CERTIK

Security Assessment

CryptoSlam - DroppingNow

May 19th, 2022

Table of Contents

About

Summary

This report has been prepared for CryptoSlam to discover issues and vulnerabilities in the source code of the CryptoSlam - DroppingNow 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	CryptoSlam - DroppingNow
Platform	Ethereum
Language	Solidity
Codebase	https://bitbucket.org/cryptoslam/droppingnow-evm-contracts/src/master/12d5f5d0b4baa26f6c5b454f8c40a30194ad61ad
Commit	12d5f5d0b4baa26f6c5b454f8c40a30194ad61ad

Audit Summary

Delivery Date	May 19, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

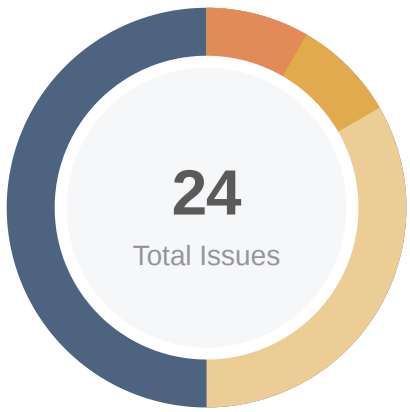
Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	2	0	0	2	0	0	0
Medium	2	0	0	2	0	0	0
Minor	8	0	0	8	0	0	0
Informational	12	0	0	12	0	0	0
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
PCD	contracts/PriceCalculatorDrop25PerDay.sol	ac48b9f001ea28f1ad9ef661c78d038976797078f702c47b0a27977a2cd56927
TME	contracts/TokenManagerERC1155.sol	d665944346380580db916e9b6fff1848de56ca5f4e7fca2e32349dc26346698e
ICR	contracts/interfaces/ICollectionsRegistry.sol	8af90a71ec755372f25998743d50f0abc299ad50e7ce1cf18876ecc146aaface
IDT	contracts/interfaces/IDropperToken.sol	8c8f3d2f2efa46b09cd4a67b90e406a4928866ba40e2f735b4804cf68a2e8738
SEC	contracts/SimpleERC20Token.sol	14a8ab0e91411983fbee662c32baafe185adaaa36add9ac4579e2756a2bf42cc
ITS	contracts/interfaces/ITokenManagerSelector.sol	becd95e48847b71052b004c7de1ece1c4816a7f2b22bf84f45fe2ebc15387e09
HHC	contracts/libraries/HashHelper.sol	ba0f2cfba69c0846fee4577812df79c22332579514c714b1275008b733e90476
DTC	contracts/DropperToken.sol	dba3eaf0474cf3cc047b4aaf8c095d4b2d2dba0148ff9d2eb7791b8351bf74f6
CRC	contracts/CollectionsRegistry.sol	ed12e2773b63c4920bd122a2339f38ea265653d8c3ac3a78a6f961fc2f816dc2
IPM	contracts/interfaces/IPriceCalculatorManager.sol	d060fd97635852a84845a4e985fee4d7f04fed19b1ab22a1cb68373afc20c723
DNT	contracts/DroppingNowToken.sol	de3aa06b074c5d4f86e10df0ef9201936c04aea23b26b0dc97f3a97c4eeb3c16
ITM	contracts/interfaces/ITokenManager.sol	8c75c45badd1a1be480bba0e2e71d79b0da065532658ce724aa82ff2d4b7d51c
SET	contracts/SimpleERC721Token.sol	06d925ec0e086ba491a7def210c3ab0de79979a506820ef75e65d2402b3261c5
TMS	contracts/TokenManagerSelector.sol	52db4fcdfc9050e7dc467ca1e70a497e18972c9710b1eb544cc4c68fb2e3ff1a
DRE	contracts/DropRewardEscrow.sol	47f60af03207d86c72d9670441c33d6e4d71d9b4a8172d30860133cff771943c
PCM	contracts/PriceCalculatorManager.sol	46f5e4eeea722ae85c97b92707fb35dc73f2b45a0801a1eb26f940f2f99b370b

ID	File	SHA256 Checksum
DNM	contracts/DroppingNowMarketplace.sol	cc5360c602e2e4562cc3c78bf1c99197773cc58bdf6c20c6294a2a71f246c56a
IPC	contracts/interfaces/IPriceCalculator.sol	a1a8decfdda76fa4dcaa084d84d68a57e368615b6d736a168cbe8a5d99ac2362
SER	contracts/SimpleERC1155Token.sol	1ac17cdffd8d4adcac3235310f1efb2781760d094255aa6d4411c64466c29352
TMM	contracts/TokenManagerMarketplace.sol	b1d901c878a303348a245f12473cac369ce4c2e746e0f521e3a79b7acbaefd53
TMR	contracts/TokenManagerERC721.sol	ee710cfd890f087c12c2caaf611f29dcb19ce3462f43e702b6415f01c5cedf9c
IDN	contracts/interfaces/IDroppingNowToken.sol	294db8d3f4ea634eccef2c63edb05765d0ad8d62fac502ebf58ec2e9e3624809

Findings



Critical	**0** (0.00%)
Major	**2** (8.33%)
Medium	**2** (8.33%)
Minor	**8** (33.33%)
Informational	**12** (50.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CRC-01	Missing Access Control	Logical Issue	Medium	Acknowledged
CRC-02	Contract Lacks A Mechanism To Revoke Malicious Collections	Logical Issue	Medium	Acknowledged
CSC-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
CSC-02	Missing Input Validation	Volatile Code	Minor	Acknowledged
CSC-03	Incorrect Return Value	Logical Issue	Minor	Acknowledged
CSC-04	Usage Of `transfer()` For Sending Ether	Volatile Code	Minor	Acknowledged
CSC-05	Missing Zero Address Validation	Volatile Code	Minor	Acknowledged
CSC-06	Unused Return Value	Volatile Code	Minor	Acknowledged
CSC-07	Magic Numbers	Coding Style	Informational	Acknowledged
CSC-08	`PriceCalculatorManager` Contract Is Almost Identical To `TokenManagerMarketplace`	Coding Style	Informational	Acknowledged
CSC-09	Missing Emit Events	Coding Style	Informational	Acknowledged
CSC-10	Improper Usage Of `public` And `external` Type	Gas Optimization	Informational	Acknowledged

ID	Title	Category	Severity	Status
DNM-01	Inconsistent Function With Documentation `cancelSingleAuction()`	Inconsistency	Major	Acknowledged
DNM-02	Local Variable Should Be State Variable	Coding Style	Informational	Acknowledged
DNM-03	Redundant Assignment For `ownerHasCorrectAddressAndApproved`	Coding Style	Informational	Acknowledged
DNM-04	Redundant Condition	Coding Style	Informational	Acknowledged
DNM-05	Code Duplication	Coding Style	Informational	Acknowledged
DTC-01	Missing Return Value	Logical Issue	Minor	Acknowledged
PCD-01	Price Recalculation	Logical Issue, Coding Style, Gas Optimization	Informational	Acknowledged
PCD-02	`currentPrice` Loop Calculation Instead Of Direct Calculation	Gas Optimization	Informational	Acknowledged
SEC-01	Repeated Condition On `supportsInterface()`	Logical Issue	Minor	Acknowledged
TMM-01	Unclear Error Message	Coding Style	Informational	Acknowledged
TMR-01	Inconsistent Return Values	Inconsistency	Minor	Acknowledged
TMS-01	Confusing Variable Name `tokenManagerSelectorForTokenAddress`	Coding Style	Informational	Acknowledged

CRC-01 | Missing Access Control

Category	Severity	Location	Status
Logical Issue	Medium	contracts/CollectionsRegistry.sol: 12	Acknowledged

Description

The function `approveCollection()` from contract `CollectionsRegistry` does not have any access control mechanism that limits which users can call it.

This lack of control allows any bad actor to approve their malicious contracts.

Recommendation

We advise the client to create an additional step in which the users can submit their collection for approval, and then the team can approve them or not based on their criteria.

Alleviation

`[CryptoSlam]`: It is expected behavior of the business logic on the platform. And we wouldn't consider it as an issue.

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CRC-02 | Contract Lacks A Mechanism To Revoke Malicious Collections

Category	Severity	Location	Status
Logical Issue	Medium	contracts/CollectionsRegistry.sol: 7	Acknowledged

Description

The contract `CollectionsRegistry` only can approve collections. In the case of a user who approves a malicious contract, there is no mechanism to revoke the approval from that contract.

This lack of mechanism would have the consequence of having a malicious contract permanently approved without the possibility of removing it. This scenario could lead to damage to the team's reputation.

Recommendation

We advise the team to add a black list controlled by the team. This technique would allow the possibility of revoking a collection's approval.

It also prevents a malicious contract from being approved in the future by requiring that the contract is not on the black list before approving it.

Alleviation

`[CryptoSlam]`: In the business logic we do not act like moderators and everyone should be able to list on DN. It is a part of decentralization.

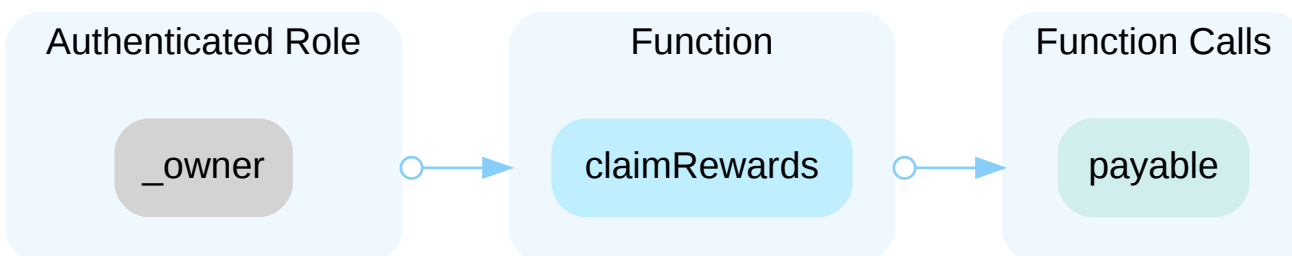
`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-01 | Centralization Related Risks

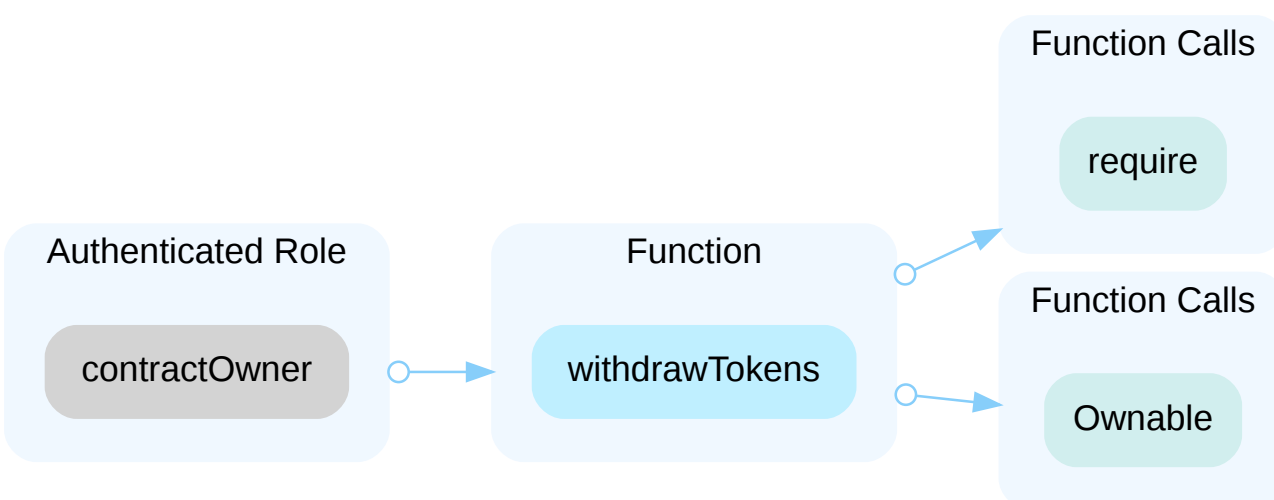
Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/TokenManagerSelector.sol: 25, 34; contracts/TokenManagerMarketplace.sol: 21, 28; contracts/SimpleERC721Token.sol: 15; contracts/SimpleERC1155Token.sol: 15; contracts/PriceCalculatorManager.sol: 16, 23; contracts/DroppingNowMarketplace.sol: 144, 148, 362, 376, 388, 402, 414, 420, 426, 432, 438, 444, 450, 456, 462, 468; contracts/DropRewardEscrow.sol: 20, 25	Acknowledged

Description

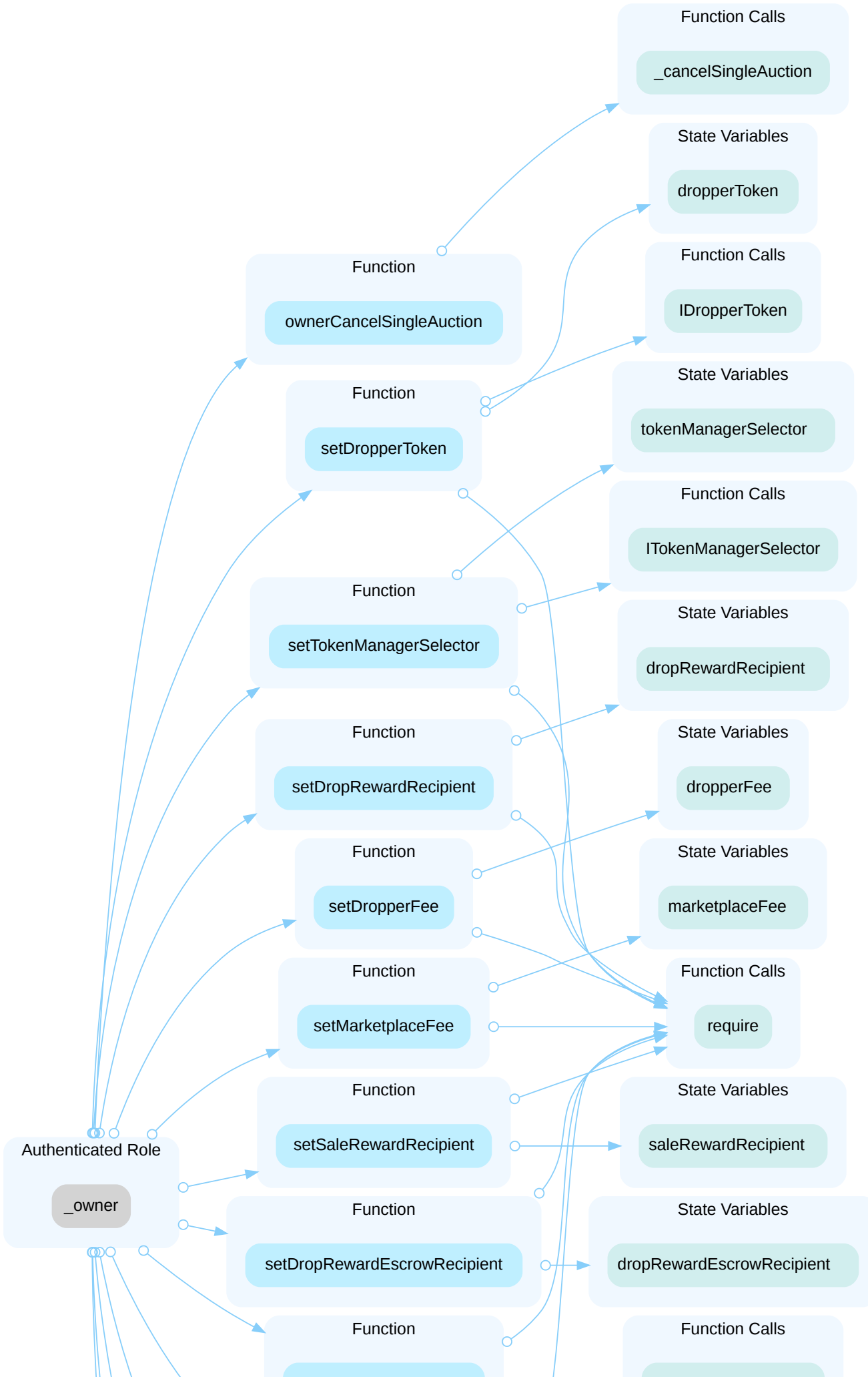
In the contract `DropRewardEscrow` the role `_owner` has authority over the functions shown in the diagram below.

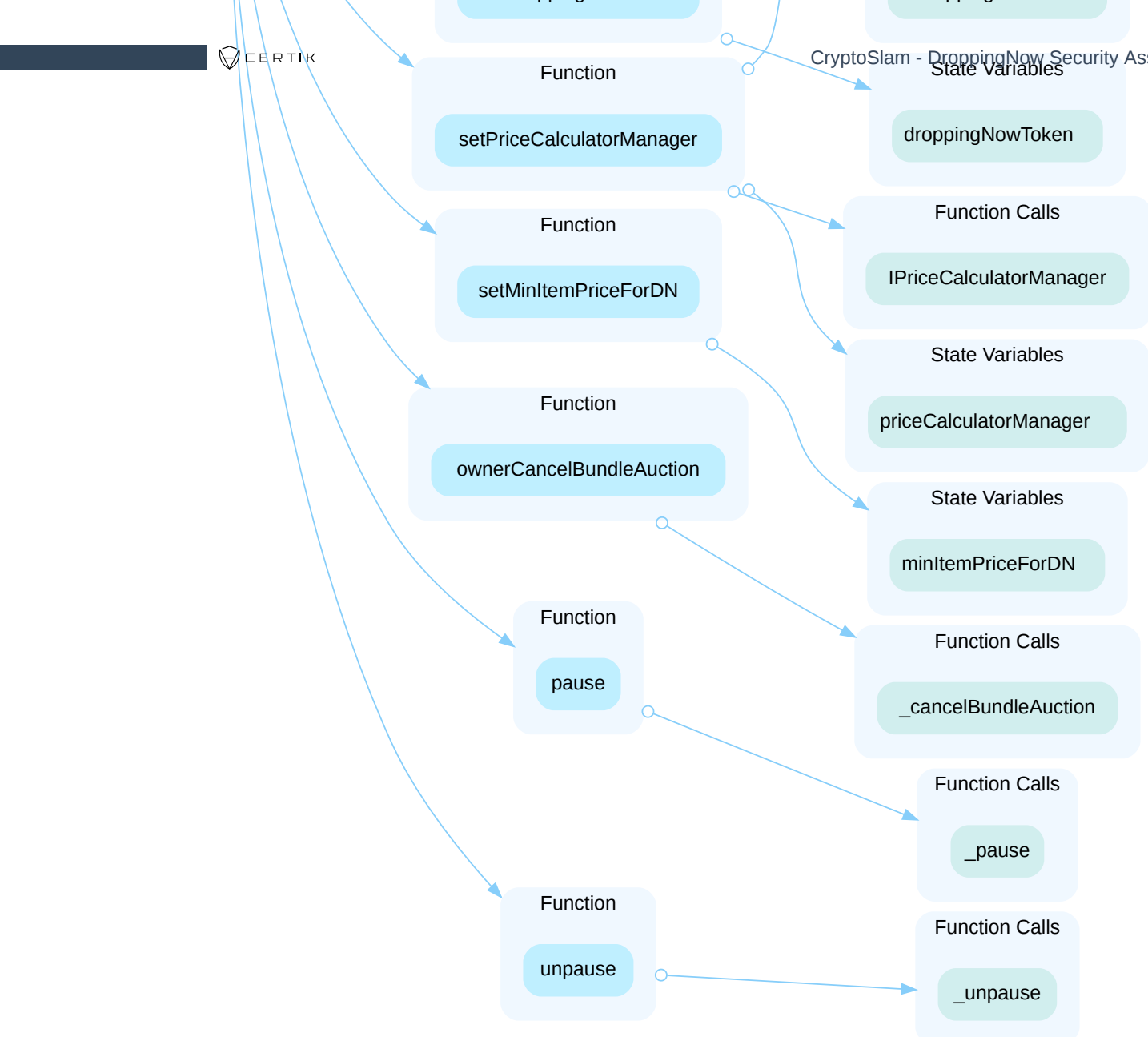


In the contract `DropRewardEscrow` the role `contractOwner` has authority over the functions shown in the diagram below.

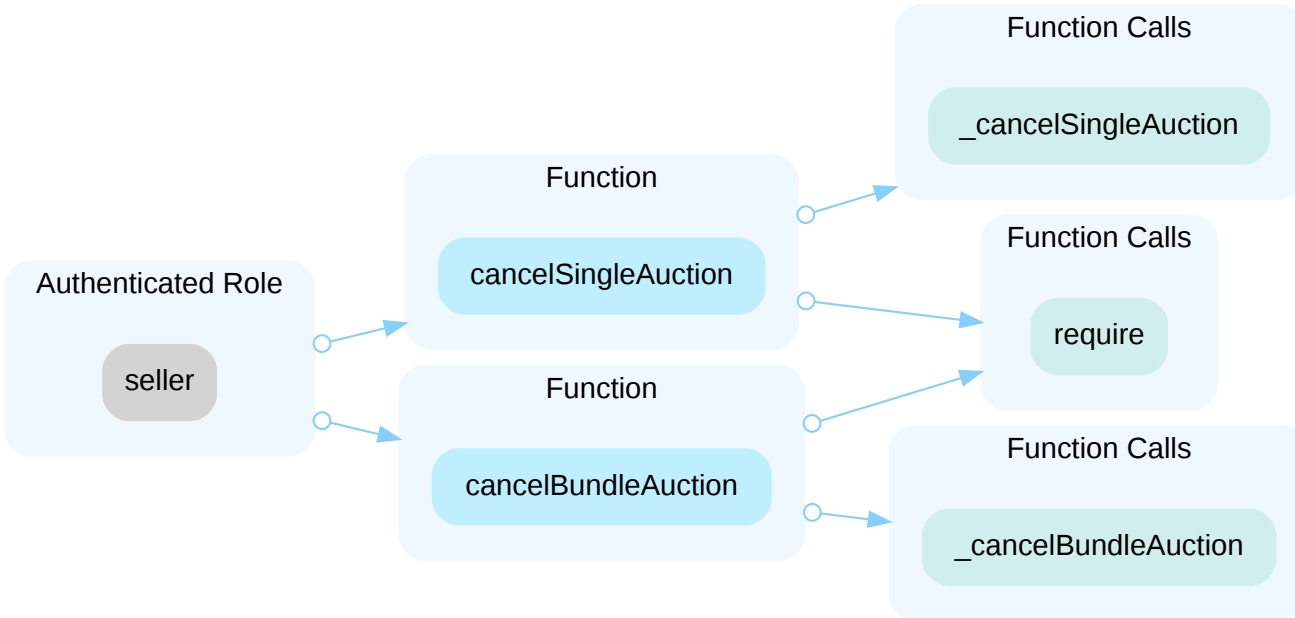


In the contract `DroppingNowMarketplace` the role `_owner` has authority over the functions shown in the diagram below.

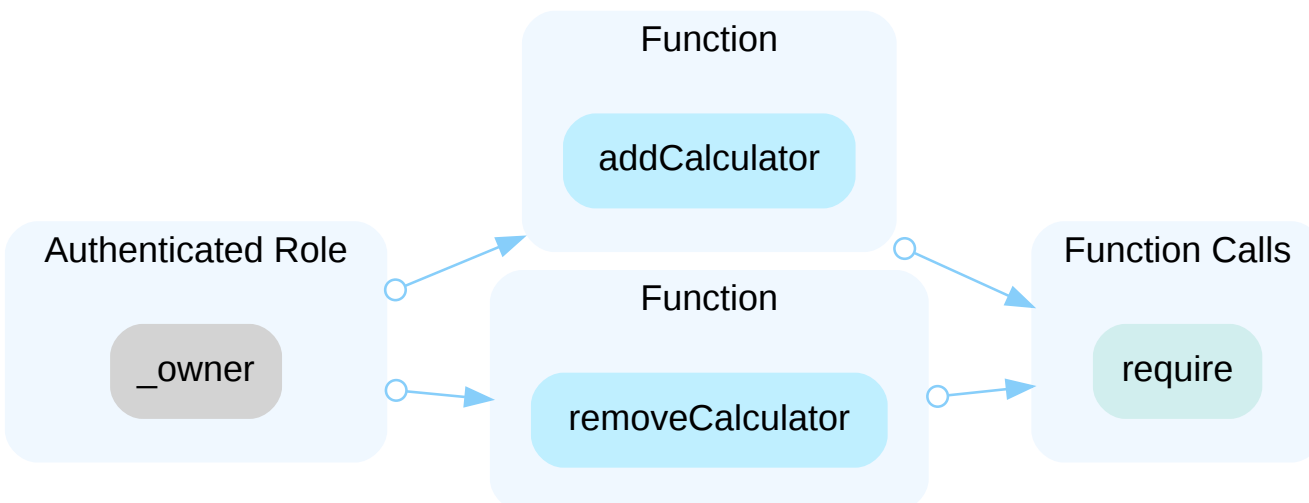

Function Calls
_cancelSingleAuction
State Variables
dropperToken
Function
ownerCancelSingleAuction
Function Calls
IDropperToken
Function
setDropperToken
State Variables
tokenManagerSelector
Function Calls
ITokenManagerSelector
Function
setTokenManagerSelector
State Variables
dropRewardRecipient
Function
setDropRewardRecipient
State Variables
dropperFee
Function
setDropperFee
State Variables
marketplaceFee
Function
setMarketplaceFee
Function Calls
require
Function
setSaleRewardRecipient
State Variables
saleRewardRecipient
Authenticated Role
_owner
Function
setDropRewardEscrowRecipient
State Variables
dropRewardEscrowRecipient
Function
Function Calls



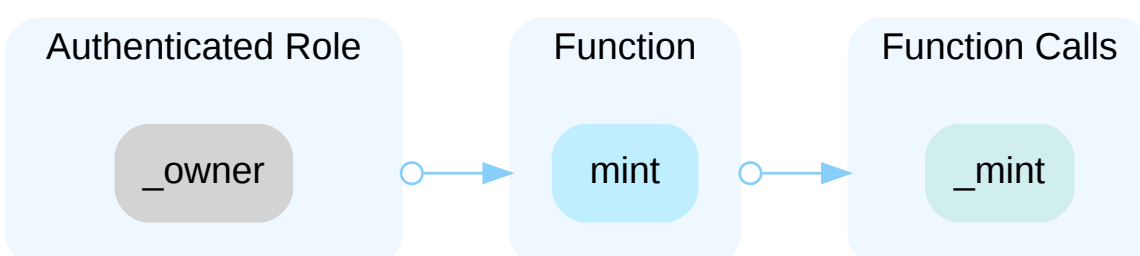
In the contract `DroppingNowMarketplace` the role `seller` has authority over the functions shown in the diagram below.



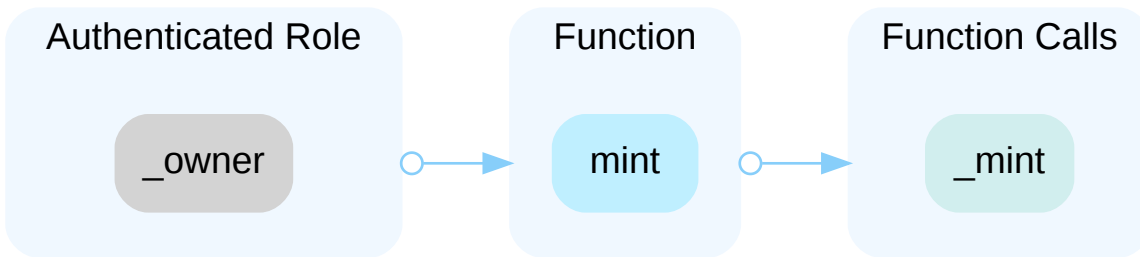
In the contract `PriceCalculatorManager` the role `_owner` has authority over the functions shown in the diagram below.



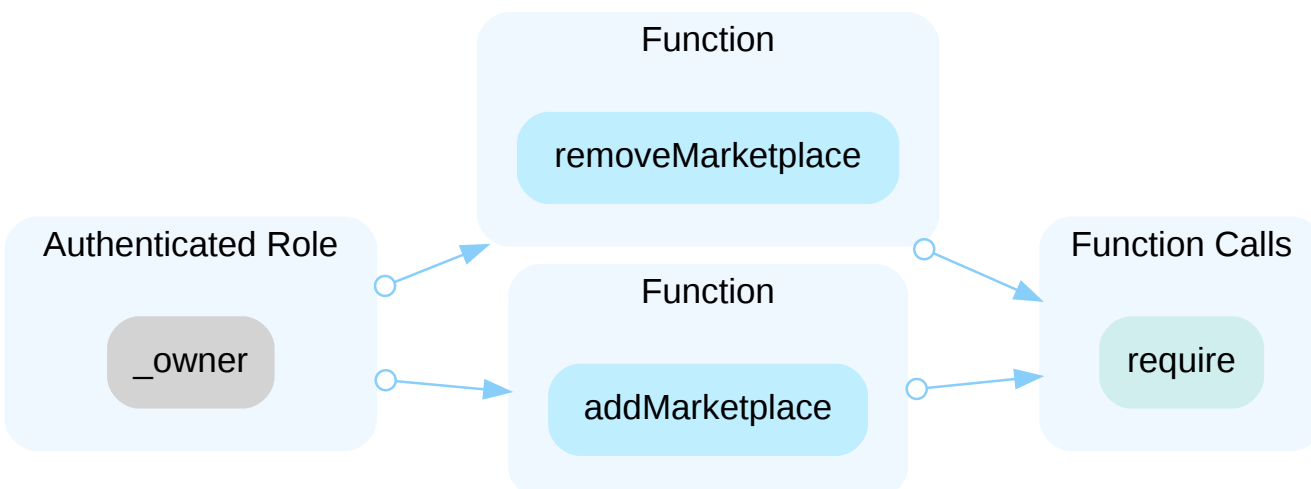
In the contract `SimpleERC1155Token` the role `_owner` has authority over the functions shown in the diagram below.



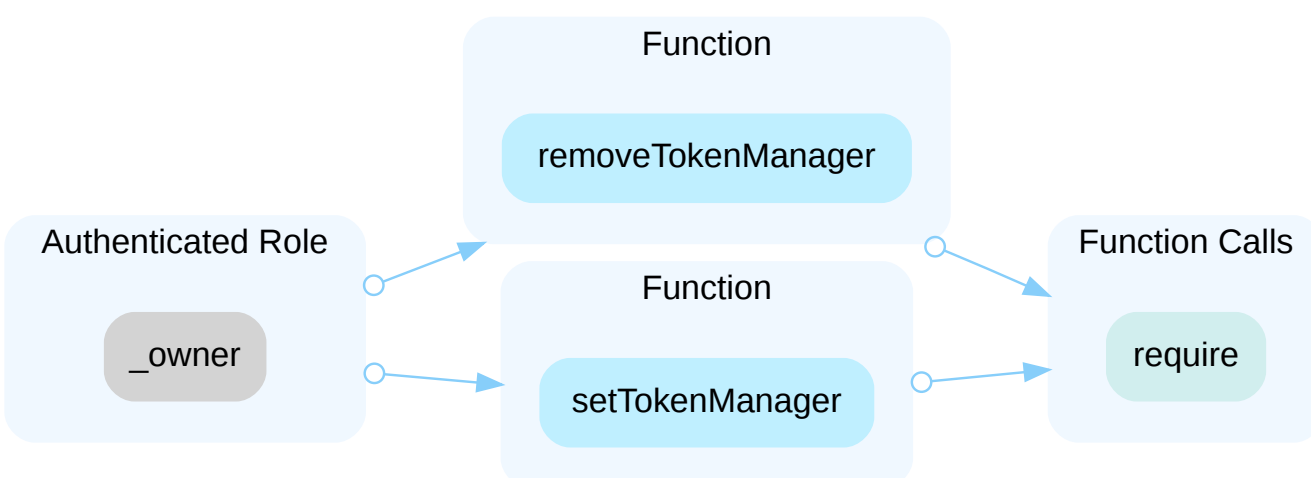
In the contract `SimpleERC721Token` the role `_owner` has authority over the functions shown in the diagram below.



In the contract `TokenManagerMarketplace` the role `_owner` has authority over the functions shown in the diagram below.



In the contract `TokenManagerSelector` the role `_owner` has authority over the functions shown in the diagram below.



Any compromise to these privileged accounts may allow the hacker to take advantage of these authorities and access sensitive functionalities.

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

Indicatively, here are some feasible suggestions that would also 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 key management 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.

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.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	contracts/DropperToken.sol: 35, 39, 245; contracts/DroppingNowMarketplace.sol: 139, 140	Acknowledged

Description

The linked statements make use of a parameter that was not validated before its usage.

In the case of the variables `dropperFeeValue` and `marketplaceFeeValue` from the constructor of the contract `DroppingNowMarketplace`, there is no check that prevents the values from being greater than `10.000`.

In the case of the variable `contractUri` from the function `_setContractURI()` from the contract `DropperToken`, there is no check that prevents the variable from being an empty string.

In the case of the variable `newUri` from the function `setURI()` from contract `DropperToken`, there is no check that prevents the value from being an empty string. A similar case happens to the variable `contractUri` from the function `setContractURI()` from the same contract.

Recommendation

We advise the team to add `require` statements verifying in each case that the input is valid.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-03 | Incorrect Return Value

Category	Severity	Location	Status
Logical Issue	Minor	contracts/PriceCalculatorManager.sol: 55; contracts/TokenManagerMarketplace.sol: 60	Acknowledged

Description

The function `viewAllowedMarketplaces()` returns the value `cursor + length`. In the case where the cursor is bigger than the size of the array, the variable `length` will be zero, and the `allowedMarketplaces` array will be empty, but the second return value will be the cursor value rather than the size of the `allowedMarketplaces` array.

Recommendation

We advise the team to replace the second return value with `length` if this was not the intended behavior.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-04 | Usage Of `transfer()` For Sending Ether

Category	Severity	Location	Status
Volatile Code	Minor	contracts/DroppingNowMarketplace.sol: 290, 295, 351, 356; contracts/DropRewardEscrow.sol: 22; contracts/DropperToken.sol: 102, 121; contracts/DroppingNowToken.sol: 59	Acknowledged

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-05 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	contracts/DropperToken.sol: 86, 258; contracts/DroppingNowToken.sol: 32; contracts/DroppingNowMarketplace.sol: 136, 137, 138, 290, 351; contracts/TokenManagerSelector.sol: 21, 22	Acknowledged

Description

Addresses should be checked before assignment usage to make sure they are not zero addresses.

Recommendation

We advise the team to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-06 | Unused Return Value

Category	Severity	Location	Status
Volatile Code	Minor	contracts/PriceCalculatorManager.sol: 18, 25; contracts/TokenManagerMarketplace.sol: 23, 30	Acknowledged

Description

The linked statements do not handle the return value of the external functions they are calling.

Recommendation

We recommend checking or using the return values of all external function calls.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-07 | Magic Numbers

Category	Severity	Location	Status
Coding Style	Informational	contracts/DroppingNowMarketplace.sol: 692, 693, 697, 698, 725, 726, 727; contracts/PriceCalculatorDrop25PerDay.sol: 34, 47, 57, 60; contracts/TokenManagerSelector.sol: 26, 27, 36, 40, 48	Acknowledged

Description

The linked magic numbers should be set as `constant` and `internal` contract-level variables with a self-explanatory variable name as well as accompanying comments when necessary. This type of declaration is functionally equivalent to the current implementation as `constant` variables that are `internal` or `private` are simply replaced in the codebase with their literal value.

In the case of the `TokenManagerSelector` contract, the usage of the zero address can be replaced by the private variable `NO_MANAGER` to improve the code's readability.

Recommendation

We advise the team adds proper documentation specifying the purpose of the linked numbers.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-08 | `PriceCalculatorManager` Contract Is Almost Identical To `TokenManagerMarketplace`

Category	Severity	Location	Status
Coding Style	Informational	contracts/PriceCalculatorManager.sol: 8; contracts/TokenManagerMarketplace.sol: 8	Acknowledged

Description

The contracts `PriceCalculatorManager` and `TokenManagerMarketplace` are almost identical. They both manage assets that are added, removed, and listed.

A base contract could be created with this shared functionality to avoid duplicating the code.

Recommendation

We advise the team to consider creating a base contract with this shared functionality on which these two contracts will inherit.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-09 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/DroppingNowToken.sol: 29, 41; contracts/DropperToken.sol: 34, 83, 244, 248	Acknowledged

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

CSC-10 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/SimpleERC1155Token.sol: 15; contracts/DropperToken.sol: 227, 236, 240; contracts/SimpleERC20Token.sol: 13; contracts/DroppingNowToken.sol: 107; contracts/SimpleERC721Token.sol: 15	Acknowledged

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

DNM-01 | Inconsistent Function With Documentation `cancelSingleAuction()`

Category	Severity	Location	Status
Inconsistency	Major	contracts/DroppingNowMarketplace.sol: 362~374	Acknowledged

Description

In the project's documentation, it states that the auction's creator can't cancel its listing.

However, this is possible via the `cancelSingleAuction()` function, which only requires that the one calling it is the auction's seller.

Recommendation

We advise the team to remove the function `cancelSingleAuction()` to comply with the documentation.

Alleviation

`[CryptoSlam]`: Issue acknowledged. I won't make any changes for the current version. `[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

DNM-02 | Local Variable Should Be State Variable

Category	Severity	Location	Status
Coding Style	Informational	contracts/DroppingNowMarketplace.sol: 724~727	Acknowledged

Description

The local variable `amount` from the function `_getDropRewardAmounts()` from the contract `DroppingNowMarketplace` is an array with hard-coded values which is returned to the user.

As this variable does not change and it defines a parameter inside the contract, it should be a contract's `public` variable.

Recommendation

We advise the team to create a `public` state variable `dropRewardAmounts`, remove the function `_getDropRewardAmounts()` and replace every use of the function with the variable `dropRewardAmounts`.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

DNM-03 | Redundant Assignment For `ownerHasCorrectAddressAndApproved`

Category	Severity	Location	Status
Coding Style	Informational	contracts/DroppingNowMarketplace.sol: 679	Acknowledged

Description

The variable `ownerHasCorrectAddressAndApproved` from function `_saleReward()` from contract `DroppingNowMarketplace` has a redundant value when it is assigned.

The variable is assigned only if the variable `isApproved` is true. If this happens, the new value will be `isApproved && owner != address(0)`. There is no need to make an `and` operation as the value `isApproved` is already true.

Recommendation

We advise the team to remove the `and` operation and make the assignment on line 679 as: `ownerHasCorrectAddressAndApproved = owner != address(0);`.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

DNM-04 | Redundant Condition

Category	Severity	Location	Status
Coding Style	Informational	contracts/DroppingNowMarketplace.sol: 457, 463	Acknowledged

Description

The linked `require` statements have a condition that checks whether a variable of type `uint` is greater or equal to zero.

This check is redundant for these types of variables as they are non-negative integers.

Recommendation

We advise the team to replace the greater or equal sign with the strict greater than sign to check whether the variable is different from zero.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

DNM-05 | Code Duplication

Category	Severity	Location	Status
Coding Style	Informational	contracts/DroppingNowMarketplace.sol: 242, 301	Acknowledged

Description

The functions `buySingleAuction()` and `buyBundleAuction()` from contract `DroppingNowMarketplace` are almost identical. The shared functionality can be placed into a separate `internal` or `private` function to improve the code's readability.

Recommendation

We advise the team to extract the shared functionality from both functions into a separate one and use it from the mentioned functions to improve the code's readability.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

DTC-01 | Missing Return Value

Category	Severity	Location	Status
Logical Issue	Minor	contracts/DropperToken.sol: 42, 59	Acknowledged

Description

The function `tryAddMintable()` from contract `DropperToken` doesn't have a return value. This lack of feedback makes it difficult to tell if the function added or not the token address.

A similar case occurs with the function `tryAddMintableBatch()`.

Recommendation

Regarding the function `tryAddMintable()`, we advise the team to add a boolean return value to differentiate if it added or not the token address to the `_addressToMintedForToken()`

Regarding the function `tryAddMintableBatch()`, we advise the team to return the `count` local variable.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

PCD-01 | Price Recalculation

Category	Severity	Location	Status
Logical Issue, Coding Style, Gas Optimization	Informational	contracts/PriceCalculatorDrop25PerDay.sol: 27, 32	Acknowledged

Description

The function `isPriceAllowed()` makes the same multiplication twice. The first one is to check if the operation is safe, while the last one is part of the calculation of the `nextDayPrice`.

The calculation of `nextDayPrice` could use the second return value of SafeMath's `tryMul()`, which is the result of the operation, to save the cost of multiplication and the default security checks.

Recommendation

We advise the team to handle the second return value of SafeMath's `tryMul()` and reuse it to calculate `nextDayPrice`.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

PCD-02 | `currentPrice` Loop Calculation Instead Of Direct Calculation

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/PriceCalculatorDrop25PerDay.sol: 49~55	Acknowledged

Description

The function `_calculate()` from contract `PriceCalculatorDrop25PerDay` makes a `for` loop to calculate the current price devaluation given its starting price and the number of days.

The `for` loop can be replaced with the direct calculation of `(currentPrice * (3 ** daysGone)) >> (2 * daysGone);`. Being, `>>` the shift operator.

This calculation is much cheaper as it doesn't depend on the number of iterations. After making a few tests, we saw that if we call the `for` loop, it had a gas cost of `12765` when the recommended version only used `3159`. The values used were: `currentPrice = 748` and `daysGone = 15`.

Recommendation

We advise the team to consider the benefits and drawbacks of the recommended implementation.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

SEC-01 | Repeated Condition On `supportsInterface()`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/SimpleERC20Token.sol: 16	Acknowledged

Description

The function `supportsInterface()` from contract `SimpleERC20Token` has the condition `interfaceId == type(IERC20).interfaceId` repeated twice.

Given the context, it seems that, on the duplicated line, the interface used should be `IERC165`.

Recommendation

We advise the team to replace the interface `IERC20` with `IERC165` on line 16 if it was the intended behavior or delete the duplicated line if it wasn't.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

TMM-01 | Unclear Error Message

Category	Severity	Location	Status
Coding Style	Informational	contracts/TokenManagerMarketplace.sol: 29	Acknowledged

Description

The function `removeMarketplace()` from contract `TokenManagerMarketplace` has a `require` statement which has an unclear error message "TokenManagerMarketplace: Not allowed". This gives the user the intuition that the action is not allowed, but instead it refers to that the marketplace was not allowed in the first place.

Recommendation

We advise the team to rewrite the linked error messages to better express why they did not suffice.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

TMR-01 | Inconsistent Return Values

Category	Severity	Location	Status
Inconsistency	Minor	contracts/TokenManagerERC721.sol: 18, 28	Acknowledged

Description

The functions `deposit()` and `withdraw()` from contract `TokenManagerERC721` return the value zero. This value is inconsistent with the contract `TokenManagerERC1155`, where the return value is the number of tokens transferred.

Recommendation

We advise the team to replace the value `0` with `1` to maintain consistency across the different implementations and return the number of tokens transferred.

Alleviation

`[CryptoSlam]`: according to business logic we require NFTs do not have amount. Won't change. `[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

TMS-01 | Confusing Variable Name `tokenManagerSelectorForTokenAddress`

Category	Severity	Location	Status
Coding Style	Informational	contracts/TokenManagerSelector.sol: 15, 29, 36, 40, 46	Acknowledged

Description

The variable `tokenManagerSelectorForTokenAddress` from contract `TokenManagerSelector` is a bit confusing.

Recommendation

We advise the team to rename the variable to `tokenManagerFor` so that when using it, it reads as `tokenManagerFor[tokenAddress]`.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

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.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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.