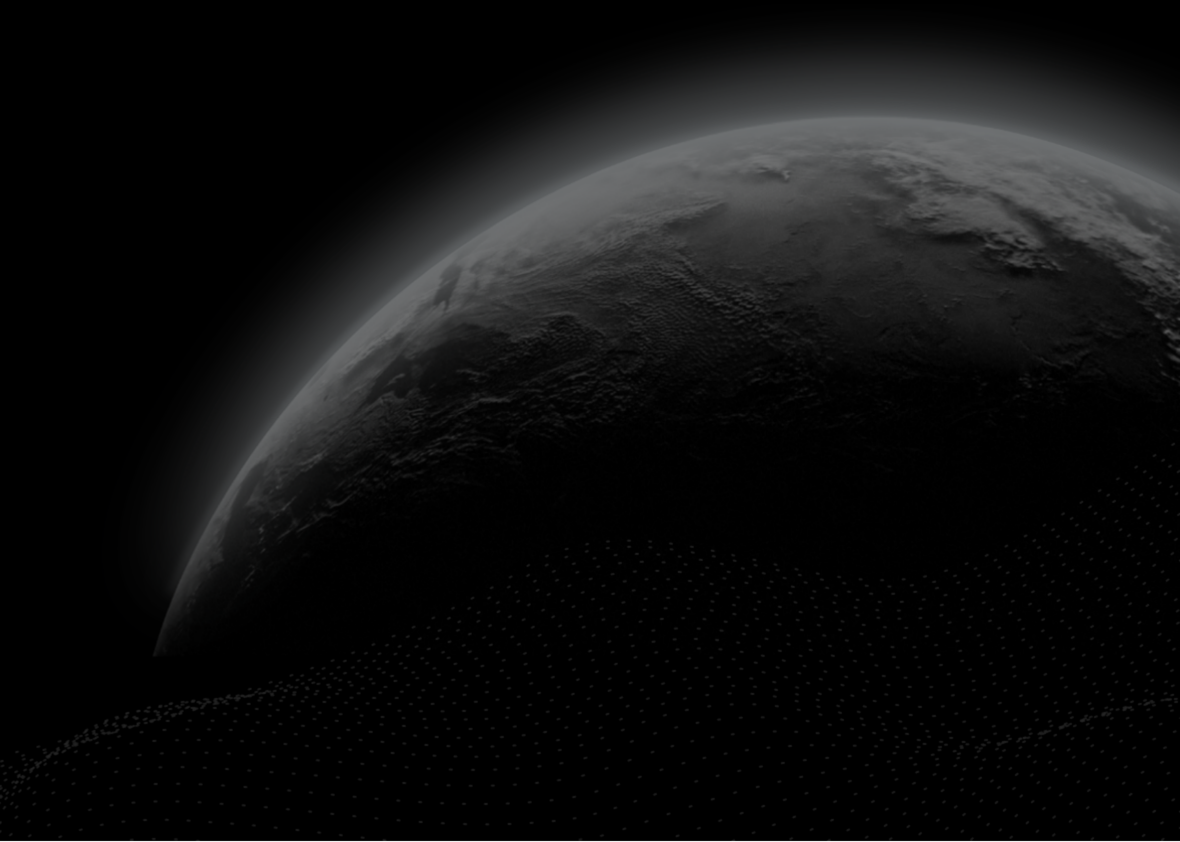
CERTIK

Security Assessment

Decentraland

CertiK Assessed on Aug 16th, 2024



CertiK Assessed on Aug 16th, 2024

Decentraland

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum (ETH) | Polygon (MATIC)	Formal Verification, Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 08/16/2024	N/A

CODEBASE

Decentraland Marketplace

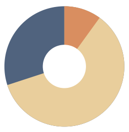
View All in Codebase Page

COMMITS

- 4b4b7252698732b546e8b4f35a675beef568d284
- 8950b0941af42140d22c3d2ef344920c0b07dde3

View All in Codebase Page

Vulnerability Summary



10	1	0	0	9	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥	0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	1	Major	1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	6	Minor	6 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	3	Informational	1 Resolved, 2 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | DECENTRALAND

CODEBASE | DECENTRALAND

Repository

Decentraland Marketplace

Commit

- 4b4b7252698732b546e8b4f35a675beef568d284
- 8950b0941af42140d22c3d2ef344920c0b07dde3

AUDIT SCOPE | DECENTRALAND

52 files audited ● 13 files with Acknowledged findings ● 1 file with Resolved findings ● 38 files without findings

ID	Repo	File	SHA256 Checksum
● IAB	decentraland/offchain-marketplace-contract	marketplace/interfaces/IAggregator.sol	bb6c382310c4959ff35ddac4a2b4177d4cd6cc020e2de56322b829c87dfa1d82
● IRM	decentraland/offchain-marketplace-contract	marketplace/interfaces/IRoyaltiesManager.sol	d9764f801e740243f91c2e07ec07bd42b4d6da04546ad54d243716d58288e9e2
● DME	decentraland/offchain-marketplace-contract	marketplace/DecentralandMarketplaceEthereum.sol	30d36c4c0f318ef2f53ea80a4c952cfb461773ddf840d6c67c32f84bb29184ba
● DMP	decentraland/offchain-marketplace-contract	marketplace/DecentralandMarketplacePolygon.sol	330cef3045ae5996aff8771dd3c9e044ae7ed228ff3858cf1835dbf67144e5b6
● FCB	decentraland/offchain-marketplace-contract	marketplace/FeeCollector.sol	73051c37be17a6d36bdff0a444aab0535b44b759c3db36dbf838162bc7c5cc80
● MAK	decentraland/offchain-marketplace-contract	marketplace/Marketplace.sol	e33933ce9862ed4ea2511d66e39ce6af329212b0c14c0059b371de62a4ea9454
● MWC	decentraland/offchain-marketplace-contract	marketplace/MarketplaceWithCouponManager.sol	ac89b054484202c16e71c90d048a5941313b1d6a4ecbae97ff902d5b33d3f2b0
● ICM	decentraland/offchain-marketplace-contract	coupons/interfaces/ICouponManager.sol	1372f0434719f26b566152f2805e97ecc6f4ea549d8709a1ef078ba5fc534b42
● CDC	decentraland/offchain-marketplace-contract	coupons/CollectionDiscountCoupon.sol	0f1ccefbdee454f422065618270d8fc211ab33bf886309810eb98f52184fe6c7
● CMB	decentraland/offchain-marketplace-contract	coupons/CouponManager.sol	3b33465193e4c865fe5240bfbb4d329e21d4af746a9b7c31471237e89a71d025
● NMT	decentraland/offchain-marketplace-contract	common/NativeMetaTransaction.sol	5ff2cc8b15b9dd0c08d21198d8adb53300374acc3b14baeacb94e359ee8bef17
● SIG	decentraland/offchain-marketplace-contract	common/Signatures.sol	cff8f29b1d16960a4a90a5493f1fc0285548b92c5908e64ff27629b3deb3601b
● VER	decentraland/offchain-marketplace-contract	common/Verifications.sol	2fea6b63fd1c775ba7d3c7b10995c3ccc72aa89bf37b512b66698374f6ed3559

ID	Repo	File	SHA256 Checksum
● ICH	decentraland/offchain-marketplace-contract	coupons/interfaces/ICoupon.sol	d220ce0eebc30fc3013668dfd10c74ebef3508846887e3767b30c2b4567a62d7
● ICB	decentraland/offchain-marketplace-contract	marketplace/interfaces/ICollection.sol	6854cec2f2131cd384b8e1ed72aa7a3bdec05ce59836de47cb45c052c9325faa
● ICU	decentraland/offchain-marketplace-contract	marketplace/interfaces/IComposable.sol	8c139456ca0c86c5782b15cc5f860d18e4ad4c853b2440736966ef5d524cb9ec
● AHB	decentraland/offchain-marketplace-contract	marketplace/AggregatorHelper.sol	ce76b01acaa6f9b1b63e1eba59a421f3ff3ef12aba4d14993261f4805c757431
● DMA	decentraland/offchain-marketplace-contract	marketplace/DecentralandMarketplaceEthereumAssetTypes.sol	e18dee9bcba99d0c6087f929da0406a8cf1acef342df9a2a1805222344fa3419
● DMT	decentraland/offchain-marketplace-contract	marketplace/DecentralandMarketplacePolygonAssetTypes.sol	aebd91ebd1b7a8fb5b90151d025c84e66589825cba0d653501c5a5a34d7bd80d
● MTB	decentraland/offchain-marketplace-contract	marketplace/MarketplaceTypes.sol	db13cf90eeae7e5206783aa32e48a6a9714f38b522221116270df91b0fef360c
● MTH	decentraland/offchain-marketplace-contract	marketplace/MarketplaceTypesHashing.sol	d33f37970c399939623f59006ef7939a6ac779eaed01e35189302e86857011df
● CTB	decentraland/offchain-marketplace-contract	coupons/CouponTypes.sol	8ab502b11ce401758dfdaa25f60ce643541df2b955b466f320d53115a78a7035
● CTH	decentraland/offchain-marketplace-contract	coupons/CouponTypesHashing.sol	ad27c79d282f1191ce02014514e46172a36367446eac51ec18dd130088b4f69a
● CTU	decentraland/offchain-marketplace-contract	common/CommonTypes.sol	0014ad8b99b0988f0ccecbfd1f3750a3c12c41cd8987e9b98d9acddaafe4e77f
● COO	decentraland/offchain-marketplace-contract	common/CommonTypesHashing.sol	8e1ae96bf1aaa9ab0baeee15b381dc130053379a192adb301b3b4f269fbb5243
● EIP	decentraland/offchain-marketplace-contract	common/EIP712.sol	ccaa51e744d9d805d30ce99f84011f0b47d88e21ad503d22f5838a5305077d12
● CTT	decentraland/offchain-marketplace-contract	common/CommonTypes.sol	8df6479da9f492600fc8facd693a6428b1053bb412838fb8379086c8290336ff
● COT	decentraland/offchain-marketplace-contract	common/CommonTypesHashing.sol	2a2af44cc7285585df070258a4be251cd9e0ba28bcd02ea56a620c6b08187e5c

ID	Repo	File	SHA256 Checksum
EI7	decentraland/offchain-marketplace-contract	common/EIP712.sol	1595a4445558371ac077d4614523b24b78a71f9bfbdc5c78a0adb7685faf770b
NAT	decentraland/offchain-marketplace-contract	common/NativeMetaTransaction.sol	2d455ebbe14152b066b8f9935a3d3963de946f1d15f3865977bd5ceaf50c570a
SIN	decentraland/offchain-marketplace-contract	common/Signatures.sol	600547ea24c766a3ce672cc763dfd9f1d052da0e5321b58e1ad6ac5fbbb96e79
VEI	decentraland/offchain-marketplace-contract	common/Verifications.sol	e1d80dbbd925f051d5d1202d3f34dd29bec9ccda0b7d5f0d896b49e7e5cad999
ICT	decentraland/offchain-marketplace-contract	coupons/interfaces/ICoupon.sol	7d4f2eee043a4248cdec06dd657c60a2267762edbb344cbda5fbd1cde797f03d
ICO	decentraland/offchain-marketplace-contract	coupons/interfaces/ICouponManager.sol	4f8256f92641d8fabd1a81e3c7ff240ccb80c12ddaed6de0ea796621fa9ba53b
COL	decentraland/offchain-marketplace-contract	coupons/CollectionDiscountCoupon.sol	d003c793ec4823781d9e3fcb3bba48d42df8fe9541f6d2f32c2124069687d2af
CMU	decentraland/offchain-marketplace-contract	coupons/CouponManager.sol	91f4ab14469c82eb37ed90fc4bc2d1960adb78af5072d15f607f752f5348608e
CTI	decentraland/offchain-marketplace-contract	coupons/CouponTypes.sol	eec4773a63be64946947d7c6a3591b8a60b1bfb421ba57613d549d63e0172260
COY	decentraland/offchain-marketplace-contract	coupons/CouponTypesHashing.sol	ae04986c5f131a7596003d3197916e73ce50f4f4aa8538ee9047c983d98d88c3
IAU	decentraland/offchain-marketplace-contract	marketplace/interfaces/IAggregator.sol	b83e90c69f73c7588876545086d52c4a89b03730cbd592f4bdaada3c9e85620b
ICI	decentraland/offchain-marketplace-contract	marketplace/interfaces/ICollection.sol	a0ea77c01e8ba963640d51c7d32ee340df4e228c5242bf5333db652fea8a524e
ICG	decentraland/offchain-marketplace-contract	marketplace/interfaces/IComposable.sol	dcbdd88083d257a0b212239d8077950fe68d06169afbf7c1ec767968c1b17f20
IRO	decentraland/offchain-marketplace-contract	marketplace/interfaces/IRoyaltiesManager.sol	8f5f460bad7253836e15638f4cd8bcee1e76079f576a2e77f083a3828e7074ee
AHU	decentraland/offchain-marketplace-contract	marketplace/AggregatorHelper.sol	f482d272933f5db5e1850753a14178948943f88bb572a56baf7c2066b5d79255

ID	Repo	File	SHA256 Checksum
DEC	decentraland/offchain-marketplace-contract	marketplace/DecentralandMarketplaceEthereum.sol	43bed2e7540ab189112ab2b19ca53cb89d70b12b262b3b15e7d84a41de432f98
DEA	decentraland/offchain-marketplace-contract	marketplace/DecentralandMarketplaceEthereumAssetTypes.sol	a619ccd2b0d762e36097b38e5b535e2dfbb84ed266119c5d936163aa8a73b93b
DEE	decentraland/offchain-marketplace-contract	marketplace/DecentralandMarketplacePolygon.sol	a039c1f5966d5c2d0500df926e2161222a145f8943a3df5fc9ab8ec34f371cf3
DPA	decentraland/offchain-marketplace-contract	marketplace/DecentralandMarketplacePolygonAssetTypes.sol	d9e42d8d85426a6d003de36df93e0419c323268280ad7312d4fa15c3e6c2823c
FCU	decentraland/offchain-marketplace-contract	marketplace/FeeCollector.sol	f7c53596c81a54bb0378a3d85bd5bc94ae83c4dbd29d1599f68dcd5fa309c17a
MAT	decentraland/offchain-marketplace-contract	marketplace/Marketplace.sol	fe8b36e7b7f34d47f8b6f8feb8639080f6cb6a85179c70c5ba9a26fccd3c910c
MTU	decentraland/offchain-marketplace-contract	marketplace/MarketplaceTypes.sol	82965432b02a8184db42af12b664143e4d7f163bc1612b0be72f1ca45eef80c0
MAP	decentraland/offchain-marketplace-contract	marketplace/MarketplaceTypesHashing.sol	46dc803101d9d810676bb9659a7011665dfba72fb6b5dd4fb966a9c51e08f527
MWM	decentraland/offchain-marketplace-contract	marketplace/MarketplaceWithCouponManager.sol	cff93f2612176852e8c11318e78912391526edd9d865087bacd9e4295bf80a96

APPROACH & METHODS | DECENTRALAND

This report has been prepared for Decentraland to discover issues and vulnerabilities in the source code of the Decentraland 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:

- Testing the smart contracts against both common and uncommon attack vectors;
- 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.

REVIEW NOTES | DECENTRALAND

Overview

The Decentraland Marketplace is a decentralized platform that enables users to conduct trades using EIP712 signatures. Through this method, users can sign trades specifying the terms of the trade, and other interested parties can accept and settle these trades on the blockchain. The marketplace supports assets specific to Decentraland and incorporates the current fee and royalty systems.

The Decentraland Marketplace has two distinct implementations, each tailored to a different blockchain network: Ethereum and Polygon.

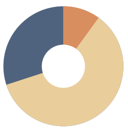
Third-Party Dependency Usage

The Decentraland Marketplace is serving as the underlying entity to interact with one or more third party protocols, such as OpenZeppelin cryptography, Chainlink Aggregator and the Off-Chain DAPP. 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. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties.

Chainlink aggregators include a circuit breaker mechanism that activates if the price of an asset moves outside a set range. Specifically, if an asset’s price plummets significantly, the aggregator will continue to return the predefined `minAnswer` rather than the asset’s actual current price.

The `latestRoundData` function in Chainlink extracts data from these aggregators. Each aggregator incorporates `minAnswer` and `maxAnswer` values as part of its circuit breaker system. When an asset's price falls below `minAnswer`, the protocol continues to value the token at this floor price instead of reflecting its true market value. This discrepancy can cause severe issues within the protocol, potentially leading to substantial financial losses.

FINDINGS | DECENTRALAND



10	0	1	0	6	3
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Decentraland. Through this audit, we have uncovered 10 issues ranging from different severity levels. Utilizing the techniques of Static Analysis & Manual Review to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
SRC-01	**Centralization Related Risks**	**Centralization**	**Major**	● **Acknowledged**
DMP-01	Unfair Fee Payment For Decentraland NFTs In Mixed Trades	Logical Issue	Minor	● Acknowledged
MAR-01	Lack Of Reasonable Upper Boundaries On Fees	Logical Issue	Minor	● Acknowledged
MAR-02	Missing Zero Address Validation	Volatile Code	Minor	● Acknowledged
SRC-02	Third-Party Dependency Usage	Design Issue	Minor	● Acknowledged
SRC-03	Lack Of Revocation Status Validation In `cancelSignature()` Function	Logical Issue	Minor	● Acknowledged
VER-01	Improper Handling Of Signature Expiration In `_verifyChecks()` Function	Logical Issue	Minor	● Acknowledged
CDC-01	Limitation In `applyCoupon()` Function For Coupon Distribution	Design Issue	Informational	● Acknowledged
COU-01	Missing Interface Implementation	Coding Issue	Informational	● Resolved
NMT-01	Solidity Version 0.8.20 May Not Work On Other Chains Due To `PUSH0`	Logical Issue	Informational	● Acknowledged

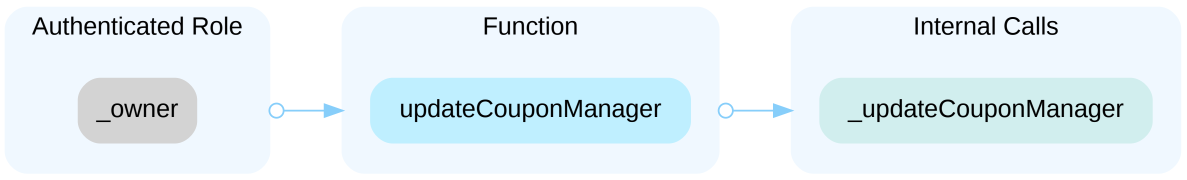
SRC-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	● **Major**	**common/Signatures.sol (commit: 4b4b72): 36; coupons/CouponManager.sol (commit: 4b4b72): 44, 49; marketplace/DecentralandMarketplaceEthereum.sol (commit: 4b4b72): 80, 86, 93, 100; marketplace/DecentralandMarketplacePolygon.sol (commit: 4b4b72): 85, 91, 97, 103, 110; marketplace/Marketplace.sol (commit: 4b4b72): 23, 28; marketplace/MarketplaceWithCouponManager.sol (commit: 4b4b72): 43**	● **Acknowledged**

Description

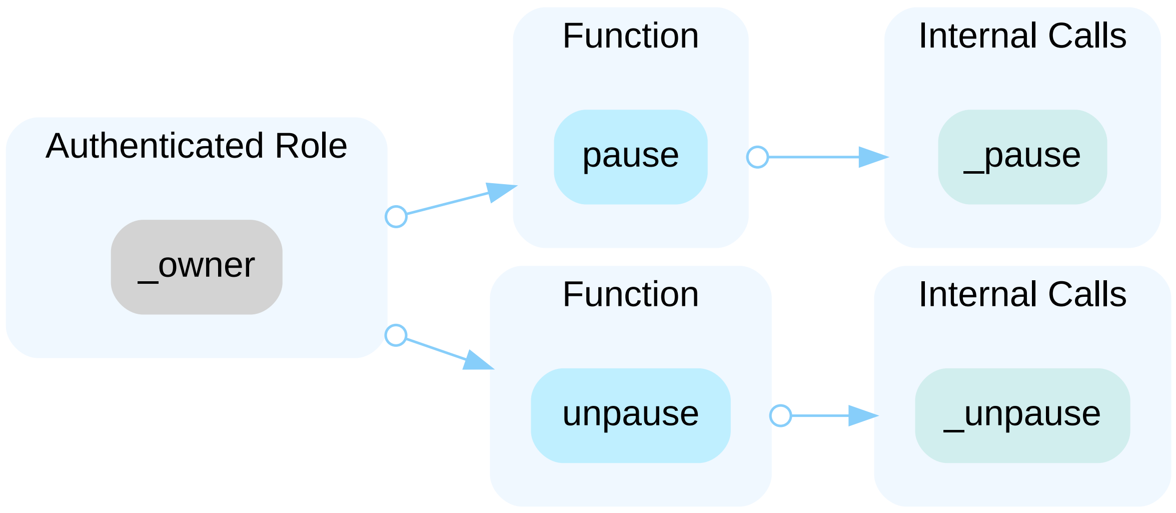
In the contract `MarketplaceWithCouponManager` the role `_owner` has authority over the function shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and:

- function `updateCouponManager()` , to update the `couponManager` .



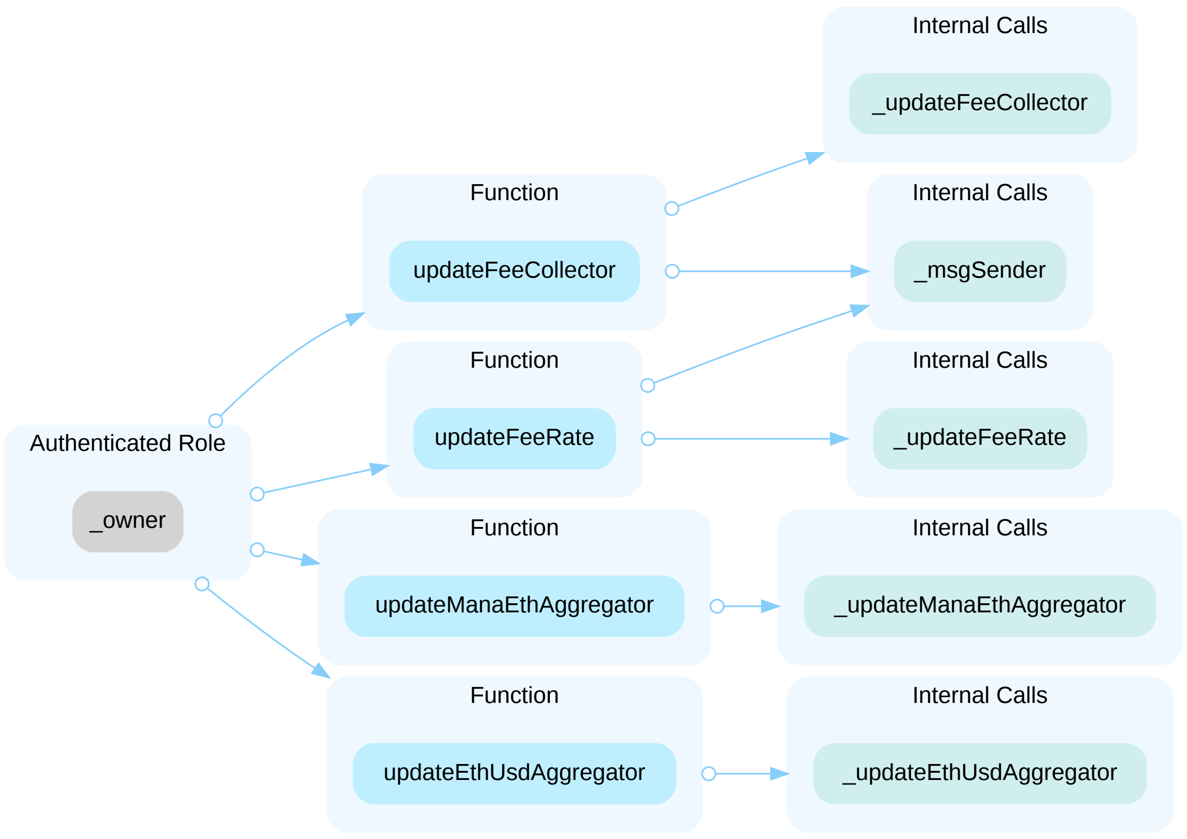
In the contract `Marketplace` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and:

- function `pause()` , to pause the contract.
- function `unpause()` , to unpause the contract.



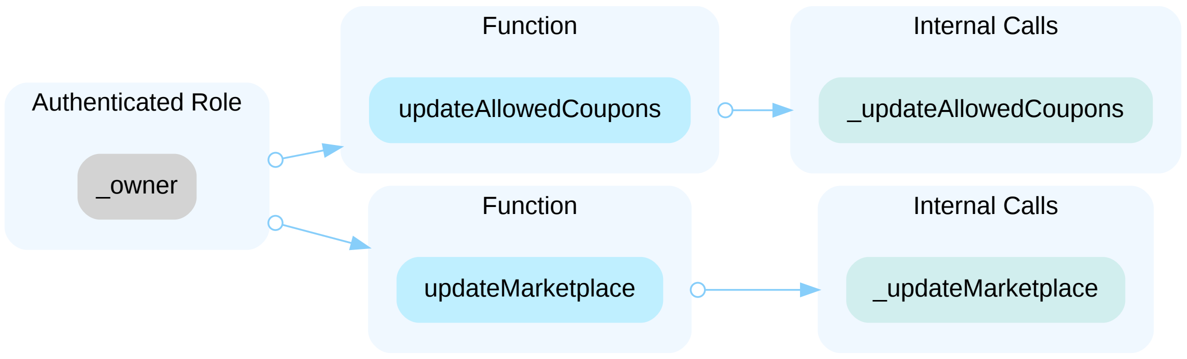
In the contract `DecentralandMarketplaceEthereum` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and:

- function `updateFeeCollector()` , to update the fee collector address.
- function `updateFeeRate()` , to update the fee rate.
- function `updateManaEthAggregator()` , to update the MANA/ETH price aggregator and tolerance.
- function `updateEthUsdAggregator()` , to update the ETH/USD price aggregator and tolerance.



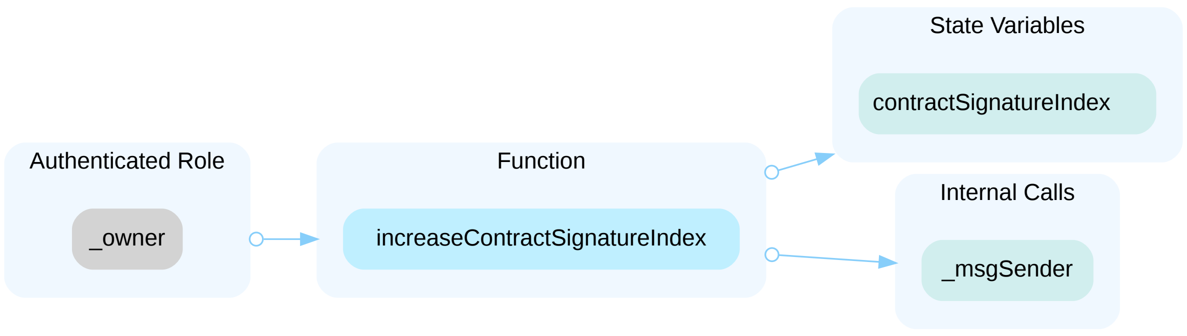
In the contract `CouponManager` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and:

- function `updateMarketplace()` , to update the address of the marketplace that will be able to apply coupons.
- function `updateAllowedCoupons()` , to update the list of allowed coupon addresses.



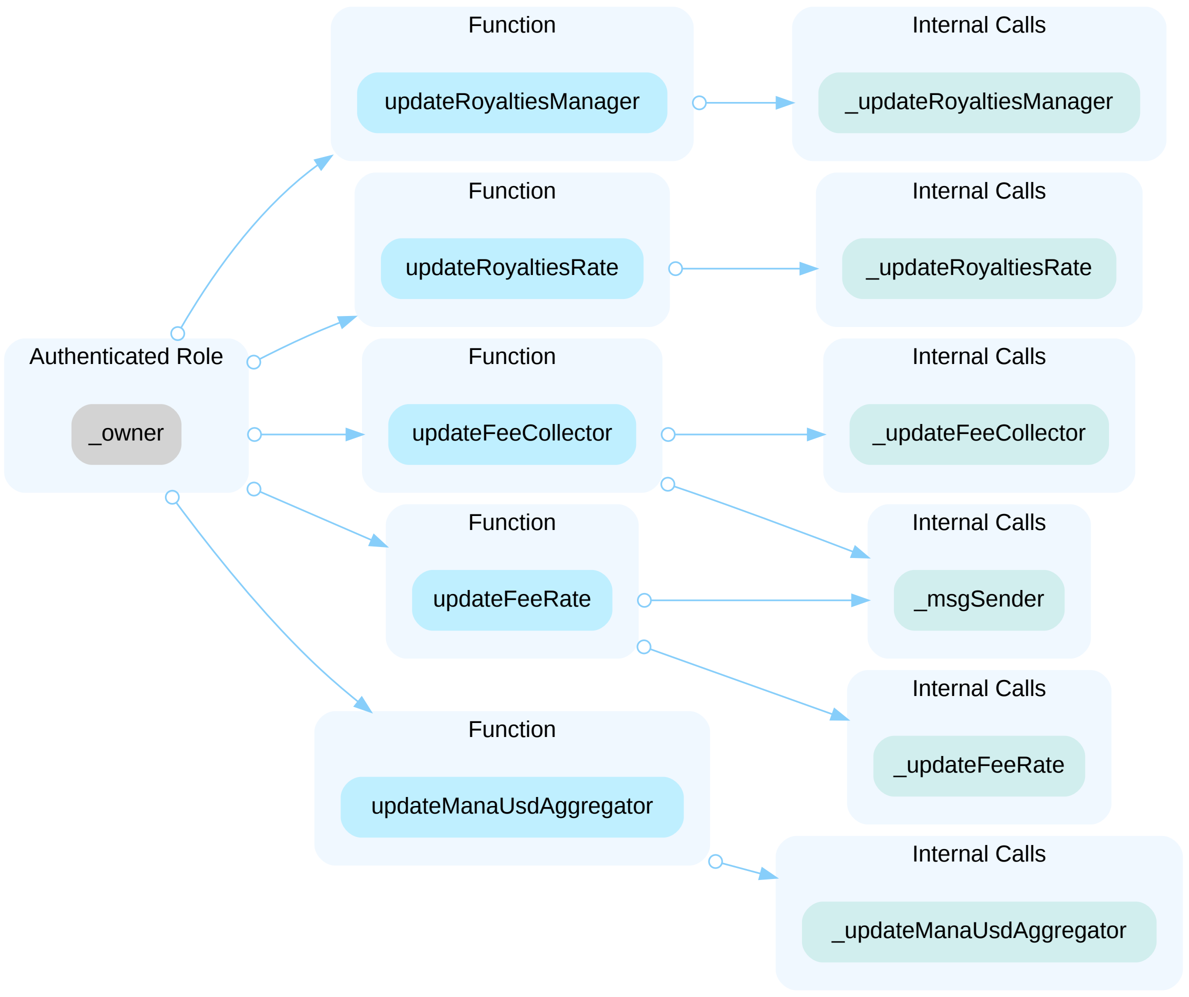
In the contract `Signatures` the role `_owner` has authority over the function shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and:

- function `increaseContractSignatureIndex()` , to increase the contract signature index. This allows the `_owner` to revoke all signatures created previously.



In the contract `DecentralandMarketplacePolygon` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and:

- function `updateFeeCollector()` , to update the fee collector address.
- function `updateFeeRate()` , to update the fee rate.
- function `updateRoyaltiesManager()` , to update the royalties manager address.
- function `updateRoyaltiesRate()` , to update the royalties rate.
- function `updateManaUsdAggregator()` , to update the MANA/USD price aggregator and tolerance.



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., multisignature 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.

Alleviation

[Decentraland Team, 08/07/2024]: The owner will be a multi-signature wallet.

[CertiK, 08/07/2024]: It is suggested to implement the aforementioned methods to avoid centralized failure. Also, CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

DMP-01 | UNFAIR FEE PAYMENT FOR DECENTRALAND NFTS IN MIXED TRADES

Category	Severity	Location	Status
Logical Issue	● Minor	marketplace/DecentralandMarketplacePolygon.sol (commit: 4b4b72): 161~166	● Acknowledged

Description

The `_modifyTrade()` function is designed to determine whether a trade should pay fees by using the `payFeeCollector` variable, which is set by the `_getFeesAndRoyalties()` function. When trades involve both Decentraland NFTs and non-Decentraland NFTs, the `payFeeCollector` flag is set to true, leading to an unfair situation where Decentraland NFTs are charged fees that they should not incur. This issue arises because `payFeeCollector` does not correctly differentiate between the types of NFTs, resulting in Decentraland NFTs being subjected to unnecessary fee payments when included in mixed trades.

```
    function _modifyTrade(Trade memory _trade) internal view override returns (Trade memory) {

        bool payFeeCollector = false;

        (payFeeCollector, royaltyBeneficiariesCount, royaltyBeneficiaries) =
            _getFeesAndRoyalties(payFeeCollector, royaltyBeneficiariesCount, royaltyBeneficiaries, _trade.sent);

        (payFeeCollector, royaltyBeneficiariesCount, royaltyBeneficiaries) =
            _getFeesAndRoyalties(payFeeCollector, royaltyBeneficiariesCount, royaltyBeneficiaries, _trade.received);

    }
    function _getFeesAndRoyalties(
        bool _payFeeCollector,
        uint256 _royaltyBeneficiariesCount,
        address[] memory _royaltyBeneficiaries,
        Asset[] memory _assets
    ) private view returns (bool, uint256, address[] memory) {
      for (uint256 i = 0; i < _assets.length; i++) {
            if (royaltyBeneficiary != address(0)) {
          _royaltyBeneficiaries[_royaltyBeneficiariesCount++] =
royaltyBeneficiary;
        } else {
// If the NFT is not a Decentraland Collection, the fee collector should be paid.
           _payFeeCollector = true;
        }
     }
  }
```

Recommendation

Adjust the `_getFeesAndRoyalties()` function to differentiate between Decentraland and non-Decentraland NFTs, ensuring only non-Decentraland NFTs trigger `payFeeCollector`.

Alleviation

[Decentraland Team, 08/07/2024]:

For the use cases we plan to give this contract, this kind of situations in which mixed NFTs are traded will not occur.

Generally they will be traded separately in the dApp. But in the case some mixed use cases arise in the future, it will be properly documented to prevent surprises.

MAR-01 | LACK OF REASONABLE UPPER BOUNDARIES ON FEES

Category	Severity	Location	Status
Logical Issue	● Minor	marketplace/DecentralandMarketplacePolygon.sol (commit: 4b4b72): 290; marketplace/FeeCollector.sol (commit: 4b4b72): 37	● Acknowledged

Description

The `royaltiesRate` and `feeRate` variables in the contracts have no enforced upper limits, which means these fees could potentially be set as high as 100%. Such high rates would result in beneficiaries receiving no tokens from the contract, leading to a total loss of their investment. This vulnerability stems from the absence of any constraints on these rates, which can be easily exploited. Ensuring that these rates are within a reasonable range is crucial to maintaining trust and fairness within the contract.

```
289    function _updateRoyaltiesRate(uint256 _royaltiesRate) internal {
290          royaltiesRate = _royaltiesRate;
291
292          emit RoyaltiesRateUpdated(_msgSender(), _royaltiesRate);
293      }
```

```
36    function _updateFeeRate(address _caller, uint256 _feeRate) internal {
37          feeRate = _feeRate;
38
39          emit FeeRateUpdated(_caller, _feeRate);
40      }
```

Recommendation

To mitigate this risk, introduce maximum limits for both `royaltiesRate` and `feeRate` to ensure that they remain within reasonable and fair boundaries, protecting users from excessive fees.

Alleviation

[Decentraland Team, 08/07/2024]: Fees are set by the owner, and variables controlled by the owner are expected to be safe and reasonable always. In this case they will reflect what is defined by the DAO.

MAR-02 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	marketplace/DecentralandMarketplaceEthereum.sol (commit: 4b4b72): 61, 72; marketplace/DecentralandMarketplacePolygon.sol (commit: 4b4b72): 66, 78	● Acknowledged

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities.

```
72          manaAddress = _manaAddress;
```

- `_manaAddress` is not zero-checked before being used.

```
78          manaAddress = _manaAddress;
```

- `_manaAddress` is not zero-checked before being used.

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[Decentraland Team, 08/07/2024]:

To reduce gas costs, we don't check on variables that are in control of Decentraland as it is expected for these to be set correctly.

In this case the address is set when deployed. On deployment it is expected to set the value as the expected one or else, redeploy another one.

SRC-02 | THIRD-PARTY DEPENDENCY USAGE

Category	Severity	Location	Status
Design Issue	● Minor	coupons/interfaces/ICouponManager.sol (commit: 4b4b72): 8~10; marketplace/DecentralandMarketplaceEthereum.sol (commit: 4b4b72): 30, 37; marketplace/DecentralandMarketplacePolygon.sol (commit: 4b4b72): 33, 39; marketplace/interfaces/IAggregator.sol (commit: 4b4b72): 5~14; marketplace/interfaces/IRoyaltiesManager.sol (commit: 4b4b72): 7~9	● Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols, such as OpenZeppelin cryptography, Chainlink Aggregator and the Off-Chain DAPP. 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. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties.

```
33      IAggregator public manaUsdAggregator;
```

- The contract `DecentralandMarketplacePolygon` interacts with third party contract with `IAggregator` interface via `manaUsdAggregator`.

```
37      IAggregator public ethUsdAggregator;
```

- The contract `DecentralandMarketplaceEthereum` interacts with third party contract with `IAggregator` interface via `ethUsdAggregator`.

```
30      IAggregator public manaEthAggregator;
```

- The contract `DecentralandMarketplaceEthereum` interacts with third party contract with `IAggregator` interface via `manaEthAggregator`.

```
39      IRoyaltiesManager public royaltiesManager;
```

- The contract `DecentralandMarketplacePolygon` interacts with third party contract with `IRoyaltiesManager` interface via `royaltiesManager`.

Recommendation

The auditors understood that the business logic requires interaction with third parties. Recommend the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Decentraland Team, 08/07/2024]: The team acknowledged the finding and decided not to change the current codebase.

SRC-03 | LACK OF REVOCATION STATUS VALIDATION IN `cancelSignature()` FUNCTION

Category	Severity	Location	Status
Logical Issue	● Minor	coupons/CouponManager.sol (commit: 4b4b72): 61; marketplace/Marketplace.sol (commit: 4b4b72): 35	● Acknowledged

Description

In the `Marketplace` contract, the `cancelSignature()` function fails to validate whether a signature has previously been cancelled. This omission could result in the improper emission of revocation events, potentially affect off-chain dependencies.

The `cancelSignature()` function in the `CouponManager` contract shares the same issue.

```
    function cancelSignature(Trade[] calldata _trades) external {
        address caller = _msgSender();

        for (uint256 i = 0; i < _trades.length; i++) {
            Trade memory trade = _trades[i];

            _verifyTradeSignature(trade, caller);

            _cancelSignature(keccak256(trade.signature));
        }
    }
```

Recommendation

Add a check that verifies whether a signature is already cancelled before proceeding with the revocation.

Alleviation

[Decentraland Team, 08/07/2024]:

The cancel signature is expected to be used when users update bids/offers etc. So it will be used many times. In order to make it cheaper, that check is ignored.

It should be properly handled off chain if a double cancelation ever occurs.

VER-01 | IMPROPER HANDLING OF SIGNATURE EXPIRATION IN `_verifyChecks()` FUNCTION

Category	Severity	Location	Status
Logical Issue	● Minor	common/Verifications.sol (commit: 4b4b72): 56~58	● Acknowledged

Description

The expiration field in the Checks struct is intended to represent the expiration date of the signature, as indicated by the code comments. In the `_verifyChecks()` function, the expiration is only checked to revert when the expiration date is passed, normally it could revert when the expiration has been reached.

```
        if (_checks.expiration < block.timestamp) {
            revert Expired();
        }
```

Recommendation

Ensure that the check will revert when the expiration timestamp is reached.

Alleviation

[Decentraland Team, 08/07/2024]: The team acknowledged the finding and decided not to change the current codebase.

CDC-01 | LIMITATION IN `applyCoupon()` FUNCTION FOR COUPON DISTRIBUTION

Category	Severity	Location	Status
Design Issue	● Informational	coupons/CollectionDiscountCoupon.sol (commit: 4b4b72): 65	● Acknowledged

Description

The `applyCoupon()` function in the `CollectionDiscountCoupon` contract is used for users to trade items with a discount. The current implementation requires that the coupon distributor be the creator of the item. This raises a concern: if the item has been traded, the new owner (not the creator) should be able to distribute a coupon to trade this item.

Recommendation

It is recommended to review the design.

Alleviation

[Decentraland Team, 08/07/2024]:

These coupons are intended for Decentraland Primary Sales. Which are the process in which Decentraland items are minted into Decentraland NFTs.

The coupon allows Decentraland Collection Creators to offer discounts for minting their collection items.

Minted Decentraland NFTs might have coupons in the future, but it is not the case for now.

COU-01 | MISSING INTERFACE IMPLEMENTATION

Category	Severity	Location	Status
Coding Issue	● Informational	coupons/CollectionDiscountCoupon.sol (commit: 4b4b72): 12~97; coupons/interfaces/ICoupon.sol (commit: 4b4b72): 8~10	● Resolved

Description

The contract `CollectionDiscountCoupon` implements the interface `ICoupon`, but does not inherit from it.

```
contract CollectionDiscountCoupon is DecentralandMarketplacePolygonAssetTypes,
 CouponTypes, MarketplaceTypes {
```

```
interface ICoupon {
    function applyCoupon(MarketplaceTypes.Trade calldata _trade, CouponTypes.
Coupon calldata _coupon) external view returns (MarketplaceTypes.Trade memory);
}
```

Recommendation

It is advised to implement the missing interface in the contract to ensure proper functionality and increase readability.

Alleviation

[Decentraland Team, 08/13/2024]: The team heeded the advice and resolved the issue in commit: c449289b2148337734620ea1f8bd0868e85dc1b8.

NMT-01 | SOLIDITY VERSION 0.8.20 MAY NOT WORK ON OTHER CHAINS DUE TO `PUSH0`

Category	Severity	Location	Status
Logical Issue	● Informational	common/NativeMetaTransaction.sol (commit: 4b4b72): 2	● Acknowledged

Description

The compiler for Solidity 0.8.20 switches the default target EVM version to Shanghai, which includes the new `PUSH0` op code. This op code may not yet be implemented on all L2s, so deployment on these chains will fail. To work around this issue, use an earlier EVM version

Recommendation

It's recommended to pay attention to the EVM complier version when using 0.8.20 solidity version in the contracts.

Alleviation

[Decentraland Team, 08/07/2024]: By far the contracts were deployed to Amoy and Sepolia without issue, so the expected Ethereum and Polygon networks should not have an issue either.

FORMAL VERIFICATION | DECENTRALAND

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of Standard Ownable Properties

We verified *partial* properties of the public interfaces of those token contracts that implement the Ownable interface. This involves:

- function `owner` that returns the current owner,
- functions `renounceOwnership` that removes ownership,
- function `transferOwnership` that transfers the ownership to a new owner.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
ownable-renounceownership-correct	Ownership is Removed
ownable-owner-succeed-normal	`owner` Always Succeeds
ownable-transferownership-correct	Ownership is Transferred

Verification Results

For the following contracts, formal verification established that each of the properties that were in scope of this audit (see scope) are valid:

Detailed Results For Contract CouponManager (src/coupons/CouponManager.sol) In Commit 8950b0941af42140d22c3d2ef344920c0b07dde3

Verification of Standard Ownable Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract DecentralandMarketplaceEthereum (src/marketplace/DecentralandMarketplaceEthereum.sol) In Commit 8950b0941af42140d22c3d2ef344920c0b07dde3

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract DecentralandMarketplacePolygon (src/marketplace/DecentralandMarketplacePolygon.sol) In Commit 8950b0941af42140d22c3d2ef344920c0b07dde3

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract CouponManager (src/coupons/CouponManager.sol) In Commit 4b4b7252698732b546e8b4f35a675beef568d284

Verification of Standard Ownable Properties

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results For Contract DecentralandMarketplaceEthereum (src/marketplace/DecentralandMarketplaceEthereum.sol) In Commit 4b4b7252698732b546e8b4f35a675beef568d284

Verification of Standard Ownable Properties

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results For Contract DecentralandMarketplacePolygon (src/marketplace/DecentralandMarketplacePolygon.sol) In Commit 4b4b7252698732b546e8b4f35a675beef568d284

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

APPENDIX | DECENTRALAND

Finding Categories

Categories	Description
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

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.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held when it was invoked.
- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.
- `constraint [cond]` - the condition `cond`, which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed Ownable Properties

Properties related to function `renounceOwnership`

ownable-renounceownership-correct

Invocations of `renounceOwnership()` must set ownership to address(0).

Specification:

```
ensures this.owner() == address(0);
```

Properties related to function `owner`

ownable-owner-succeed-normal

Function `owner` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `transferOwnership`

ownable-transferownership-correct

Invocations of `transferOwnership(newOwner)` must transfer the ownership to the `newOwner`.

Specification:

```
ensures this.owner() == newOwner;
```

DISCLAIMER | CERTIK

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.

CertiK | Securing the Web3 World

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.