PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For FootieSzn WorldCup

15 November 2022

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for FootieSzn WorldCup on the Avalanche network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	FootieSzn WorldCup
URL	TBC
Network	Avalanche
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
TeamNFT	0xda5C32d76Fd53510EC0588416aA4437503BAFD26	FAIL
WorldCupFIFAContract	0xE53D7B50228D5713152dcABe3cE78Aeb37eCD94B	FAIL

Live match notes

TeamNFT
The team has made various changes since the audit which were not covered within the audit. The minter privileges have been removed again and now each individual country can have an individual maximum mint and mint cost. The team has also added royalty support after the audit.

WorldCupFIFAContract / UserContract
The team has made various changes since the audit which were not covered within the scope of the audit. Most notably, `setPriority` can be called by anyone again (minter logic was removed) — we checked in with the client and they indicated this is desired.

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	3	3	-	-
Medium	2	2	-	-
Low	5	5	-	-
Informational	2	2	-	-
Total	**12**	**12**	**-**	**-**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 TeamNFT

ID	Severity	Summary	Status
01	HIGH	Contract severely overmints NFTs due to a critical flaw in the mint codes	✓ RESOLVED
02	HIGH	Minting is vulnerable to reentrancy exploits, allowing for a single `tokenId` to be minted several times	✓ RESOLVED
03	MEDIUM	Transfers fail once the minting is disabled	✓ RESOLVED
04	MEDIUM	The per-country limit is not properly checked, allowing more than 1000 NFTs to be minted per country	✓ RESOLVED
05	LOW	Users can figure out the `MINT_AUTH` key by reading the contract or checking the transactions of other users	✓ RESOLVED
06	INFO	Typographical errors and gas optimizations	✓ RESOLVED

1.3.2 WorldCupFIFAContract

ID	Severity	Summary	Status
07	HIGH	Users can withdraw their winnings multiple times, draining any ETH in the contract	✓ RESOLVED
08	LOW	Lack of validation on `tokenIds`	✓ RESOLVED
09	LOW	`setWinner` ETH requirement inaccurate	✓ RESOLVED
10	LOW	`setWinner` overrides the user's unclaimed amount if they win again and have not claimed yet	✓ RESOLVED
11	LOW	`MAX_PER_MINT` does not actually check for a maximum	✓ RESOLVED
12	INFO	Typographical errors and gas optimizations	✓ RESOLVED

2 Findings

2.1 TeamNFT

`TeamNFT` is an ERC-1155 NFT token which can be purchased for 1 ETH. With each purchase, the buyer can indicate the team they are supporting and the purchase count of that team will increase. The on-chain code does not however explicitly link the team with the NFT. Up to 1000 NFTs are buyable per team. The on-chain code also does not validate whether the given team really exists, this will need to be done off-chain.

The `tokenIds` start from 1.

2.1.1 Privileged Functions

- `setMintingAllowed`
- `setMinter`
- `removeMinter`
- `transferOwnership`
- `renounceOwnership`

2.1.2 Issues & Recommendations

Issue #01	Contract severely overmints NFTs due to a critical flaw in the mint codes
Severity	HIGH SEVERITY

Location	Lines 57 and 88
	`_mint(_msgSender(), newTokenID, newTokenID, "");`
	`. . .`
	`_mintBatch(_msgSender(), ids, ids, "");`

Description	Instead of minting 1 NFT to the user, the code mints a number of NFTs equal to the current `nftID`. This means if a user is the 10th person to purchase an NFT, they receive 10 NFTs instead of just 1. This is obviously a critical flaw and also indicates a lack of testing.

Recommendation	Consider adjusting the code as follows:
	
	`_mint(_msgSender(), newTokenID, 1, "");`
	
	`uint256[] memory amounts = new uint256[](_count);`
	`for (uint256 i = 0; i < _count; i++) {`
	`  amounts[i] = 1;`
	`}`
	`. . .`
	`_mintBatch(_msgSender(), ids, amounts, “”);`
	
	Note that this last code should be structured so that the existing for-loop is re-used.

Resolution	RESOLVED

Issue #02	**Minting is vulnerable to reentrancy exploits, allowing for a single `tokenId` to be minted several times**
Severity	HIGH SEVERITY
Description	The `mint` functions do not adhere to checks-effects-interactions, nor do they have reentrancy guards. This allows an exploiter to reenter on the `_mint` call (which executes code on the recipient) and call the `mint` functions again to mint another token before the counter is incremented.
Recommendation	Consider adding a reentrancy guard to all mint functions (ideally also to the various transfer functions of ERC1155). Consider rewriting the mint functions to adhere to checks-effects-interactions.
Resolution	RESOLVED

Issue #03	**Transfers fail once the minting is disabled**
Severity	MEDIUM SEVERITY
Description	The contract allows for the owner to enable and disable minting. However, due to the way this is implemented, users will no longer be able to transfer their tokens either once minting is disabled.
Recommendation	Consider simply adding a guard to the top of the `mint` functions and removing the `_beforeTokenTransfer` check. `require(mintingAllowed, "Minting is disabled");`
Resolution	RESOLVED

Issue #04	**The per-country limit is not properly checked, allowing more than 1000 NFTs to be minted per country**
Severity	MEDIUM SEVERITY
Location	Line 85 `require(countryCount <= MAX_MINT, "Country Max Mint Reached");`
Description	The code contains a check so that each country can have up to 1000 NFTs. This check is flawed in two ways: - It should have been < - It should also be present in the mint function. Within this iteration of the code, this check can therefore be completely circumvented.
Recommendation	Consider rewriting the requirement: `require(countryCount < MAX_MINT, "Country Max Mint Reached");` Consider also copying this in the `mint` function so it applies there as well.
Resolution	RESOLVED This limit has been rewritten and now applies to `mint` as well.

Issue #05	**Users can figure out the `MINT_AUTH` key by reading the contract or checking the transactions of other users**
Severity	LOW SEVERITY
Description	The contract uses an authorization key which is required to be passed to mints. However, users can simply inspect the contract or the transactions of other users to figure out this key so it does not add much security.
Recommendation	Consider using an ECDSA signature scheme where the users' transactions need to be signed by the governance, if real governance approval is desired.
Resolution	RESOLVED The client has switched out this logic with a whitelist.

Issue #06	Typographical errors and gas optimizations
Severity	INFORMATIONAL

Description	We have consolidated the typographical errors and the sections which can be further optimized for gas usage below.
	
	Line 7
	`import "@openzeppelin/contracts/utils/math/SafeMath.sol";`
	
	This is not necessary on Solidity 0.8.
	
	Line 16-17
	`address private escrowAddress;`
	`address private developerAddress;`
	
	These variables should be made `immutable` and `public`.
	
	Line 18
	`uint256 MAX_MINT = 1000;`
	
	This variable should be explicitly marked as `private` and made `constant`.
	
	Line 24
	`event SentAmt(uint256 indexed amount, uint256 indexed devFee);`
	
	Indexing amounts has no value as they are hardly searchable. The actual emitted parameters are also different than the ones which are here claimed to be emitted.
	
	Line 27
	`require(msg.value == MINT_FEE, "Minting fee too low");`
	
	If the fee is too high, it will still be too low. Maybe "Wrong mint fee" is more adequate?

Line 49

```
function mint(string memory country, string memory authkey)
```

The parameters can be provided as `calldata` to save gas. The same goes for the parameters of `mintMultiple`.

Line 55

```
require(msg.value == MINT_FEE, "Minting fee too low");
```

This check already occurred with the modifier. It can be removed.

Line 74

```
string[] memory countries,
```

It should be validated that the length of this equals `_count`. Or `_count` should be removed altogether.

Line 102

```
require(mintingAllowed == true, "Minting is disabled");
```

This can be simplified to `require(mintingAllowed, "Minting is disabled");`.

Line 108

```
uint256 devFee = sendAmount.div(10).mul(1);
```

This should be rewritten to:

```
uint256 devFee = sendAmount.mul(PERCENT_FEE).div(100);
```

In our opinion `compareStrings` can also be made `internal` as this does not provide any value to users who use this function. The commented out code underneath it should also be removed.

`setMintingAllowed`, `mint` and `mintMultiple` can be marked as external.

Recommendation Consider fixing the typographical errors and implementing the recommended gas optimizations.

Resolution RESOLVED

Most of these have been resolved.

2.2 WorldCupFIFAContract

`WorldCupFIFAContract` allows users to upload a sequence of priorities. At some point, the admin can assign a winner and grant ethereum to that winner. The winner must then withdraw these tokens as a reward for choosing the right priority.

2.2.1 Privileged Functions

- `setAllowEditing [ admin ]`
- `setAllowWithdrawal [ admin ]`
- `SetWinner [ admin ]`
- `setMinter [admin ]`
- `removeMinter [ admin ]`
- `transferOwnership`
- `renounceOwnership`

2.2.2 Issues & Recommendations

Issue #07	**Users can withdraw their winnings multiple times, draining any ETH in the contract**
Severity	HIGH SEVERITY

Location	Line 105-112
	`function WithdrawalWinings() public {`
	`    require(allowWithdrawal == true, "Withdrawal is disabled!");`
	`    uint256 balance = winners[msg.sender];`
	`    require(balance > 0, "No ether left to withdraw");`
	
	`    (bool success, ) = (msg.sender).call{value: balance}("");`
	`    require(success, "Transfer failed.");`
	`}`

Description	Users can withdraw their winnings multiple times as they are not reset. This allows any winner to fully drain the contract of ETH.

Recommendation	Consider rewriting the function:
	
	`function WithdrawalWinings() public {`
	`    require(allowWithdrawal == true, "Withdrawal is disabled!");`
	`    uint256 balance = winners[msg.sender];`
	`    require(balance > 0, "No ether left to withdraw");`
	`        winners[msg.sender] = 0;`
	
	`    (bool success, ) = (msg.sender).call{value: balance}("");`
	`    require(success, "Transfer failed.");`
	`}`
	
	Consider fixing the typographical error in the function name: `withdrawWinnings`.

Resolution	RESOLVED The reset line has been added.

Issue #08	Lack of validation on `tokenIds`
Severity	LOW SEVERITY
Description	The contract allows users to submit priorities of `tokenIds`, however, these priorities or ids are not validated in any way. This means that presently anyone can upload anything, without paying anything, as long as it is a sequence of 4 numbers.
Recommendation	Consider whether further on-chain validation is necessary. If it is desired that users can freely submit whatever they want, than this issue will be resolved.
Resolution	RESOLVED The client has implemented the feature that only whitelisted wallets can pick a configuration. There is still no checks on the specific configurations which can be picked (or on-chain specification to the requirements to become whitelisted), though the client has indicated this is how they want things to be.

Issue #09	`setWinner` ETH requirement inaccurate
Severity	LOW SEVERITY
Location	Line 98 `getBalance() > amountWon,`
Description	This check requires more ETH than is necessary, as the balance must be greater. This check also fails if multiple winners are assigned as it does not account for any unclaimed amounts.
Recommendation	Consider using >= instead, consider using an `unclaimed` counter to adjust the check for unclaimed amounts.
Resolution	RESOLVED

Issue #10	**`setWinner` overrides the user's unclaimed amount if they win again and have not claimed yet**
Severity	LOW SEVERITY
Location	Line 101 `winners[winnerAddress] = amountWon;`
Description	This line overrides any existing winnings of the user.
Recommendation	Consider incrementing the winnings instead: `winners[winnerAddress] += amountWon;`
Resolution	RESOLVED

Issue #11	**`MAX_PER_MINT` does not actually check for a maximum**
Severity	LOW SEVERITY
Location	Line 63 `tokenIds.length == MAX_PER_MINT`
Description	This variable indicates a maximum, however the check indicates that it must be exact.
Recommendation	Consider whether this check is supposed to check for an inequality.
Resolution	RESOLVED The client clarified that this variable is supposed to be the exact amount of tokens.

Issue #12	Typographical errors and gas optimizations
Severity	INFORMATIONAL

Description	We have consolidated the typographical errors and the sections which can be further optimized for gas usage below.
	
	Line 6
	`contract WorldCupFIFAContract is Ownable {`
	
	The owner appears to be unused, the Ownable dependency and import can therefore be removed.
	
	Line 14
	`mapping(address => uint256) winners;`
	
	This should be marked as `public`.
	
	Line 15
	`address public _admin;`
	
	This should be marked as `immutable`.
	
	Line 30
	`event AllowEdting(bool allowed);`
	
	This should be written as `Editing`.
	
	Line 40
	`fallback() external payable {}`
	
	This should be removed as wrong calls to the contract will not revert with this code. `receive()` suffices to be able to receive Eth.
	
	Line 59
	`function setPriority(uint256[] memory tokenIds) public {`
	
	`tokenIds` should be made `calldata`.

Line 61

```
require(allowEditing == true, "Priority Update is
disabled!");
```

`require(allowEditing, “Prioritiy Update is disabled!”)` suffices.

Line 72

```
emit PriorityModified(msg.sender, tokenIds[0],
block.timestamp);
```

Emitting a timestamp wastes gas as this timestamp is already present in the data of the nodes.

Line 76

```
function getUserPriority() public view returns (uint256[]
memory) {
```

We typically avoid `view` functions which use `msg.sender` instead of a user parameter since there is no benefit to not simply using the other function with the user parameter.

Line 92

```
function SetWinner(address winnerAddress, uint256 amountWon)
```

This should be `setWinner`.

Line 99

```
"Insufficeint Balance to add winner for"
```

This should say “Insufficient”.

	The following functions can be made external: - `setAllowEditing` - `setAllowWithdrawal` - `setPriority` - `getUserPriority` - `getAddressPriority` - `SetWinner` - `WithdrawalWinings`
Recommendation	Consider fixing the typographical errors and implementing the recommended gas optimizations.
Resolution	RESOLVED Most of these have been resolved.


PALADIN
BLOCKCHAIN SECURITY