX router.

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/

July 31th 2024