Security Assessment

Superlauncher

Nov 15th, 2021

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- CSC-01 : Centralization Risk
- CSC-02 : Lack of Address Validation
- CSC-03 : Lack of Access Control for Initialization
- CSC-04 : Potential Reentrancy Attack
- CSC-05 : Divide Before Multiply
- CSC-06 : Lack of Zero Address Validation
- CSC-07 : Missing Emit Events
- FSC-01 : Centralization Risk
- FSC-02 : Missing Emit Events
- FSC-03 : Duplicate `Campaign` and `NFT` may Generated
- MSC-01 : Centralization Risk
- MSC-02 : Potential Front-Running Risk
- MSC-03 : Unbounded Loop
- MSC-04 : Missing Emit Events
- MSC-05 : Lack of Zero Address Validation
- SDN-01 : Privilege Role Declared As A Private Variable
- SDN-02 : Centralization Risk
- SDN-03 : Potential Reentrancy Attack
- SDN-04 : Remained Information After `combine`
- SDN-05 : Uninitialized Local Variable
- SDN-06 : Lack of Zero Address Validation

Appendix

Disclaimer

About

Summary

This report has been prepared for Superlauncher to discover issues and vulnerabilities in the source code of the Superlauncher project 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;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Superlauncher
Platform	other
Language	Solidity
Codebase	https://github.com/SuperLauncher/v2-OTC
Commit	e215059bd33742f2e4567498b29bdbefd306abfa e9a93b62938f8e746240ab486ef2f55b8e16d64d 7812d7ab36120ce804ef3cac504817f5fac5418f

Audit Summary

Delivery Date	Nov 15, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	Campaign, Factory, Marketplace, SuperDeedNFT, IEmergency, ISuperDeedNFT

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	5	0	0	4	0	1
Medium	4	0	0	1	0	3
Minor	2	0	0	1	0	1
Informational	10	0	0	4	0	6
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
IES	contracts/interfaces/IEmergency.sol	3ea0a803157d3a7693984d9f6c4d728777e0cc27af2702e04d130a35bd5bbbb7
ISD	contracts/interfaces/ISuperDeedNFT.sol	005b6e7b7a0532092abd787838b62e15088bb384e419c909fcc212929d082c8a
CSC	contracts/Campaign.sol	1b7d584ff3bfaa26a686b24b126eed92159698dc8d4870196296d5161a3998be
FSC	contracts/Factory.sol	19d32fffb7e9f6e573bf79796bd59cb73c7b857736fcaed68d1d59cfd97ea8e5
MSC	contracts/Marketplace.sol	3d32050b6d99a95c91aa503a118fc376ee331810bd7d3a8854d7da34ada6d49f
SDN	contracts/SuperDeedNFT.sol	610b45c105d2002fefd1c217b2ecaade9867e09f8a5a49fc6408d842b2a3c2ad

Overview

External Dependencies

The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets.

There are a few depending injection contracts or addresses in the current project:

- Contract `Campaign` will setup `svLaunchAddress`, `currencyAddress`, `deedNftAddress` and `campaignOwner`.
- Contract `Factory` will setup `svLaunchAddress`, `daoFeeAddress` and `deployerAddress`.
- Contract `Marketplace` will setup `daoFeeAddress`.
- Contract `SuperDeedNFT` will setup `_minter`, `_distributor` and `daoFeeAddress`.

In addition, the contract is serving as the underlying entity to interact with third-party contracts and interfaces:

- `OpenZeppelin` library
- Contracts fulfill `ISuperDeedNFT`, `IEmergency`, `ERC20`, `IERC721Upgradeable` and `IERC20Upgradeable` interfaces.

We assume these contracts or addresses are valid and non-vulnerable actors and implement proper logic to collaborate with the current project.

Privileged Functions

The contract `Campaign` contains the following privileged functions that are restricted by role: `factory`. They are used to modify the contract configurations and address attributes. The `factory` role is specified by the `onlyFactory` modifier. We grouped these functions below.

- `Campaign.initialize(address,address,address,address)`
- `Campaign.setup(uint256[4],uint256[3],uint256[5],uint256[2])`
- `Campaign.setCancelled()`
- `Campaign.daoMultiSigEmergencyWithdraw(address to, address tokenAddress, uint amount)`

The contract `SuperDeedNFT` contains the following privileged functions that are restricted by roles: `_minter` and `_distributor`. They are used to modify the contract configurations and address attributes. These roles are specified by the `onlyMinter` and `onlyDistributor` modifiers. We grouped these functions below.

The role `_minter`:

- `SuperDeedNFT.mint(address, uint)`
- `SuperDeedNFT.setTotalRaise(uint, uint)`

The role `_distributor`:

- `SuperDeedNFT.setAssetInfo(string)`
- `SuperDeedNFT.setDaoFee(uint)`
- `SuperDeedNFT.setAssetAddress(address)`
- `SuperDeedNFT.distributeTokens(uint)`

The contract `Marketplace` contains the following privileged functions that are restricted by role: `_owner`. They are used to modify the contract configurations and address attributes. The `_owner` role is specified by the `onlyOwner` modifier. We grouped these functions below.

- `Marketplace.setupCurrency(address, address, address, address)`
- `Marketplace.setFee(uint)`
- `Marketplace.setAllowedNft(address, bool)`
- `Marketplace.cancelListing(address, uint)`
- `Marketplace.daoMultiSigEmergencyWithdraw(address, address, uint)`
- `OwnableUpgradeable.renounceOwnership()`
- `OwnableUpgradeable.transferOwnership(address)`

The contract `Factory` contains the following privileged functions that are restricted by roles: `_owner` and `deployerAddress`. They are used to modify the contract configurations and address attributes. These roles are specified by the `onlyOwner` and `onlyDeployer` modifiers. We grouped these functions below.

The role `_owner`:

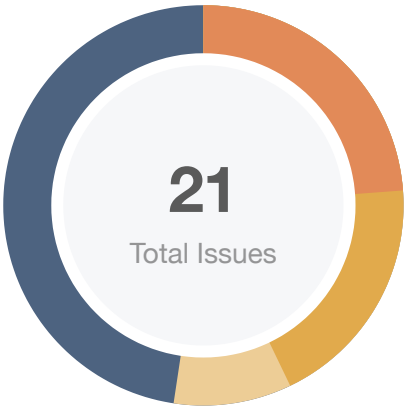
- `Factory.setDeployer(address)`
- `Factory.setDaoFeeAddress(address)`
- `Factory.daoMultiSigEmergencyWithdraw(address, address, uint)`
- `Ownable.renounceOwnership()`
- `Ownable.transferOwnership(address)`

The role `deployerAddress`:

- `Factory.createCampaign(address,address,address,string)`
- `Factory.setupCampaign(uint256,address,uint256[4],uint256[3],uint256[5],uint256[2])`
- `Factory.cancelCampaign(uint256, address)`

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the Timelock contract.

Findings



Critical	**0** (0.00%)
Major	**5** (23.81%)
Medium	**4** (19.05%)
Minor	**2** (9.52%)
Informational	**10** (47.62%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CSC-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
CSC-02	Lack of Address Validation	Volatile Code	Medium	Acknowledged
CSC-03	Lack of Access Control for Initialization	Control Flow	Medium	Resolved
CSC-04	Potential Reentrancy Attack	Logical Issue	Medium	Resolved
CSC-05	Divide Before Multiply	Mathematical Operations	Informational	Acknowledged
CSC-06	Lack of Zero Address Validation	Volatile Code	Informational	Acknowledged
CSC-07	Missing Emit Events	Coding Style	Informational	Resolved
FSC-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
FSC-02	Missing Emit Events	Coding Style	Informational	Resolved
FSC-03	Duplicate `Campaign` and `NFT` may Generated	Logical Issue	Informational	Resolved
MSC-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
MSC-02	Potential Front-Running Risk	Volatile Code	Minor	Acknowledged
MSC-03	Unbounded Loop	Logical Issue	Informational	Acknowledged

ID	Title	Category	Severity	Status
MSC-04	Missing Emit Events	Coding Style	Informational	Resolved
MSC-05	Lack of Zero Address Validation	Volatile Code	Informational	Resolved
SDN-01	Privilege Role Declared As A Private Variable	**Centralization / Privilege**	**Major**	Resolved
SDN-02	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
SDN-03	Potential Reentrancy Attack	Logical Issue	Medium	Resolved
SDN-04	Remained Information After `combine`	Logical Issue	Minor	Resolved
SDN-05	Uninitialized Local Variable	Volatile Code	Informational	Acknowledged
SDN-06	Lack of Zero Address Validation	Volatile Code	Informational	Resolved

CSC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/Superlauncher/contracts/Campaign.sol (f06ee54): 95, 108, 230, 269	Acknowledged

Description

In the contract `Campaign`, the role `factory` has the authority over the following function:

- `Campaign.initialize(address,address,address,address)` will initialize the contract.
- `Campaign.setup(uint256[4],uint256[3],uint256[5],uint256[2])` will setup `Campaign` states.
- `Campaign.setCancelled()` will cancel `Campaign`.
- `Campaign.daoMultiSigEmergencyWithdraw(address to, address tokenAddress, uint amount)` will withdraw specific tokens from `Campaign` address.

Any compromise to the `factory` account may allow the hacker to take advantage of this and change the `Campaign` states. Although in general workflow, the `factory` role may refer to the `Factory` contract, in auditing, we consider it as a black box. Also, these functions can be triggered by the privileged role `_owner` in `Factory` contract.

Recommendation

We advise the client to carefully manage the `factory` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

(**Superlauncher Team Response**)

The factory role will be using Gnosis Safe MultiSig 3/5 wallet address.

CSC-02 | Lack of Address Validation

Category	Severity	Location	Status
Volatile Code	Medium	projects/Superlauncher/contracts/Campaign.sol (f06ee54): 90~92, 95	Acknowledged

Description

Although `Campaign` contract is intended deployed by the `Factory` contract, it can also be deployed by the EOA in the current setting. If the contract is deployed by an EOA, the contract can be initialized multiple times by the `initialize` function, which may cause a centralization risk.

Recommendation

We recommend the team add an address check in constructor for `msg.sender` to make sure the `factory` is a contract.

Alleviation

(**Superlauncher Team Response**)

The SuperLauncher frontend website will get a list of official campaigns from `factory.allCamapigns[n]`. Any unofficial `Campaign` contracts created by EOA or other smart contract will not get registered into factory smart contract.

CSC-03 | Lack of Access Control for Initialization

Category	Severity	Location	Status
Control Flow	Medium	projects/Superlauncher/contracts/Campaign.sol (f06ee54): 95	Resolved

Description

The current initialize function is not restricted. The initialization can be invoked multiple time, this is might be dangerous operation if the there any incident happened which the attacker take the controlled, they might able to replace all the contracts with the malicious one.

Recommendation

We advise that the function should be more restricted in term of invocation and sanity check for inputs.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d.

CSC-04 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logical Issue	Medium	projects/Superlauncher/contracts/Campaign.sol (f06ee54): 134, 158, 177, 196	Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

Example:

- buyFund()
- finishup()
- claimNFT()
- refund()

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d.

CSC-05 | Divide Before Multiply

Category	Severity	Location	Status
Mathematical Operations	Informational	projects/Superlauncher/contracts/Campaign.sol (f06ee54): 305	Acknowledged

Description

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

Recommendation

Consider ordering multiplication before division. For example:

```
305     return (_minPrivateBuyLimit, alloc * VALUE_10 / VALUE_10 );
```

Alleviation

(**Superlauncher Team Response**)

The purpose of doing division first and then multiple, is to round off to the nearest 10 BUSD. This way, the user will get a nice round number allocation.

CSC-06 | Lack of Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Informational	projects/Superlauncher/contracts/Campaign.sol (f06ee54): 102~105	Acknowledged

Description

In the function `initialize`, the input variables `svLaunch`, `campOwner`, `deed` and `currencyToken` should not be zero addresses.

Recommendation

We advise the client to check that the aforementioned variables are not zero address.

Alleviation

(**Superlauncher Team Response**)

The factory pattern is used to deploy `Campaign`. Any `Campaign` that is deployed independently of the `Factory` will not be registered in the official factory contract instance. Our website will only load registered campaign from `Factory`. In the factory contract, the `deployerAddress`, `svLaunchAddress` & `daoFeeAddress` are verified to be non-zero.

In the `Factory.creatCampaign()`, `campaignOwner`, currency are checked to make sure it is non-zero. As mentioned in CSC-02, any `Campaign` contract created by EOA will not be registered in the factory contract and will not be loaded by website.

CSC-07 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	projects/Superlauncher/contracts/Campaign.sol (f06ee54): 108, 230	Resolved

Description

The function `setup` affects the status of sensitive variables for `Campaign` should emit events as notifications. And the function `setCancelled` affects the status of `Campaign` should also emit events as notifications.

Recommendation

We recommend the team consider emitting an event in `setup` and `setCancelled` for the update of the sensitive variables.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d.

FSC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/Superlauncher/contracts/Factory.sol (f06ee54): 41, 46, 116, 51, 83, 108	Acknowledged

Description

In the contract `Factory`, the role `_owner` has the authority over the following function:

- `Factory.setDeployer(address)` will set new deployer.
- `Factory.setDaoFeeAddress(address)` will set new fee address.
- `Factory.daoMultiSigEmergencyWithdraw(address, address, uint)` will withdraw tokens from one `Campaign` to owner.
- `Ownable.renounceOwnership()` will transfer ownership to `address(0)`.
- `Ownable.transferOwnership(address)` will transfer ownership to new owner.

The role `deployerAddress` has the authority over the following function:

- `Factory.createCampaign(address,address,address,string)` will create `Campaign` and `SuperDeedNFT` contracts, also initialize `Campaign` contract.
- `Factory.setupCampaign(uint256,address,uint256[4],uint256[3],uint256[5],uint256[2])` will setup states in `Campaign` contract.
- `Factory.cancelCampaign(uint256, address)` will cancel `Campaign`.

Any compromise to the `_owner` and `deployerAddress` accounts may allow the hacker to take advantage of this and violate the functionality of contracts.

Recommendation

We advise the client to carefully manage the `_owner` and `deployerAddress` accounts' private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

(**Superlauncher Team Response**)

Factory owner address & deployer address are Gnosis Safe Multisig wallet.

FSC-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	projects/Superlauncher/contracts/Factory.sol (f06ee54): 41, 46	Resolved

Description

The function `setDeployer` and `setDaoFeeAddress` that affect the status of sensitive variable `deployerAddress` and `daoFeeAddress` should be able to emit events as notifications.

Recommendation

We recommend team consider adding events and emit it in the aforementioned functions.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d.

FSC-03 | Duplicate `Campaign` and `NFT` may Generated

Category	Severity	Location	Status
Logical Issue	Informational	projects/Superlauncher/contracts/Factory.sol (f06ee54): 61~66	Resolved

Description

In function `createCampaign`, it will deploy `Campaign` contract and `SuperDeedNFT` contract, although it may not override the existing contract, it still can deploy the contract with the same information, especially the `SuperDeedNFT` contract.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d by making sure that the salt for SuperDeedNft and Campaign will be unique. The team uses “count” as part of the salt construction since the count increases for every campaign created.

(**Superlauncher Team Note**)

It is a valid workflow to have multiple campaigns (and NFT) for a single project. For example:

Campaign 1: MetaWar SEED (500k allocation)

Campaign 2: MetaWar SEED (100k allocation)

MSC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/Superlauncher/contracts/Marketplace.sol (f06ee54): 88, 96, 105, 174, 324	Acknowledged

Description

In the contract `Marketplace`, the role `_owner` has the authority over the following function:

- `Marketplace.setupCurrency(address, address, address, address)` will setup accepted currency.
- `Marketplace.setFee(uint)` will setup fee percent.
- `Marketplace.setAllowedNft(address, bool)` will set allowed NFT list.
- `Marketplace.cancelListing(address, uint)` will call function `_removeListing` to cancel listing.
- `Marketplace.daoMultiSigEmergencyWithdraw(address, address, uint)` will withdraw token or BNB to an address.
- `OwnableUpgradeable.renounceOwnership()` will transfer ownership to `address(0)`.
- `OwnableUpgradeable.transferOwnership(address)` will transfer ownership to new owner.

Any compromise to the `_owner` account may allow the hacker to take advantage of this and violate contract states.

Recommendation

We advise the client to carefully manage the `_owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

(**Superlauncher Team Response**)

Marketplace owner is a Gnosis Safe MultiSig wallet

MSC-02 | Potential Front-Running Risk

Category	Severity	Location	Status
Volatile Code	Minor	projects/Superlauncher/contracts/Marketplace.sol (f06ee54): 80~86	Acknowledged

Description

Malicious hackers may observe the pending transaction which will execute the `initialize` function, and launch a similar transaction but with malicious states info.

Recommendation

We recommend the team monitor the contract states closely after deployment or consider adding access restrictions to `initialize` function.

Alleviation

(**Superlauncher Team Response**)

The initialize() function is marked with `initializer` modifier. We will call `initialize()` asap and monitor the blockchain tx carefully.

MSC-03 | Unbounded Loop

Category	Severity	Location	Status
Logical Issue	Informational	projects/Superlauncher/contracts/Marketplace.sol (f06ee54): 133, 160, 244, 303	Acknowledged

Description

The `for` loop within functions `getAllowedNfts`, `getListedNftIds`, `getListingsForNFT` and `getListingIDsBySeller` take variable `returnCount` which is generated by the function `_getEndIndex`, as the maximal iteration times. If the size of the array or set is very large, it could exceed the gas limit to execute the functions. In this case, the contract might suffer from DoS (Denial of Service) situation.

Recommendation

We recommend the team review the design and ensure this would not cause loss to the project.

Alleviation

(**Superlauncher Team Response**)

The function `getAllowedNfts()` take a start-index and a count. If we are dealing with a huge value, then we can split it into multiple calls instead of a single call.

MSC-04 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	projects/Superlauncher/contracts/Marketplace.sol (f06ee54): 96~99	Resolved

Description

The function `setFee` that affects the status of sensitive variable `feePcnt` should be able to emit events as notifications.

Recommendation

We recommend team consider adding events for the `setFee` update and emit it in the function.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d.

MSC-05 | Lack of Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Informational	projects/Superlauncher/contracts/Marketplace.sol (f06ee54): 80	Resolved

Description

In the constructor, the input variable `feeAddress` should not be zero addresses.

Recommendation

We advise the client to check that the aforementioned variable is not zero address.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d.

SDN-01 | Privilege Role Declared As A Private Variable

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/Superlauncher/contracts/SuperDeedNFT.sol (f06ee54): 21~23	Resolved

Description

In contract `SuperDeedNFT`, privileged role `_minter` and `_distributor` are set as `private`. Users have no easy way to find out the addresses. It leads to concerns about centralization risk and project transparency and gives a hard time for users to verify if privilege roles have been transferred to a timelock contract or not.

```
21      address private _minter;
22      address private _distributor;
```

Recommendation

We recommend the project provide the information on addresses behinds the owner roles to the community in a public way for improving the project's transparency.

Alleviation

The team heeded our advice and resolved this issue in the commit [e9a93b62938f8e746240ab486ef2f55b8e16d64d](e9a93b62938f8e746240ab486ef2f55b8e16d64d).

SDN-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/Superlauncher/contracts/SuperDeedNFT.sol (f06ee54): 96, 101, 108, 112, 121, 128	Acknowledged

Description

In the contract `SuperDeedNFT`, the role `_minter` has the authority over the following function:

- `SuperDeedNFT.mint(address, uint)` will mint NFT(assign weights) to an address.
- `SuperDeedNFT.setTotalRaise(uint, uint)` will set `totalRaise`.

the role `_distributor` has the authority over the following function:

- `SuperDeedNFT.setAssetInfo(string)` will set `asset`'s `symbol`.
- `SuperDeedNFT.setDaoFee(uint)` will set fee percent.
- `SuperDeedNFT.setAssetAddress(address)` will set `asset`'s address.
- `SuperDeedNFT.distributeTokens(uint)` will distribute tokens.

Any compromise to the `_minter` and `_distributor` accounts may allow the hacker to take advantage of this and violate contract states.

Recommendation

We advise the client to carefully manage `_minter` and `_distributor` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

(**Superlauncher Team Response**)

`_minter` is set once (in the constructor), to be the address of the Campaign smart contract. Once set, it is not changeable. The only minting is in the `Campaign` contract, when the user claim their NFT. The only `setTotalRaise()` call is in the `Campaign` contract, when `finishUp()` is called (once only). For `_distributor`, it is a Gnosis Safe MultiSig wallet.

SDN-03 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logical Issue	Medium	projects/Superlauncher/contracts/SuperDeedNFT.sol (f06ee54): 172~189	Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

In function `claim`, it will generate an external call by :

```
181         ERC20(asset.tokenAddress).safeTransfer(msg.sender, amt);
```

and update state variable later by the following code:

```
183         NftInfo storage item = _nftInfoMap[id];
184         item.nextClaimIndex = indexTo + 1;
185         item.claimedPtr = totalTokensReleased;
```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d.

SDN-04 | Remained Information After `combine`

Category	Severity	Location	Status
Logical Issue	Minor	projects/Superlauncher/contracts/SuperDeedNFT.sol (f06ee54): 224~232	Resolved

Description

In function `combine`, although `id2` was burned after the combination, but the `_nftInfoMap[id2]` still exist and valid, functions like `getClaimable`, `getItemInfo`, `weightOf` can still return original value for burned `id2`.

```
224     function combine(uint id1, uint id2) external {
225         require(ownerOf(id1) == msg.sender && ownerOf(id2) == msg.sender, "Not
owner");
226         require(_nftInfoMap[id1].nextClaimIndex == _nftInfoMap[id2].nextClaimIndex,
"Please claim before combining");
227
228         _nftInfoMap[id1].weight += _nftInfoMap[id2].weight;
229
230         // Burn NFT 2
231         _burn(id2);
232     }
```

Recommendation

We recommend the team reconsider the combine/burned logic and handle such information correctly after the burned or combined.

Alleviation

The team heeded our advice and resolved this issue in the commit [e9a93b62938f8e746240ab486ef2f55b8e16d64d](e9a93b62938f8e746240ab486ef2f55b8e16d64d).

SDN-05 | Uninitialized Local Variable

Category	Severity	Location	Status
Volatile Code	Informational	projects/Superlauncher/contracts/SuperDeedNFT.sol (f06ee54): 137	Acknowledged

Description

Local variable `fee` is only and first initialized when `daoFeePcnt > 0`.

However, the value of `fee` will be used in the event emitting of `DistributeTokens`, no matter if `fee` is set or not.

Recommendation

We recommend initializing the local variable to an acceptable default value.

Alleviation

(**Superlauncher Team Response**)

The default value of `fee` is 0. If the `daoFeePcnt is > 0`, then the fee will be set to a non-zero amount. The `DistributeToken` event will log 0 as the fee amount if the fee is 0.

SDN-06 | Lack of Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Informational	projects/Superlauncher/contracts/SuperDeedNFT.sol (f06ee54): 84~86	Resolved

Description

In the constructor, the input variables `minter`, `distributor`, and `feeAddress` should not be zero addresses. Although these addresses might be checked in the `Factory` contract, contract `SuperDeedNFT` can be deployed separately.

Recommendation

We advise the client to check that the aforementioned variables are not zero address.

Alleviation

The team heeded our advice and resolved this issue in the commit e9a93b62938f8e746240ab486ef2f55b8e16d64d.

(**Superlauncher Team Note**)

Although `SuperDeedNFT` can be deployed by any EOA or malicious contract, but it will not be registered in the official Factory contract. Only valid `Campaign` and `SuperDeedNft` from factory will be loaded in our frontend website.

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Coding Style

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

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.