invozone

InvoAudit

SMART CONTRACT AUDIT REPORT

for

InvoBlox NFT Minter 1155 Contract

Prepared By: Muhammad Talha

InvoAudit

March 2, 2023

Document Properties

Client	InvoBlox
Title	Smart Contract Audit Report
Target	NFT Minter 1155 Contract
Version	1.0
Author	Muhammad Talha
Auditors	Muhammad Talha
Reviewed by	Auditing Team
Approved by	Auditing Team
Classification	Public

Version Properties

Version	Date	Author(s)	Description
1.0	March 2, 2023	Muhammad Talha	Final Release
1.0-rc	February 24, 2023	Muhammad Talha	Release Candidate

Contact

Name	InvoAudit
Phone	+98798-4567
Email	InvoAudits@invoblox.com

1 | Introduction

Given the opportunity to review the design document and related smart contract source code of Auditchain, we outline in this report our systematic approach to evaluate potential security issues in the smart contract implementation, expose possible semantic inconsistencies between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contract can be further improved due to the presence of several issues. This document outlines our audit results.

1.1 About NFT Minter 1155.

ERC1155 is a multi-token standard that allows the creation of fungible, non-fungible, and semi-fungible tokens all in one contract. Before ERC1155, if a use case needed both ERC20 (fungible) and ERC721 (non-fungible) tokens, then separate contracts were required to achieve this. ERC1155 also allows for multiple NFT collections to be launched in just one smart contract instead of creating a different contract for each collection; this increases efficiency in smart contract construction and minimizes the transaction count, which is very important as it consumes less blockchain space. With ERC1155, batch transfer of tokens is also possible instead of transferring a token to a single address in previous standards.

In the following, we show the Git repository of reviewed files and the commit hash value used in this audit.

- https://gitlab.invozone.com/invoblox/stake-game-dapp/smart-contracts.git

And this is the commit ID after all fixes for the issues found in the audit have been checked in:

- https://gitlab.invozone.com/invoblox/stake-game-dapp/smart-contracts.git

1.2 About InvoAudit

InvoAudit Inc. is a leading blockchain security company with the goal of elevating the security, privacy, and usability of current blockchain ecosystems by offering top-notch, industry-leading services and products (including the service of smart contract auditing).

1.3 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology:

- Likelihood represents how likely a particular vulnerability is to be uncovered and exploited in the wild;

- Impact measures the technical loss and business damage of a successful attack;

- Severity demonstrates the overall criticality of the risk.

  Likelihood and impact are categorized into three ratings: H, M and L, i.e., high, medium and low respectively. Severity is determined by likelihood and impact and can be classified into four categories accordingly, i.e., Critical, High, Medium,

  To evaluate the risk, we go through a list of check items and each would be labeled with a severity category. For one check item, if our tool or analysis does not identify any issue, the contract is considered safe regarding the check item. For any discovered issue, we might further deploy contracts on our private testnet and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation.

In particular, we perform the audit according to the following procedure:

- Basic Coding Bugs: We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.

- Semantic Consistency Checks: We then manually check the logic of implemented smart con- tracts and compare with the description in the white paper.

- Advanced DeFi Scrutiny: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

- Additional Recommendations: We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

- To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699), which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development. Though some categories used in CWE-699 may not be relevant in smart contracts.

1.4 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release, and does not give any warranties on finding all possible security issues of the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contract(s). Last but not least, this security audit should not be used as investment advice.

1.5 Vulnerability Classifications

Vulnerability Classes	
Class	**Description**
Access Controls	Related to authorization of users and assessment of rights
Auditing and Logging	Related to auditing of actions or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices or software
Cryptography	Related to protecting the privacy or integrity of data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to race conditions, locking or order of operations
Undefined Behavior	Related to undefined behavior triggered by the program

2 | Findings

2.1 Summary

Here is a summary of our findings after analyzing the NFt 1155 Minting implementation. During the first phase of our audit, we study the smart contract source code and run our in-house static code analyzer through the codebase. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by our tool. We further manually review business logic, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No of Findings
Critical	
High	
Medium	
Low	
Informational	
Suggestions	
Total	

We have so far identified a list of potential issues: some of them involve subtle corner cases that might not be previously thought of, while others refer to unusual interactions among multiple contracts. For each uncovered issue, we have therefore developed test cases for reasoning, reproduc- tion, and/or verification. After further analysis and internal discussion, we determined a few issues of varying severities that need to be brought up and paid more attention to, which are categorized in the above table. More information can be found in the next subsection.

2.2 Key Findings

Overall, these smart contracts are well-designed and engineered, though the implementation can be improved by resolving the identified issues.

- 1 critical-severity vulnerability
- 2 medium-severity vulnerabilities
- 2 low-severity vulnerabilities.

ID	Severity	Title	Category	Status
001	High	3.1: _ Addition of a require check for total no of approval to add a new owner.	Error Reporting	
002	High	3.2: _ Addition of a require check for total no of approval for removing an owner.	Error Reporting	
003	High	3.3: _ Add require check _ to address != 0	Error Reporting	
004	High	3.4: _ Add a require check for that _ data != 0	Error Reporting	
005	Low	3.5: _ No need to define the value of variables with default values.	Patching	
006	Low	3.6: _ Removal of extra getOwners function.	Patching	
007	Low	3.7: _ Removal of extra getTransaction function.	Patching	
008	Low	3.8: _ Remove public bool	Patching	

		confirmed state variable.		
009	Medium	3.9: _Define the visibility of the totalTransaction state variable.	Patching	
010	Low	3.10: _Removed this extra isExistOwner mapping.	Patching	
011	Low	3.11: _Change all these mappings into single mapping against a struct.	Data Validation	
012	High	3.12: _Add a require statement that _owner address != 0 in addNewOwner.	Error Reporting	
013	Medium	3.13: _Remove this extra !isExistOwner[msg.sender] require statement in addNewOwner(address _owner) method.	Patching	
014	Medium	3.14: _Remove this isExistOwner[msg.sender] = true statement.	Patching	
015	High	3.15: _ “confirmationsForAddOwner" should be updated in addNewOwner method.	Error Reporting	
016	High	3.16: _Major new owner addition case handling is missing.	Denial of Service	
017	Medium	3.17: _Add new function to set numConfirmationsRequired.	Denial of Service	
018	High	3.18: _Logical error in removeOwner(uint256	Denial of Service	

		_ownerIndex) method.		
019	High	3.19:_ Logical error in removeOwner(uint256 _ownerIndex) method.	Denial of Service	
020	High	3.20:_ Logical error in removeOwner(uint256 _ownerIndex) method.	Denial of Service	
021	High	3.2:_ Addition of a require check for total no of approval for removing an owner.	Error Reporting	
022	High	3.3:_ Add require check _to address != 0	Error Reporting	
023	High	3.4:_ Add a require check for that _data != 0	Error Reporting	
024	Low	3.5:_ No need to define the value of variables with default values.	Patching	
025	Low	3.6:_ Removal of extra getOwners function.	Patching	
026	Low	3.7:_ Removal of extra getTransaction function.	Patching	
027	Low	3.8:_ Remove public bool confirmed state variable.	Patching	
028	Medium	3.9:_ Define the visibility of the totalTransaction state variable.	Patching	
029	Low	3.10:_ Removed this extra isExistOwner mapping.	Patching	
030	Low	3.11:_ Change all these	Data	

		mappings into single mapping against a struct.	Validation	
031	High	3.12: _Add a require statement that _owner address != 0 in addNewOwner.	Error Reporting	
032	Medium	3.13: _Remove this extra !isExistOwner[msg.sender] require statement in addNewOwner(address _owner) method.	Patching	
033	Medium	3.14: _Remove this isExistOwner[msg.sender] = true statement.	Patching	
034	High	3.15: _ “confirmationsForAddOwner" should be updated in addNewOwner method.	Error Reporting	
035	High	3.16: _Major new owner addition case handling is missing.	Denial of Service	
036	Medium	3.17: _Add new function to set numConfirmationsRequired.	Denial of Service	
037	High	3.18: _Logical error in removeOwner(uint256 _ownerIndex) method.	Denial of Service	
038	High	bool nftMinter; // remove this extra bool variable in NFt struct	Denial of Service	
039	High	bool nfl Existed; // remove this bool variable! in struct Nft	Error Reporting	

040	High	// token Id must be used as a state variable instead of taking to every single minter.	Error Reporting	
041	High	the creator address couldn't be set by every user. they can set their own addresses in lazy minting Method	Error Reporting	
042	Low	require(amount != 0, "NFT amount cannot be zero"); // add this require statement in the already defined check statement in Lazyminting method	Patching	
043	Low	require(msg.sender != address(0), "NFT amount cannot be zero"); // remove this extra check! in Lazyminting method	Patching	
044	Low	// remove this require statement checking (nft[tokenId].nfl Existed == false) in Lazyminting method	Patching	
045	Low	nft[tokenId].nftMinter = true; // remove this extra value assigning code line in Lazyminting method	Patching	
046	Medium	nft[tokenId].nfl Existed = true; // remove this extra value assigning code line in Lazyminting method	Patching	

047	Low	payable(creator).transfer(msg.value); // remove this transfer code line and add withdraw method for receiver person in Lazyminting method	Patching	
048	Low	require(amount <= 100, "Each NFT can have no more than 100 copies"); // add this require statement in the already defined check statement in mint method	Data Validation	
049	High	require(msg.sender != address(0), "Address cannot be zero"); // remove this extra check! in mint method	Error Reporting	
050	Medium	require(amount <= 100 , "can not create copies more than 100"); // remove this extra check! in mint method	Patching	
051	Medium	require(nft[tokenId].nflExisted == false, "token id already exist"); // remove this require statement checking (nft[tokenId].nflExisted == false) in mint method	Patching	
052	High	nft[tokenId].nftMinter = true;// remove this extra value assigning code line in	Error Reporting	

		mint method		
053	High	nft[tokenId].nftExisted = true;// remove this extra value assigning code line in mint method	Denial of Service	
054	Medium	require(msg.sender != address(0), "Address cannot be zero");// remove this extra check! in mint batch method	Denial of Service	
055	High	require(nft[tokenId[ib]].nftExisted == false, "Token id already exist");// remove this extra check! in mint batch method	Denial of Service	
056	High	nft[tokenId[ib]].nftMinter = true;// remove this extra value assigning code line in mint batch method	Denial of Service	
057	High	nft[tokenId[ib]].nftExisted = true;// remove this extra value assigning code line in mint batch method	Denial of Service	
058	High	// this is a logical error the require check isnt checking the owner of the token id it would allow any one to	Error Reporting	

		update the nft uri in EditNftUri method		
059	High	// the pause() function isn't used anywhere in the whole contract.	Denial of Service	
060	Medium	// the unpause() function isn't used anywhere in the whole contract.	Denial of Service	
061	High	// Remove this extra function from the contract _beforeTokenTransfer().	Denial of Service	
062	High	// set this address of fundReceiver in the initialize;		
063	Medium	uint256 royaltyRecord; // change this variable name into royalty Percentage;		
064	High	private ownerOf; // this mapping is not set against any of the mint function		
065	Medium	require(balanceOf(msg.sender, _tokenId) == 0, "Token already exists");// this		

		require statement doest work according to the requirement any other user can mint the same id if he doesn't have any balance against that id in all the functions.		
066	Medium	require(_tokenUris.length > 0, "tokenUris cannot be empty"); // Remove this extra require statement		
067	Medium	require(_amounts.length > 0, "amounts cannot be empty"); // Remove this extra require statement		

Beside the identified issues, we emphasize that for any user-facing applications and services, it is always important to develop necessary risk-control mechanisms and make contingency plans, which may need to be exercised before the mainnet deployment. The risk-control mechanisms should kick in at the very moment when the contracts are being deployed on mainnet. Please refer to Section 3 for details.

3 | Detailed Results

3.1:_ Removal of dead code.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

string private _baseURI;

There is no need to use this state variable because it isn't used anywhere to set the base uri instead every minter uses its own uri against its own minting.

3.2:_ Removal of dead code.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

uint256 public LatestTokenId;

//audit: Remove this extra variable that isn't used for anything in the contract.

3.3:Removal of dead code.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

```
_baseURI = "";
```

Remove this extra variable that isn't used for anything in the contract.

3.4:_ Removal of dead code.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

```
mapping(uint256 => string)  private _tokenURIs;
```

//audit: This is an extra mapping which isn't used in the whole contract.

3.5:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

```
mapping(address => mapping(uint => uint)) public mnftCopies;
```

Change this double mapping into a single against id and store the data against it in a struct this will save a lot more gas cost.

3.6:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

```
mapping(address => mapping(uint => bool)) private mSetMinter;
```

Remove this mapping and store this data against an id in a struct, this will save a lot more gas cost.

3.7:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

mapping(address => mapping(uint => uint)) public mRoyaltyRecord;

Remove this mapping and store this data against an id in a struct, this will save a lot more gas cost.

3.8:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

mapping(address => mapping(uint => address)) public mSetRoyaltyReceiver;

Remove this mapping and store this data against an id in a struct, this will save a lot more gas cost.

3.9:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

```
mapping(address => mapping(uint => uint)) public mRoyaltyRecord;
```

Remove this mapping and store this data against an id in a struct, this will save a lot more gas cost.

3.10:_ Add onlyOwner modifier.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function “ChangeNFTAmountLimit(uint256 _amount)” Add an only owner modifier instead of setting an extra required statement to check manually that the msg.sender == to a specific address.

3.11:_ Add a require check that _amount != 0

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “ChangeNFTAmountLimit(uint256 _amount)” Add a required statement that the _amount != 0.

3.12:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

Remove this function “ _fIsTokenidAvailable(uint256 _tokenid)” because this function is giving the value of a private mTokenidAvailable mapping, if you made this mapping public you can use its default getter function so there will be no need for this function.

3.13:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

Remove this function “ _fIsMinter(address _address , uint256 _tokenId)” because this function is giving the value of a private mSetMinter mapping, if you made this mapping public you can use its default getter function so there will be no need for this function.

3.14:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

Remove this function “ _fSetLatestTokenId(uint256 _tokenid)” because this function is used to set the LatestTokenId state variable which is an extra variable and hasn't any use in the contract.

3.15:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

Remove this function " checkuserBalance(address _address)" because this function is a dead code which doesn't have any use in the whole contract.

3.16:_ Add a require statement.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function "mint(address account , uint256 tokenId , uint256 amount , string memory tokenuri , uint256 _royaltyFee)"

Add a require statement that the amount != 0.

3.17:_ Add a require statement.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function “mint(address account , uint256 tokenId , uint256 amount , string memory tokenuri , uint256 _royaltyFee)”

Add a require statement that the account address != 0.

3.18:_ Require check calculation isn't right.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function “mint(address account , uint256 tokenId , uint256 amount , string memory tokenuri , uint256 _royaltyFee)”

The require statement checking the _royaltyFee should be between 1.5 % to 20 % Isn't right. There is not any calculation that is 1.5 % or 20 % of which amount that is going to be checked there.

3.19:_ Removal of an “if else” statement.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function “mint(address account , uint256 tokenId , uint256 amount , string memory tokenuri , uint256 _royaltyFee)”

Add a require check for “ if(_fIsTokenidAvailable(tokenId)==false)” this instead of an if else statement .There is no need for this here because we are using the revert function in else which could take all the function gas.

3.20:_ Removal of an extra function call.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “mint(address account , uint256 tokenId , uint256 amount , string memory tokenuri , uint256 _royaltyFee)”

Remove this extra function “_fSetLatestTokenId(tokenId);” call which is updating an extra state variable.

3.21:_ Add a require statement.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “Lazymint(address minter , uint256 tokenId , uint256 amount , address creator ,string memory tokenuri , uint256 _creatorRoyaltyfee)”

Add a required statement that the minter address wouldn't be zero.

3.22:_ Logical error.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “Lazymint(address minter , uint256 tokenId , uint256 amount , address creator ,string memory tokenuri , uint256 _creatorRoyaltyfee)”

There must be a require statement to check the mag.value against some specific value that has already been set instead of any price the user adds and mint an nft.

3.23:_ Add a require check.

- ID: PVE-023
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “Lazymint(address minter , uint256 tokenId , uint256 amount , address creator ,string memory tokenuri , uint256 _creatorRoyaltyfee)”

Add a require check that the amount wouldn't be equal to zero.

3.24:_ Require check calculation isn't right..

- ID: PVE-024
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “Lazymint(address minter , uint256 tokenId , uint256 amount , address creator ,string memory tokenuri , uint256 _creatorRoyaltyfee)”

The require statement checking the _royaltyFee should be between 1.5 % to 20 % Isn't right. There is not any calculation that is 1.5 % or 20 % of which amount that is going to be checked there.

3.25:_ Removal of an “if else” statement.

- ID: PVE-025
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “Lazymint(address minter , uint256 tokenId , uint256 amount , address creator ,string memory tokenuri , uint256 _creatorRoyaltyfee)”

Add a require check for “if(mTokenidAvailable[tokenId]==false)” this instead of an if else statement .There is no need for this here because we are using the revert function in else which could take all the function gas.

3.26:_ Add a require check statement.

- ID: PVE-026
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)”

Add a require statement that “to” address wouldn't be a zero address.

3.27:_ Add a require check statement.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)”

Add a require statement that “tokenId” length must be equal to the “”amounts” length.

3.28:_ Add a require check statement.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function “mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)”

Add a require statement that “tokenId” length must be equal to the “”_royaltyFee length” length.

3.29:_ Require check calculation isn't right.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function “mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)”

The require statement checking the _royaltyFee should be between 1.5 % to 20 % Isn't right. There is not any calculation that is 1.5 % or 20 % of which amount that is going to be checked there.

3.30:_ Add a require check statement.

- ID: PVE-001
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)”

Add a require check that each nft has no more copies than 100.

3.31:_ Removal of an “if else” statement.

- ID: PVE-031
- Severity: Critical
- Likelihood: High
- Target: NFT Minter 1155
- Category: Business Logic [4]
- CWE subcategory: CWE

Description:

In the function “mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)”

Add a require check for “if(_fIsTokenidAvailable(tokenId[ib])==false)” this instead of an if else statement .There is no need for this here because we are using the revert function in else which could take all the function gas.

3.32:_ Code optimization to reduce extra gas cost.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function "mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)"

In this statement "if(_fIsTokenidAvailable(tokenId[ib])==false)" use a real mapping getter function instead of _fIsTokenidAvailable.

3.33:_ Removal of an extra for loop.

- ID: PVE-033
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function "mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)"

Remove this extra for loop in this "if(_fIsTokenidAvailable(tokenId[ib])==false)" statement.

3.34:_ Removal of an extra function call.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function "mintBatch(address to, uint256[] memory tokenId, uint256[] memory amounts, string[] memory tokenUris, uint256[] memory _royaltyFee)"

Remove this extra function call "_fSetLatestTokenId(tokenId[ib])" because it is updating an extra state variable.

3.35:_ Removal of an extra "if else" statement.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

In the function "EditNFTUri(uint256 tokenId, string memory newUri)"

Remove the if else statement and add a require check statement for "if(_fIsMinter(msg.sender,tokenId) == true)" this statement.

3.36:_ Removal of dead code.

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

Remove this extra function “safeTransferFrom(address from,address to,uint256 id,uint256 amount,bytes memory data) public virtual override ()”. Already defined in erc1155 contract no need to define it again.

3.37:_ Removal of dead code..

- ID: PVE-001
- Target: NFT Minter 1155
- Severity: Critical
- Category: Business Logic [4]
- Likelihood: High
- CWE subcategory: CWE

Description:

Removal of this extra “function _beforeTokenTransfer(address operator, address from, address to, uint256[] memory ids, uint256[] memory amounts, bytes memory data) internal whenNotPaused override” Already defined in erc1155 contract no need to define it again.