Security Assessment

opsNft

Jun 24th, 2021

Table of Contents

Summary

This report has been prepared for opsNft smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	opsNft
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/AceLuodan/opsNft
Commit	1efe4f9f1b8231e8d0d77012d0a8c6849522c56e cb2057727d29e52263a4cc3245a5afc752668bdb

Audit Summary

Delivery Date	Jun 24, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

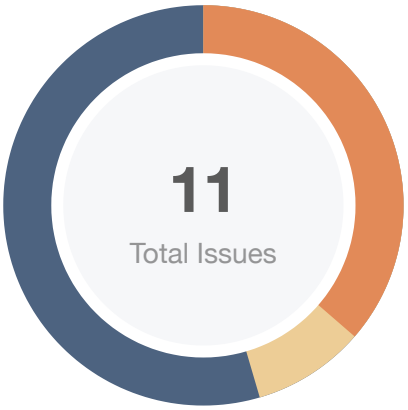
Total Issues	11
Critical	0
Major	4
Medium	0
Minor	1
Informational	6
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
OEN	OpsErc20.sol	10d02eef10e7304cd87a188fc10502e2b87e8d8be562bb66d6106b3faccc47e3
OHN	OpsHolder.sol	219892649c41fcb2b1c3024819d5ecbeade47051cb1e3a7022925008fcc7a2d5
OLR	OpsLpRewards.sol	2f7ea095130dbab366618133a0e8e7a502cbe14541330ba71d12d5f5e7cf3275
ONN	OpsNft.sol	dd26c99fa4fda9ead9c941f53ad025582eae197fc6c390ce4e28baae0ba47a83
ONC	OpsNft1155.sol	f89b60a51536ed3b92b0604a4725c4e872a1c52ad88bb0e666466fc54a55c3e6
TNN	Timelock.sol	75ceeca6b58037a53cb62f1e28834b6080ed14c6c8fc00c24647ae32112913ca
ERC	interface/ERC1155.sol	63913120c55625bf5e3461708da7e15b27a91529a5eb9c3e17134e51269b954f
ERH	interface/ERC1155Holder.sol	45d80265e6f9f73d072e098bfb3eb0435ce40d5ea689bde2c9bc9038321fce2f
ERR	interface/ERC1155Receiver.sol	afeafada0aef591298d19b1fd68274a70e97367c3b6a537faa8f0dd50cfec963
ERN	interface/ERC165.sol	e6a3cba0775773bd92c8de6ac14d0614ca443ad63464a4e0241ca345940ea973
ERK	interface/ERC20.sol	7a21f1290b518f4e9a77c6e524eca8558fdbe88c63894b5f5ce6649bbb34d642
ERB	interface/ERC20Burnable.sol	a835976ddf21af0edce231175323014366996a96f220701e131f1ac571f4ac57
ERP	interface/ERC20Capped.sol	f19e7154d8836089c2295de1f2dea3e816f3dcb95be912786dacd7bf0c5fae5f
IER	interface/IERC1155.sol	8a0e9c841e7f6bf6f02d8013a7d0cf62306a67729557626b6190230688edf703
IEC	interface/IERC1155MetadataURI.sol	e46b551826c4497fe033f841b920a21a6be33543c0f9fedc3d2f9bcdee96ee14
IEN	interface/IERC1155Receiver.sol	e1954e649815d883afaa820808accebf2068018bf9049401a3b67afa725eac45
IEK	interface/IERC165.sol	24d63fd063d0d9e954ce1a039404b4c01d2141f787143bbd3d5090a0220a2bcc
IEP	interface/IERC20.sol	0573c2961569aa4906845d0cd428b5b7394956170054ceeaa8f8af96cd44875c
IRC	interface/IERC721.sol	1802b20515694649f4e98bca15248b1caefcb5bf454ac50f99b8bef353fa3833
IEE	interface/IERC721Enumerable.sol	da6fa0593fd96281d88df725727540d0c61551ed756a31a2ef6e1e8ccfbbe59d
IEM	interface/IERC721Metadata.sol	17a75a430e00aa592ec076cecb7c1fee37b4b21c10cec9b84f57faac13fb3cb5
IRR	interface/IERC721Receiver.sol	7e3d89b564e70918bc4e71e8346271f90dc3359d65b542baf24ce4de4e73d0a8

ID	file	SHA256 Checksum
ANN	utils/Address.sol	11ad5e3e21434e00c4ceba1f5a977b7a68bdd7d16b849276ce4ff4495129eec7
CNN	utils/Context.sol	9a3d1e5be0f0ace13e2d9aa1d0a1c3a6574983983ad5de94fc412f878bf7fe89
CNC	utils/Counters.sol	21bd8d48434f129aaaf41d5400ee65a2f36d221c497b74d126acd232da59c236
EMN	utils/EnumerableMap.sol	26c7ec2df617e9420a3782d911dc6c339e83b02eac442de4c3c4bbbd18fe3273
ESN	utils/EnumerableSet.sol	c8b73a000476872a00f6153d66be31a4a99b7565068f05336129748bfad704ea
ONK	utils/Ownable.sol	d4b4b5ee57093cacd5b20058bba8efe7d4158288032c0a2483dc1a0b7efbb328
PNN	utils/Pausable.sol	20fdf7a18aabd402e481e65a1b18cb4b43bda99b4849b4e2c5b3dcdd01121dd0
RNN	utils/Roles.sol	072661ea14540b092fcd5a34da1705c494cdd16c16db18f3aad9ee69e0127a02
SER	utils/SafeERC20.sol	e55fdae48e01d02f76d6232de63635a027a475a11ba17661f606906fad99e3bc
SMN	utils/SafeMath.sol	4a04d0a20a19e3ef1dcabae9cad9ba006430a4e7eec4d9b519db87999722c98a
SNN	utils/Strings.sol	129fbdf69a2318aeb93dc8425ab57870149d27b3bc1d7bebf6724dc314e87890

Findings



Critical	**0** (0.00%)
Major	**4** (36.36%)
Medium	**0** (0.00%)
Minor	**1** (9.09%)
Informational	**6** (54.55%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
OHN-01	Redundant Statement	Gas Optimization	Informational	Resolved
OHN-02	Centralization Risk	**Centralization / Privilege**	**Major**	**Partially Resolved**
OLR-01	Unhandled Return Value	Volatile Code	Informational	Resolved
OLR-02	add() Function Not Restricted	Volatile Code	Major	Resolved
OLR-03	Centralized Control of Bonus Multiplier	Logical Issue	Informational	Resolved
OLR-04	Missing Emit Events	Gas Optimization	Informational	Resolved
OLR-05	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Resolved
OLR-06	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Resolved
OLR-07	Centralization Risk	**Centralization / Privilege**	**Major**	**Resolved**
ONC-01	Code Simplification	Gas Optimization	Informational	Resolved
ONC-02	Incorrect Token Amount Update	Logical Issue	Major	Resolved

OHN-01 | Redundant Statement

Category	Severity	Location	Status
Gas Optimization	Informational	OpsHolder.sol: 76	Resolved

Description

`require(governor_ != address(0), "governor_ zero address");` has already been checked in function `_transferGovernorship()`

Recommendation

We advise the client to remove the `require(governor_ != address(0), "governor_ zero address");` in constructor of the contract `OpsHolder`

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

OHN-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	OpsHolder.sol: 116, 144, 159	**Partially Resolved**

Description

The `governance` role of the contract `OpsHolder` can transfer any amount of fungible tokens or any non-fungible token by calling the following to functions:

- `transferERC20()`
- `transferETH()`
- `transferERC721()`
- `transferERC1155()`

Such privileges might expose the project to economic exploits if the attackers manage to obtain an account granted with the `governance` role.

Recommendation

We advise the client to handle the accounts granted with `governance` roles carefully to avoid any potential hack. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

Alleviation

`[CertiK]`: Same issue is found in functions `transferBatchERC20()`, `transferBatchIdERC1155()`, `transferBatchERC1155()` which are added in the contract `OpsHolder.sol` in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

`[opsNFT]`: Tmelock contract will be deployed to reduce the centralization of the project

OLR-01 | Unhandled Return Value

Category	Severity	Location	Status
Volatile Code	Informational	OpsLpRewards.sol: 291, 293	Resolved

Description

Return value of function `transfer` based on 'OpsLpRewards.sol' is ignored in function `safeOpsTransfer`.

Recommendation

We advise the client to handle the return value of `transfer` to check if it's implementation is executed without any error.

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

OLR-02 | add() Function Not Restricted

Category	Severity	Location	Status
Volatile Code	Major	OpsLpRewards.sol: 129	Resolved

Description

The total amount of reward in function `updatePool()` will be incorrectly calculated if the same LP token is added into the pool more than once in function `add()`. However, the code is not reflected in the comment behaviors as there isn't any valid restriction on preventing this issue. The current implementation is relying on the trust of the owner to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

Recommendation

Detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using a mapping of `addresses` -> `booleans`, which can restricted the same address being added twice.

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

OLR-03 | Centralized Control of Bonus Multiplier

Category	Severity	Location	Status
Logical Issue	Informational	OpsLpRewards.sol: 115	Resolved

Description

The function can alter the `BONUS_MULTIPLIER` variable and consequently the output of which is directly utilized for the minting of new tokens.

Recommendation

This is the intended functionality of the protocol, however, users should be aware of this functionality.

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue by adding an emit event in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

OLR-04 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	OpsLpRewards.sol: 115, 299, 159	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `fee()`
- `setMigrator()`
- `updateMultiplier()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

```
event SetFee(address indexed user, address indexed _feeAddr);

function fee(address _feeAddr) public {
    require(msg.sender == feeAddr, "fee: wut?");
    feeAddr = _feeAddr;
    emit SetFee(msg.sender, _feeAddr);
}
```

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

OLR-05 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	OpsLpRewards.sol: 145, 164, 181, 204, 229, 251, 273	Resolved

Description

There's no sanity check to validate if a pool is existing.

Recommendation

We advise the client to adopt following modifier `validatePoolByPid` to functions `set()`, `migrate()`, `deposit()`, `withdraw()`, `emergencyWithdraw()`, `pendingOps()` and `updatePool()`.

```
1 modifier validatePoolByPid(uint256 _pid) {
2     require (_pid < poolInfo.length , "Pool does not exist") ;
3     _;
4 }
```

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

OLR-06 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	OpsLpRewards.sol: 251, 229	Resolved

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, the amount inconsistency will occur and the transaction may fail due to the validation checks.

Recommendation

We advise the client to regulate the set of LP tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

OLR-07 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	OpsLpRewards.sol: 109, 221, 299	**Resolved**

Description

With the modifier `onlyOwner`, the owner has authority to call below functions that might modify the sensitive states of the `OpsLpRewards` contract:

- `setFeeRatio()`: `owner` can modify the value of `feeRatio` to control the percentage of `opsReward` that will be sent to `feeAddr`.
- `fee()`: `owner` can modify the `feeAddr` to any arbitrary address.

Recommendation

We advise the client to handle the accounts granted with `owner` role carefully to avoid any potential hack. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

ONC-01 | Code Simplification

Category	Severity	Location	Status
Gas Optimization	Informational	OpsNft1155.sol: 125	Resolved

Description

As `_id` of each ERC1155 token is newly created in function `createBatch()` and `tokenSupply` of each `_id` is therefore 0, the following code snippet can be simplified.

```
125  tokenSupply[_id] = tokenSupply[_id].add(quantity);
```

Recommendation

We advise the client to consider simplifying the code snippet to save gas with the following:

```
125  tokenSupply[_id] = quantity; //quantity works as initial total supply for the
specific `_id` ERC1155 token
```

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

ONC-02 | Incorrect Token Amount Update

Category	Severity	Location	Status
Logical Issue	Major	OpsNft1155.sol: 283	Resolved

Description

When burning an amount of specific token by calling function `burn()`, the number of the burnt tokens should be deducted from the total supply of this specific token in L283.

Recommendation

We advise the client to update L283 with following code snippet:

```
283  tokenSupply[id] = tokenSupply[id].sub(value);
```

Alleviation

`[opsNFT]`: The client heeded the advice and resolved the issue in the commit cb2057727d29e52263a4cc3245a5afc752668bdb

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.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

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.

Volatile Code

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

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.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.