DeHacker

Code Security Assessment

Oggy Inu - Audit

Aug 17th, 2023

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
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	Oggy Inu - Audit
Platform	BSC
Website	https://oggyinu.com/
Type	Others
Language	Solidity

Vulnerability Summary

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

Audit scope

ID	File	SHA256 Checksum
OGG	OGGY.sol	2f1c975f71794cd36a6c2894dc185410ace73ae5ec649520349ff76d83396399

Findings

ID	Category	Severity	Status
OGG-05	Centralization /Privilege	Major	Resolved
OGG-01	Volatile Code	Minor	Acknowledged
OGG-02	Mathematical Operations	Minor	Acknowledged
OGG-03	Logical Issue	Informational	Acknowledged

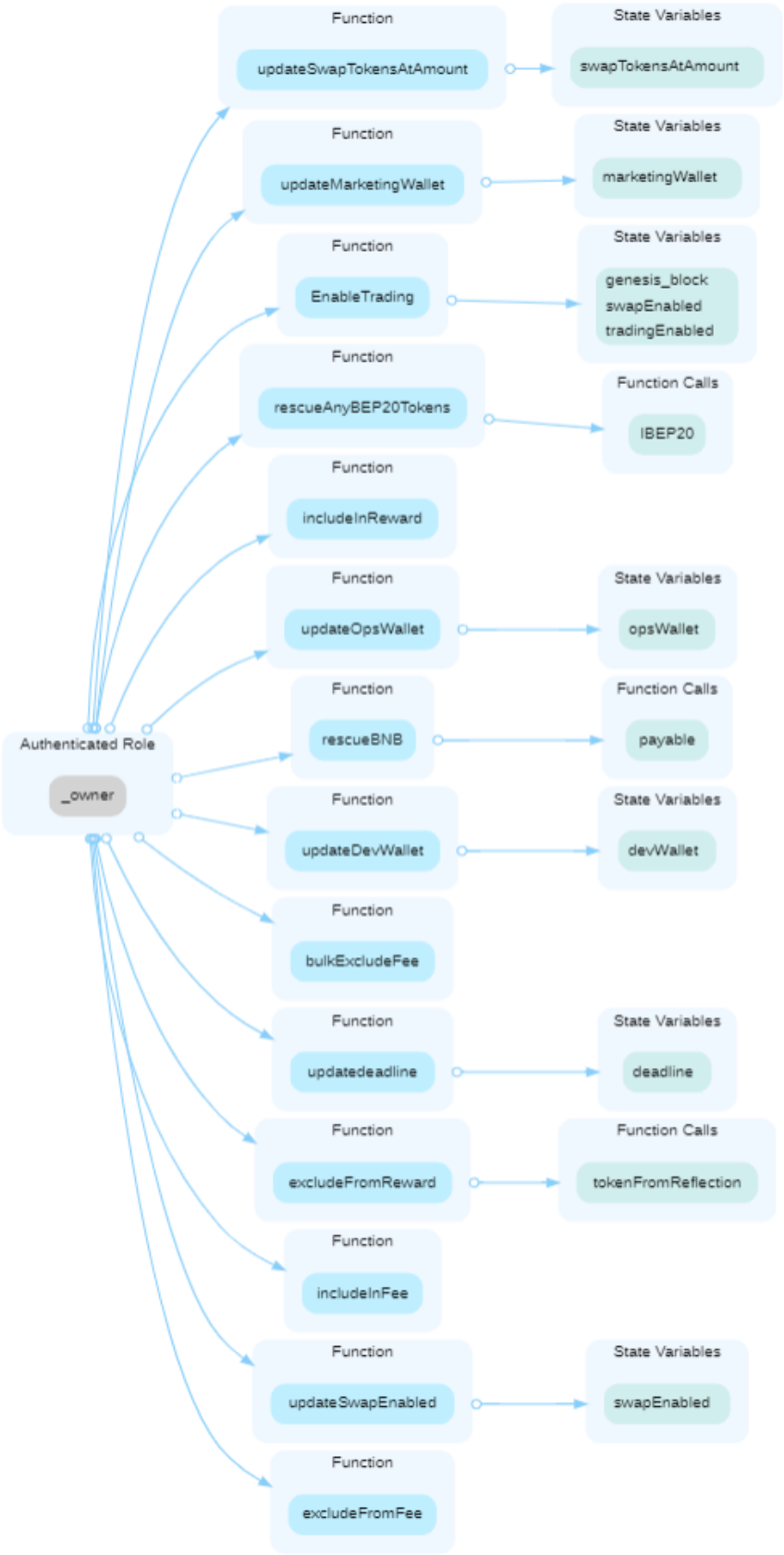
MAJOR

OGG-05|CENTRALIZATION RISKS IN OGGY.SOL

Category	Severity	Location	Status
Centralizatio n /Privilege	Major	OGGY.sol: 56, 60, 311, 318, 331, 340, 353, 357, 698, 704, 709, 715, 720, 725, 730, 736	Resolved

Description

In the contract OGGY the role _owner has authority over the functions shown in the diagram below. Any compromise to the_owner account may allow the hacker to take advantage of this authority and update important parameters such as feepercentage, marketing wallet, exclude/include address from reward/fee, transfer tokens out of the contract etc.


Function
updateSwapTokensAtAmount
State Variables
swapTokensAtAmount
Function
updateMarketingWallet
State Variables
marketingWallet
Function
EnableTrading
State Variables
genesis_block
swapEnabled
tradingEnabled
Function
rescueAnyBEP20Tokens
Function Calls
IBEP20
Function
includeInReward
Function
updateOpsWallet
State Variables
opsWallet
Function
rescueBNB
Function Calls
payable
Authenticated Role
_owner
Function
updateDevWallet
State Variables
devWallet
Function
bulkExcludeFee
Function
updatedeadline
State Variables
deadline
Function
excludeFromReward
Function Calls
tokenFromReflection
Function
includeInFee
Function
updateSwapEnabled
State Variables
swapEnabled
Function
excludeFromFee

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, 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 (,) 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 keycompromised;
AND
A medium/blog link for sharing the timelock contract and multi-signers addresses information with the publicaudience.

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 publicaudience.

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.

MINOR

OGG-01|THIRD PARTY DEPENDENCY

Category	Severity	Location	Status
Volatile Code	Minor	OGGY.sol: 133	Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treatsthird party entities as black boxes and assume their functional correctness. However, in the real world, third parties can becompromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severeimpacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
133     IRouter public router;
```

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the
team to constantlymonitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

MINOR

OGG-02|DIVIDE BEFORE MULTIPLY

Category	Severity	Location	Status
Mathematical Operations	Minor	OGGY.sol: 641 , 642	Acknowledged

Description

Performing integer division before multiplication truncates the low bits, losing the precision of calculation.

```
641         uint256 unitBalance = deltaBalance / (denominator - temp.liquidity);
```

```
642         uint256 bnbToAddLiquidityWith = unitBalance * temp.liquidity;
```

Recommendation

We recommend applying multiplication before division to avoid loss of precision.

INFORMATIONAL

OGG-03|VARIABLE _rOwned[account] NOT UPDATED INFUNCTION includeInReward()

Category	Severity	Location	Status
Logical Issue	Informational	OGGY.sol: 340	Acknowledged

Description

Variable _rOwned[account] is not updated in the function includeInReward() , which will make the accounts includedsiphon off the tokens out of the balances of all token holders . The Rate was higher at the moment of theexcludeFromReward(account) call, so the _ rOwned[account] / _tOwned[account] ratio is bigger than expected foraccounts included in the reward.

Recommendation

We advise the team to properly update _rOwned[account] before setting _tOwned[account] to 0 to keep the rateunchanged. The correct solution should have the following characteristics:
Do not change the rate.
Do not change the account's token balance.
Do not change the total supply.

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/

Aug 2023