Competitive Security Assessment

TevaEra-NFT-Marketplace

Aug 11th, 2023

Secure3

secure3.io

Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

- Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.
- Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.
- Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.
- Verify the code base is compliant with the most up-to-date industry standards and security best practices.
- Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

Overview

Project Detail

Project Name	TevaEra-NFT-Marketplace
Platform & Language	Solidity
Codebase	• https://github.com/tevaera-labs/contracts • audit commit - e5dd6cf2a1aa149e90b9ce6b8d9747bfeecbc33b • final commit - b01819fbbf7e5a582da9c3f06b53f5e8dd80c24e
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

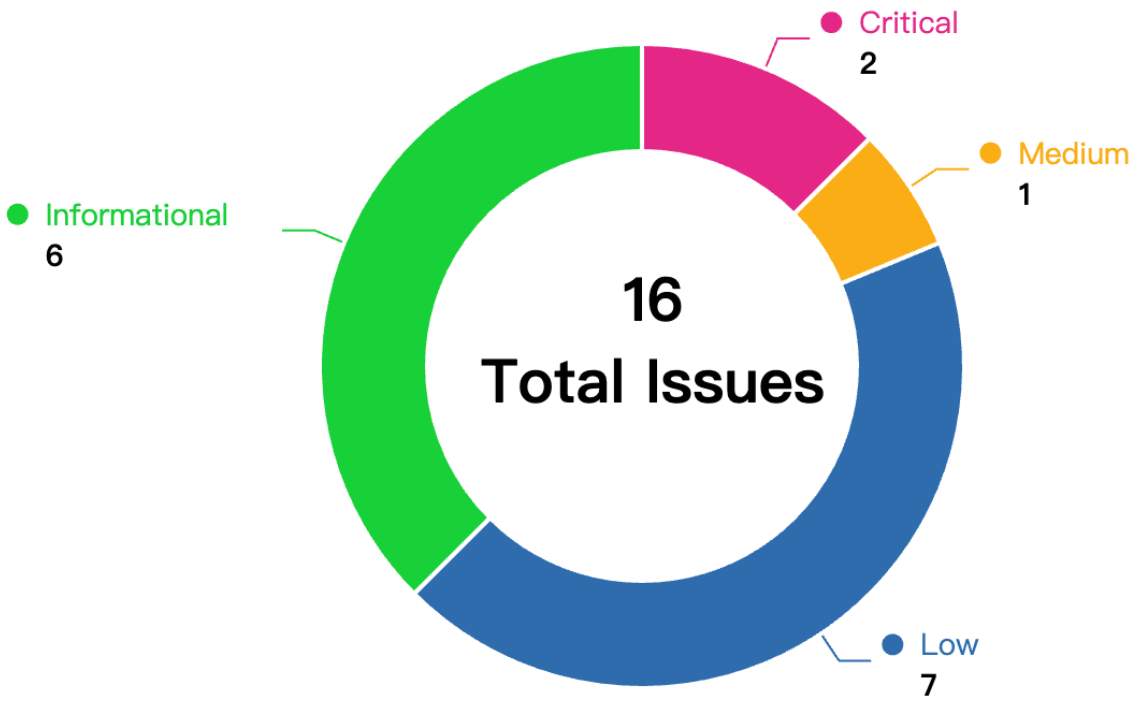
Code Vulnerability Review Summary

Vulnerability Level	Total	Reported	Acknowledged	Fixed	Mitigated	Declined
Critical	2	0	0	2	0	0
Medium	1	0	0	1	0	0
Low	7	0	2	5	0	0
Informational	6	0	6	0	0	0

Audit Scope

File	SHA256 Hash
./contracts/marketplace/MarketplaceV1.sol	eb4bd348c317bef2ee7c284c8e726198a3a5d8e5092af6b7d3bd55e0f10ec5a6
./interfaces/marketplace/IMarketplace.sol	c2a494e0cd72a5a406961238abaed08d613e813e37e3d09e2c5862d70f81a9aa
./lib/external/TWAddress.sol	1d6f334e89b3c8fede1f57e6a5a4b06938728aad5d624c7211bfefd7b6681b1c
./lib/CurrencyTransferLib.sol	5289128c893862475afa7f998803470cd17d780c3df542f574c1683c2cead925
./lib/external/SafeERC20.sol	14c58ee48c268caad5ef8768eac51f24abf4fecd992161a824db32dd83be79a8
./lib/external/ERC2771ContextUpgradeable.sol	d6975347412c7e00bc16006bbddf8aaae53ca6bc9808b08b02bf4280bda7a99e
./lib/FeeType.sol	0f935d88c60289b75a1c19846ad40e69a8eafbed72226880ba61afcc96911fbe

Code Assessment Findings



ID	Name	Category	Severity	Client Response	Contributor
TEV-1	A malicious winning bidder can make the next winning bidder spend a large amout of gas.	DOS	Critical	Fixed	jayphbee
TEV-2	Front Run Risk When Owner Accepting An Offer	Logical	Critical	Fixed	danielt
TEV-3	Error assignment logic in `MarketplaceV1::updateListing()` function	Logical	Medium	Fixed	Hupixiong3
TEV-4	Ower can set fees to close to 100%	Privilege Related	Low	Fixed	danielt

TEV-5	Whitelist for the listing tokens	DOS	Low	Acknowledged	danielt
TEV-6	The import contract is not found	Logical	Low	Acknowledged	danielt
TEV-7	Missing check the msg.value if the _currency is not the native token	Logical	Low	Fixed	jayphbee
TEV-8	Missing check the `msg.value` if the `_currency` is not the native token	Logical	Low	Fixed	danielt
TEV-9	Completed orders should be cleared in `MarketplaceV1::executeSale()` function	Code Style	Low	Fixed	Hupixiong3
TEV-10	Lack of price check in `MarketplaceV1::createListing()` function	Logical	Low	Fixed	Hupixiong3
TEV-11	Missing emit events	Code Style	Informational	Acknowledged	danielt
TEV-12	Zero address check	Language Specific	Informational	Acknowledged	danielt
TEV-13	The hosting logic doesn't make sense in `MarketplaceV1::updateListing()` function	Logical	Informational	Acknowledged	Hupixiong3
TEV-14	Restrict the safeQuantity != 0 in `MarketplaceV1::getSafeQuantity()` function	Code Style	Informational	Acknowledged	Hupixiong3
TEV-15	supportsInterface() miss super.supportsInterface() call.	Logical	Informational	Acknowledged	jayphbee
TEV-16	restrict msg.sender == nativeTokenWrapper in receive() function.	Logical	Informational	Acknowledged	jayphbee

TEV-1:A malicious winning bidder can make the next winning bidder spend a large amout of gas.

Category	Severity	Client Response	Contributor
DOS	Critical	Fixed	jayphbee

Code Reference

- code/lib/CurrencyTransferLib.sol#L96

```
96:(bool success, ) = to.call{ value: value }("");
```

Description

jayphbee : The auction mechanism will guarantee that the previous winning bidder's balance will be returned back.

```
        // Payout previous highest bid.
        if (currentWinningBid.offeror != address(0) && currentOfferAmount > 0) {
            CurrencyTransferLib.transferCurrencyWithWrapper(
                _targetListing.currency,
                address(this),
                currentWinningBid.offeror,
                currentOfferAmount,
                _nativeTokenWrapper
            );
        }
```

The previous winning bidder can be malicious contract account and eat all gas forwarded from the caller by the low level call if the `_currency` is native token.

```
    /// @dev Transfers `amount` of native token to `to`. (With native token wrapping)
    function safeTransferNativeTokenWithWrapper(
        address to,
        uint256 value,
        address _nativeTokenWrapper
    ) internal {
        // solhint-disable avoid-low-level-calls
        // slither-disable-next-line low-level-calls
        (bool success, ) = to.call{ value: value }(""); // @audit-issue consume all forwarded gas
        if (!success) {
            IWETH(_nativeTokenWrapper).deposit{ value: value }();
            IERC20(_nativeTokenWrapper).safeTransfer(to, value);
        }
    }
```

Here is a proof of concept how much gas a malicious winning bidder can consume at the most.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

import "forge-std/Test.sol";
import "forge-std/console2.sol";

interface WETH9 {
    function deposit() external payable;
    function transfer(address dst, uint256 wad) external returns (bool);
}

contract Attacker {
    fallback() external {
        assembly {
            mstore(10000000000000000, 1) // access large memory offset to consume all gas
        }
    }
}

contract Attack is Test {
    Attacker attacker;
    function setUp() public {
        attacker = new Attacker();
    }
    function testAttack() public {
        (bool success, ) = address(attacker).call("");
        if (!success) {
            WETH9(0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2).deposit{value: 10}();
            WETH9(0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2).transfer(address(attacker), 10);
        }
    }
}
```

We run the test with `gas-limit=30000000`.

```
forge test --mt testAttack -vvvv --rpc-url=$mainnet --gas-limit 30000000

Traces:
  [29549964] Attack::testAttack()
    ├─ [51] Attacker::fallback()
    │   └─ ← "EvmError: MemoryLimitOOG"
    ├─ [23974] WETH9::deposit{value: 10}()
    │   ├─ emit Deposit(dst: Attack: [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496], wad: 10)
    │   └─ ← ()
    ├─ [20130] WETH9::transfer(Attacker: [0x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f], 10)
    │   ├─ emit Transfer(from: Attack: [0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496], to: Attacker:
[0x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f], value: 10)
    │   └─ ← true
    └─ ← ()
```

We can see that the `Attacker` contract consumes almost all the gas(29549964), but the remaining code can be run due to the 63/64 gas rule.

This will lead to the new bidder unwilling to make a new bid due to the unreasonable gas usage, the previous malicious winning bidder can exploit this to become the final auction winner.

Recommendation

jayphbee : Add a gas limit to the low level call to restrict the gas usage and i think 20000 is reasonable.

```
(bool success, ) = to.call{ value: value, gas: 20000 }("");
```

Client Response

Fixed.

TEV-2:Front Run Risk When Owner Accepting An Offer

Category	Severity	Client Response	Contributor
Logical	Critical	Fixed	danielt

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L536-L550

```
536:} else if (targetListing.listingType == ListingType.Direct) {
537:            // Prevent potentially lost/locked native token.
538:            require(msg.value == 0, "no value needed");
539:
540:            // Offers to direct listings cannot be made directly in native tokens.
541:            newOffer.currency = _currency == CurrencyTransferLib.NATIVE_TOKEN
542:                ? nativeTokenWrapper
543:                : _currency;
544:            newOffer.quantityWanted = getSafeQuantity(
545:                targetListing.tokenType,
546:                _quantityWanted
547:            );
548:
549:            handleOffer(targetListing, newOffer);
550:        }
```

Description

danielt : Users make offers to a direct listing by invoking the `offer` function:

```
    function offer(
        uint256 _listingId,
        uint256 _quantityWanted,
        address _currency,
        uint256 _pricePerToken,
        uint256 _expirationTimestamp
    ) external payable override nonReentrant onlyExistingListing(_listingId) {
        ...
        } else if (targetListing.listingType == ListingType.Direct) {
            // Prevent potentially lost/locked native token.
            require(msg.value == 0, "no value needed");

            // Offers to direct listings cannot be made directly in native tokens.
            newOffer.currency = _currency == CurrencyTransferLib.NATIVE_TOKEN
                ? nativeTokenWrapper
                : _currency;
            newOffer.quantityWanted = getSafeQuantity(
                targetListing.tokenType,
                _quantityWanted
            );

            handleOffer(targetListing, newOffer);
        }
    }
```

The `offer` function allows users to update their offers, which lacks checking if the `totalOfferAmount` of the updated offer is greater or not.

As a result, before the owner of a listing accepts an offer for a target user, like Alice. Alice monitors the owner's transaction and can front-run the owner's transaction and hugely decrease Alice's offer's `totalOfferAmount`, which makes the owner a big token loss.

Recommendation

danielt : Consider checking the `totalOfferAmount` when users update their offers.

Client Response

Fixed.

TEV-3:Error assignment logic in `MarketplaceV1::updateListing()` function

Category	Severity	Client Response	Contributor
Logical	Medium	Fixed	Hupixiong3

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L311-L313

```
311:uint256 newStartTime = _startTime == 0
312:            ? targetListing.startTime
313:            : _startTime;
```

Description

Hupixiong3 : When the newStartTime variable is assigning, _startTime will never be 0.When _startTime is 0, it does not pass the (block.timestamp - _startTime < 1 hours, "ST") check

Recommendation

Hupixiong3 : Modify the assignment logic.

Client Response

Fixed.

TEV-4:Ower can set fees to close to 100%

Category	Severity	Client Response	Contributor
Privilege Related	**Low**	**Fixed**	**danielt**

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L836-L840
- code/contracts/marketplace/MarketplaceV1.sol#L879-L885
- code/contracts/marketplace/MarketplaceV1.sol#L1045-L1062

```
836:uint256 platformFeeBps = citizenIdContract.balanceOf(msg.sender) > 0
837:            ? tevanPlatformFeeBps
838:            : nonTevanPlatformFeeBps;
839:        uint256 platformFeeCut = (_totalPayoutAmount * platformFeeBps) /
840:            MAX_BPS;

879:CurrencyTransferLib.transferCurrencyWithWrapper(
880:            _currencyToUse,
881:            _payer,
882:            _payee,
883:            _totalPayoutAmount - (platformFeeCut + royaltyCut),
884:            _nativeTokenWrapper
885:        );

1045:function setPlatformFeeInfo(
1046:        address _platformFeeRecipient,
1047:        uint256 _tevanPlatformFeeBps,
1048:        uint256 _nonTevanPlatformFeeBps
1049:    ) external onlyOwner {
1050:        require(_tevanPlatformFeeBps <= MAX_BPS, "bps <= 10000.");
1051:        require(_nonTevanPlatformFeeBps <= MAX_BPS, "bps <= 10000.");
1052:
1053:        platformFeeRecipient = _platformFeeRecipient;
1054:        tevanPlatformFeeBps = uint64(_tevanPlatformFeeBps);
1055:        nonTevanPlatformFeeBps = uint64(_nonTevanPlatformFeeBps);
1056:
1057:        emit PlatformFeeInfoUpdated(
1058:            _platformFeeRecipient,
1059:            _tevanPlatformFeeBps,
1060:            _nonTevanPlatformFeeBps
1061:        );
1062:    }
```

Description

danielt : The `payout` function will distribute fees to fee recipients. However, the fee percentage could be set as higher as 100%, which may lead the payee to get 0 tokens.

```
    function payout(
        address _payer,
        address _payee,
        address _currencyToUse,
        uint256 _totalPayoutAmount,
        Listing memory _listing
    ) internal {
        uint256 platformFeeBps = citizenIdContract.balanceOf(msg.sender) > 0
            ? tevanPlatformFeeBps
            : nonTevanPlatformFeeBps;
        uint256 platformFeeCut = (_totalPayoutAmount * platformFeeBps) /
            MAX_BPS;
        ...
        CurrencyTransferLib.transferCurrencyWithWrapper(
            _currencyToUse,
            _payer,
            _payee,
            _totalPayoutAmount - (platformFeeCut + royaltyCut),
            _nativeTokenWrapper
        );
    }

    function setPlatformFeeInfo(
        address _platformFeeRecipient,
        uint256 _tevanPlatformFeeBps,
        uint256 _nonTevanPlatformFeeBps
    ) external onlyOwner {
        require(_tevanPlatformFeeBps <= MAX_BPS, "bps <= 10000.");
        require(_nonTevanPlatformFeeBps <= MAX_BPS, "bps <= 10000.");

        platformFeeRecipient = _platformFeeRecipient;
        tevanPlatformFeeBps = uint64(_tevanPlatformFeeBps);
        nonTevanPlatformFeeBps = uint64(_nonTevanPlatformFeeBps);
        ...
    }
```

Recommendation

danielt : Consider setting reasonable boundaries for the fees.

Client Response

Fixed.

TEV-5:Whitelist for the listing tokens

Category	Severity	Client Response	Contributor
DOS	Low	Acknowledged	danielt

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L210

```
210:function createListing(ListingParameters memory _params) external override {
```

Description

danielt : There is no token whitelist when creating a listing in the `createListing` function. As a result, potential malicious tokens can also be listed in the `MarketplaceV1` contract.

Taking the below malicious tokens into consideration:

- What if the listed token has a pausable function and the owner pauses the transfer function expectedly;
- What if the token has a high royalty fee, like 90%;
- What if the token contact on the chain has not been verified.

Recommendation

danielt : It is highly recommended to add the whitelist function for the listing tokens.

Client Response

Acknowledged.This is intentional, as we aim to display all ERC721 and ERC1155 tokens in the marketplace without any intervention from our side as the owner. To address concerns regarding malicious contracts, we have implemented a contract approval feature on the website. This feature provides users with an idea of genuine contracts. However, users are free to engage in trades involving unapproved contracts at their own risk.

TEV-6:The import contract is not found

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	danielt

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L29

```
29:import "../citizenid/CitizenIDV2.sol";
```

Description

danielt : The `MarketplaceV1` contract imports the `../citizenid/CitizenIDV2.sol` contract, which is not found. Importing a non-existed file will not pass the compile.

Recommendation

danielt : Make sure the imported file exists and pass the compile.

Client Response

Acknowledged.The code is available in the repository. It was audited as part of a previous audit, so it may not be listed in the audit files, but it can be found in the repository.

TEV-7:Missing check the msg.value if the _currency is not the native token

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	jayphbee

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L972-L976

```
972:if (_currency == CurrencyTransferLib.NATIVE_TOKEN) {
973:            require(msg.value == settledTotalPrice, "msg.value != price");
974:        } else {
975:            validateERC20BalAndAllowance(_payer, _currency, settledTotalPrice);
976:        }
```

Description

jayphbee : Both native and ERC20 tokens can be used to buy NFT in the marketplace. msg.value should be restrict to zero when user buy the NFT with ERC20 token, otherwise user may accidently send ether when calling `buy` function becasue it is payable.

```
        // Check: buyer owns and has approved sufficient currency for sale.
        if (_currency == CurrencyTransferLib.NATIVE_TOKEN) {
            require(msg.value == settledTotalPrice, "msg.value != price");
        } else {// @audit-issue restrict msg.value == 0
            validateERC20BalAndAllowance(_payer, _currency, settledTotalPrice);
        }
```

Recommendation

jayphbee : restrict msg.value == 0

```
        // Check: buyer owns and has approved sufficient currency for sale.
        if (_currency == CurrencyTransferLib.NATIVE_TOKEN) {
            require(msg.value == settledTotalPrice, "msg.value != price");
        } else {
            require(msg.value == 0, "unexpected ether send");
            validateERC20BalAndAllowance(_payer, _currency, settledTotalPrice);
        }
```

Client Response

Fixed.

TEV-8:Missing check the `msg.value` if the `_currency` is not the native token

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	danielt

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L972-L976

```
972:if (_currency == CurrencyTransferLib.NATIVE_TOKEN) {
973:            require(msg.value == settledTotalPrice, "msg.value != price");
974:        } else {
975:            validateERC20BalAndAllowance(_payer, _currency, settledTotalPrice);
976:        }
```

Description

danielt : The `validateDirectListingSale` function missing check the `msg.value` when the `_currency` is not the native token:

```
        if (_currency == CurrencyTransferLib.NATIVE_TOKEN) {
            require(msg.value == settledTotalPrice, "msg.value != price");
        } else {
            validateERC20BalAndAllowance(_payer, _currency, settledTotalPrice);
        }
```

Do the check on `msg.value` to prevent potentially lost/locked native token if the `_currency` is not the native token is required, because `validateDirectListingSale` will be invoked by a payable function `buy()`

Recommendation

danielt : Recommend adding the check to ensure the `msg.value` to be zero when the `_currency` is not the native token.

Client Response

Fixed.

TEV-9:Completed orders should be cleared in `MarketplaceV1::executeSale()` function

Category	Severity	Client Response	Contributor
Code Style	Low	Fixed	Hupixiong3

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L464-L465

```
464:_targetListing.quantity -= _listingTokenAmountToTransfer;
465:        listings[_targetListing.listingId] = _targetListing;
```

Description

Hupixiong3 : When _targetListing.quantity is 0, it indicates that the order has been completed, and the order information should be cleaned in time to free memory.

Recommendation

Hupixiong3 : Timely clearance of completed orders.

Client Response

Fixed.

TEV-10:Lack of price check in `MarketplaceV1::createListing()` function

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	Hupixiong3

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L257-L270

```
257:if (newListing.listingType == ListingType.Auction) {
258:            require(
259:                newListing.buyoutPricePerToken == 0 ||
260:                    newListing.buyoutPricePerToken >=
261:                    newListing.reservePricePerToken,
262:                "RESERVE"
263:            );
264:            transferListingTokens(
265:                tokenOwner,
266:                address(this),
267:                tokenAmountToList,
268:                newListing
269:            );
270:        }
```

Description

Hupixiong3 : The price of the order is set to 0.For direct listings:buyoutPricePerToken must be greater than 0 .For auctions:reservePricePerToken must be greater than 0.

Recommendation

Hupixiong3 : Add price check.

Client Response

Fixed.

TEV-11:Missing emit events

Category	Severity	Client Response	Contributor
Code Style	Informational	Acknowledged	danielt

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L116-L145

```
116:constructor(address _nativeTokenWrapper) initializer {
117:        nativeTokenWrapper = _nativeTokenWrapper;
118:    }
119:
120:    /// @dev Initiliazes the contract, like a constructor.
121:    function initialize(
122:        string memory _contractURI,
123:        address[] memory _trustedForwarders,
124:        address _platformFeeRecipient,
125:        uint256 _tevanPlatformFeeBps,
126:        uint256 _nonTevanPlatformFeeBps,
127:        CitizenIDV2 _citizenIdContract
128:    ) external initializer {
129:        // Initialize inherited contracts, most base-like -> most derived.
130:        __Pausable_init();
131:        __ReentrancyGuard_init();
132:        __Ownable_init();
133:        __ERC2771Context_init(_trustedForwarders);
134:
135:        // Initialize this contract's state.
136:        timeBuffer = 15 minutes;
137:        bidBufferBps = 500;
138:
139:        contractURI = _contractURI;
140:        platformFeeRecipient = _platformFeeRecipient;
141:        tevanPlatformFeeBps = uint64(_tevanPlatformFeeBps);
142:        nonTevanPlatformFeeBps = uint64(_nonTevanPlatformFeeBps);
143:
144:        citizenIdContract = _citizenIdContract;
145:    }
```

Description

danielt : An update of key state variables in functions is recommended to emit events for them. Like the initialize of `nativeTokenWrapper`, `platformFeeRecipient`, etc.

Example:

- `setContractURI()`

Recommendation

danielt : Emit events for the update of state variables.

Client Response

Acknowledged.Deferring it this time

TEV-12:Zero address check

Category	Severity	Client Response	Contributor
Language Specific	Informational	Acknowledged	danielt

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L116-L118
- code/contracts/marketplace/MarketplaceV1.sol#L121-L145

```
116:constructor(address _nativeTokenWrapper) initializer {
117:        nativeTokenWrapper = _nativeTokenWrapper;
118:    }

121:function initialize(
122:        string memory _contractURI,
123:        address[] memory _trustedForwarders,
124:        address _platformFeeRecipient,
125:        uint256 _tevanPlatformFeeBps,
126:        uint256 _nonTevanPlatformFeeBps,
127:        CitizenIDV2 _citizenIdContract
128:    ) external initializer {
129:        // Initialize inherited contracts, most base-like -> most derived.
130:        __Pausable_init();
131:        __ReentrancyGuard_init();
132:        __Ownable_init();
133:        __ERC2771Context_init(_trustedForwarders);
134:
135:        // Initialize this contract's state.
136:        timeBuffer = 15 minutes;
137:        bidBufferBps = 500;
138:
139:        contractURI = _contractURI;
140:        platformFeeRecipient = _platformFeeRecipient;
141:        tevanPlatformFeeBps = uint64(_tevanPlatformFeeBps);
142:        nonTevanPlatformFeeBps = uint64(_nonTevanPlatformFeeBps);
143:
144:        citizenIdContract = _citizenIdContract;
145:    }
```

Description

danielt : Zero addresses assigned to address-type variables will result in unexpected results, like `platformFeeRecipient`, `citizenIdContract`, etc.

```
    constructor(address _nativeTokenWrapper) initializer {
        nativeTokenWrapper = _nativeTokenWrapper;
    }
```

Recommendation

danielt : Adding zero address check for address-type state variables.

Client Response

Acknowledged.Deferring it this time

TEV-13:The hosting logic doesn't make sense in `MarketplaceV1::updateListing()` function

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	Hupixiong3

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L332-L361

```
332:if (targetListing.quantity != safeNewQuantity) {
333:            // Transfer all escrowed tokens back to the lister, to be reflected in the lister's
334:            // balance for the upcoming ownership and approval check.
335:            if (isAuction) {
336:                transferListingTokens(
337:                    address(this),
338:                    targetListing.tokenOwner,
339:                    targetListing.quantity,
340:                    targetListing
341:                );
342:            }
343:
344:            validateOwnershipAndApproval(
345:                targetListing.tokenOwner,
346:                targetListing.assetContract,
347:                targetListing.tokenId,
348:                safeNewQuantity,
349:                targetListing.tokenType
350:            );
351:
352:            // Escrow the new quantity of tokens to list in the auction.
353:            if (isAuction) {
354:                transferListingTokens(
355:                    targetListing.tokenOwner,
356:                    address(this),
357:                    safeNewQuantity,
358:                    targetListing
359:                );
360:            }
361:        }
```

Description

Hupixiong3 : When the user updates an order, targetListing.quantity is smaller than safeNewQuantity, the user's remaining assets should be checked and supplemented; safeNewQuantity is smaller than targetListing.quantity, the user's excess assets should be returned. Rather than return all the assets of the user and then supplement.

Recommendation

Hupixiong3 : Modify the order update logic.

Client Response

Acknowledged.Deferring it this time

TEV-14:Restrict the safeQuantity != 0 in `MarketplaceV1::getSafeQuantity()` function

Category	Severity	Client Response	Contributor
Code Style	Informational	Acknowledged	Hupixiong3

Code Reference

-code/contracts/marketplace/MarketplaceV1.sol#L997-L999

```
997:if (_quantityToCheck == 0) {
998:            safeQuantity = 0;
999:        } else {
```

Description

Hupixiong3 : The getSafeQuantity function is used to return the number of tokens valid for transactions, however 0 is meaningless and should be restricted in the getSafeQuantity function.Other functions check that the getSafeQuantity function returns a value other than 0, but you should check it directly in the getSafeQuantity function.

Recommendation

Hupixiong3 : Check that the return value is not 0 directly in the getSafeQuantity function.

Client Response

Acknowledged.Deferring it this time

TEV-15:supportsInterface() miss super.supportsInterface() call.

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	jayphbee

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L197-L203

```
197:function supportsInterface(
198:        bytes4 interfaceId
199:    ) public view virtual override(IERC165Upgradeable) returns (bool) {
200:        return
201:            interfaceId == type(IERC1155ReceiverUpgradeable).interfaceId ||
202:            interfaceId == type(IERC721ReceiverUpgradeable).interfaceId;
203:    }
```

Description

jayphbee : In the `supportsInterface()` function, the `super.supportsInterface()` is missed.

Recommendation

jayphbee : Add the missing `super.supportsInterface()` call in the `supportsInterface` function.

Client Response

Acknowledged.Deferring it this time

TEV-16:restrict msg.sender == nativeTokenWrapper in receive() function.

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	jayphbee

Code Reference

- code/contracts/marketplace/MarketplaceV1.sol#L152

```
152:receive() external payable {}
```

Description

jayphbee : `receive()` function in MarketplaceV1.sol is solely used to receive ether from `nativeTokenWrapper`, so restrict `msg.sender == nativeTokenWrapper` can avoid user accidently sending ether to the Marketplace contract.

Recommendation

jayphbee : restrict msg.sender == nativeTokenWrapper in receive() function.

Client Response

Acknowledged.Deferring it this time

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 Invoices, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. 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 Secure3’s prior written consent in each instance.

This report is not an “endorsement” or “disapproval” of any particular project or team. This report is not an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3’s position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. 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.