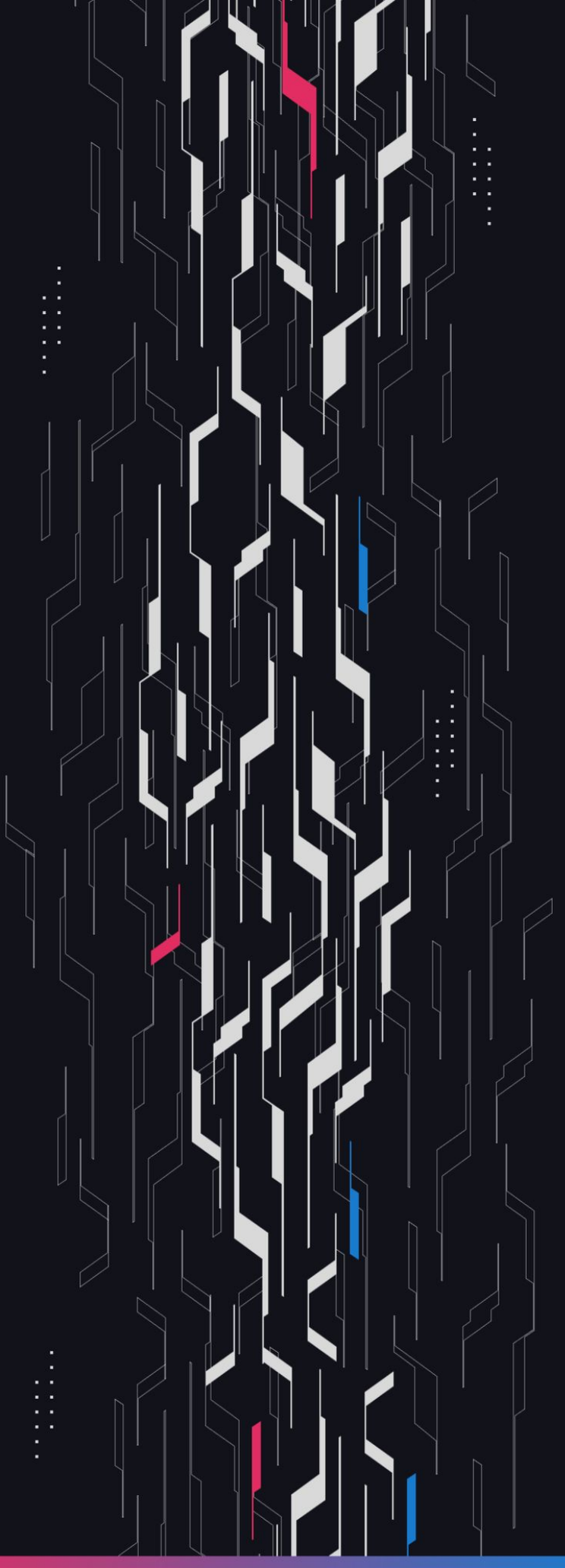
GA GUARDIAN

PleasrDAO

The Album

Security Assessment

June 11th, 2024



Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Owen Thurm, Wafflemakr, Giraffe, Osman Ozdemir, Mike Lett

Client Firm PleasrDAO

Final Report Date June 11, 2024

Audit Summary

PleasrDAO engaged Guardian to review the security of its DN404 tokenization of the coveted “Once Upon A Time In Shaolin” Wu Tang Clan album. From the 20th of May to the 27th of May, a team of 6 auditors reviewed the source code in scope. All findings have been recorded in the following report.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Ethereum**

✅ Verify the authenticity of this report on Guardian’s GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/album-fuzzing/tree/main

Table of Contents

Project Overview

Project Summary

Project Name	PleasrDAO
Language	Solidity
Codebase	https://github.com/strollinghome/album
Commit(s)	72b71cc67959ed0bf5548af977a94ddefb10ceb5

Audit Summary

Delivery Date	June 11, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	1	0	0	0	0	1
Medium	7	0	0	1	1	5
Low	17	0	0	9	0	8

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

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. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of the Pleasr album contracts, fuzz-testing with Foundry was performed on the protocol's main functions. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 1,000,000+ runs up to a depth of 500 with a prepared Foundry fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
PD-01	Sum of Owned NFTs == Mirror Total Supply	✅	✅	✅	1,000,000+
PD-02	Sum of Owned ERC20 == Token Total Supply	✅	✅	✅	1,000,000+
PD-03	No User Owns type(uint32).max NFT	✅	✅	✅	1,000,000+
PD-04	Allowance Matches Approved Amount	✅	✅	✅	1,000,000+
PD-05	Owner Auxiliary Data Is Not Modified Upon Approval	✅	✅	✅	1,000,000+
PD-06	Spender Auxiliary Data Is Not Modified Upon Approval	✅	✅	✅	1,000,000+
PD-07	ERC20 Balance Changes By Amount For Sender And Receiver Upon Transfer	✅	✅	✅	1,000,000+
PD-08	ERC20 Balance Remains The Same Upon Self-Transfer	✅	✅	✅	1,000,000+
PD-09	ERC20 Total Supply Remains The Same Upon Transfer	✅	✅	✅	1,000,000+
PD-10	Auxiliary Data Is Not Modified Upon Transfer	✅	✅	✅	1,000,000+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
PD-11	ERC20 Balance Changes By Amount For Sender And Receiver Upon TransferFrom	✅	✅	✅	1,000,000+
PD-12	ERC20 Balance Is the Same Upon Self-Transfer Upon TransferFrom	✅	✅	✅	1,000,000+
PD-13	ERC20 Total Supply Remains The Same Upon TransferFrom	✅	✅	✅	1,000,000+
PD-14	Auxiliary Data Is Not Modified Upon TransferFrom	✅	✅	✅	1,000,000+
PD-15	From Address != Address 0 Upon TransferFrom When Not Transferrable	✅	❌	✅	1,000,000+
PD-16	From/To Address Should Match Transfer Event	✅	✅	✅	1,000,000+
PD-17	User Balance Increased By Mint Amount	✅	✅	✅	1,000,000+
PD-18	Total ERC20 Supply Increased By Mint Amount	✅	✅	✅	1,000,000+
PD-19	Total NFT Supply Post-Mint Is At Least Total NFT Supply Pre-Mint	✅	✅	✅	1,000,000+
PD-20	Auxiliary Data Increased By Mint Amount	✅	✅	✅	1,000,000+
PD-21	Approved NFT Spender == Requested Approval	✅	✅	✅	1,000,000+
PD-22	Owner Of NFT ID Is Not Modified Upon Approval Of NFT	✅	✅	✅	1,000,000+
PD-23	Owner Auxiliary Data Is Not Modified Upon Approval	✅	✅	✅	1,000,000+
PD-24	Spender Auxiliary Data Is Not Modified Upon Approval	✅	✅	✅	1,000,000+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
PD-25	NFT Balance Of Sender and Receiver Accurately Updated Upon TransferNFT	✅	✅	✅	1,000,000+
PD-26	Sender/Receiver ERC20 Balance Decremented/Incremented By Unit	✅	✅	✅	1,000,000+
PD-27	NFT Balance Is the Same Upon Self-Transfer Upon TransferNFT	✅	✅	✅	1,000,000+
PD-28	Receiver Address Is The Owner At The Sent NFT ID	✅	✅	✅	1,000,000+
PD-29	Total NFT Supply Is Unchanged Upon NFT Transfer	✅	✅	✅	1,000,000+
PD-30	Approval Is Reset Upon NFT Transfer	✅	✅	✅	1,000,000+
PD-31	Sender Auxiliary Data Is Not Modified Upon NFT Transfer	✅	✅	✅	1,000,000+
PD-32	Receiver Auxiliary Data Is Not Modified Upon NFT Transfer	✅	✅	✅	1,000,000+
PD-33	Skip NFT Status Is Updated To Requested Status	✅	✅	✅	1,000,000+
PD-34	Auxiliary Data Is Not Modified Upon Set Skip NFT	✅	✅	✅	1,000,000+
PD-35	Set Approval For All Updated To Requested Status	✅	✅	✅	1,000,000+
PD-36	Owner Auxiliary Data Is Not Modified Upon Set Approval For All	✅	✅	✅	1,000,000+
PD-37	Spender Auxiliary Data Is Not Modified Upon Set Approval For All	✅	✅	✅	1,000,000+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
PD-38	_ownerAt(id) Is Always The Same As NFT Holder	✅	✅	✅	1,000,000+
PD-39	NFT ID Must Be Less Than Or Equal To Total Supply Of NFTs	✅	✅	✅	1,000,000+
PD-40	Intervals should not be reduced when mint is not live	✅	❌	✅	1,000,000+
PD-41	User Mint Count Equals Intervals Reduced Minus Admin Reduced Intervals	✅	✅	✅	1,000,000+
PD-42	Current Intervals Should Be Greater Than or Equal To Intervals Reduced	✅	✅	✅	1,000,000+
PD-43	Intervals Reduced Should Increase By Mint Amount	✅	✅	✅	1,000,000+
PD-44	URI returns Expected Information Upon contractURI	✅	✅	✅	1,000,000+
PD-45	Token URI Fields Are Updated Accurately Upon setMedia	✅	✅	✅	1,000,000+

Findings & Resolutions

ID	Title	Category	Severity	Status
H-01	Missing onlyLive Modifier	Access Control	High	Resolved
M-01	NFT Marketplaces Will Not Read Royalty Info	Logical Error	Medium	Resolved
M-02	contractURI Metadata Is Not Queryable From ERC721 Mirror	Logical Error	Medium	Resolved
M-03	Malicious Bid Gas Griefing	Griefing	Medium	Resolved
M-04	Purchase Price May Significantly Differ Based On The Currency	Logical Error	Medium	Acknowledged
M-05	Contract URI Does Not Conform To ERC-7572	Logical Error	Medium	Partially Resolved
M-06	Circumvented Token Transfer Restrictions	Validation	Medium	Resolved
M-07	Zero Amount Purchases Allowed Before And After Minting Period	Logical Error	Medium	Resolved
L-01	Excess ETH Not Refunded	Logical Error	Low	Acknowledged
L-02	USDC Token Purchase DoS'ed By Blacklisted fundsRecipient	DoS	Low	Acknowledged
L-03	Invalid secondsReduced Emitted	Logical Error	Low	Acknowledged
L-04	Countdown May Not Reach Zero	Logical Error	Low	Acknowledged
L-05	reduceIntervals Lacking Input Validation	Validation	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-06	tokenURI Fields Not Fully Configurable	Logical Error	Low	Acknowledged
L-07	Lack Of Indexed Parameter In Event	Events	Low	Acknowledged
L-08	Liquidity Pool Considerations	Unexpected Behavior	Low	Acknowledged
L-09	Typo	Typo	Low	Resolved
L-10	Unused Custom Error	Optimization	Low	Resolved
L-11	Memory Parameters Can Be Calldata	Optimization	Low	Resolved
L-12	Variables Can Be Declared Immutable	Mutability	Low	Resolved
L-13	Mirror Contract Owner Should Be Synced In The Constructor	Logical Error	Low	Resolved
L-14	Inconsistent Media Field Names	Best Practices	Low	Resolved
L-15	Unimplemented Feature	Optimization	Low	Acknowledged
L-16	NFTMetadataRenderer Library Collection Size Is 0	Optimization	Low	Acknowledged
L-17	Users Can Game The Leaderboard System	Gaming	Low	Resolved

H-01 | Missing onlyLive Modifier

Category	Severity	Location	Status
Access Control	High	Token.sol: 173	Resolved

Description

The `purchaseWithUSDC` function lacks an `onlyLive` modifier, therefore mints can still occur with USDC as a payment token when the contract is disabled.

Recommendation

Add an `onlyLive` modifier to the `purchaseWithUSDC` function.

Resolution

PleasrDAO Team: The issue was resolved in commit bb5c069.

M-01 | NFT Marketplaces Will Not Read Royalty Info

Category	Severity	Location	Status
Logical Error	Medium	Global	Resolved

Description

The Token contract implements the ERC2981 standard to signal royalty fees to be taken when NFT sales are made on NFT marketplaces. However the Token contract is the ERC20 compatible contract, not the ERC721 compatible contract, therefore NFT exchanges which will interact with the `DN404Mirror` contract will not read the `royaltyRecipient` and `royaltyFee` that are configured in the `Token` contract.

As the team currently does not planning on utilizing the royalty feature, the severity of the omission is limited.

Recommendation

Create a contract which inherits `DN404Mirror` and implements the ERC2981 standard with the royalty configurations. Consider adding a function to update the bps for future-proofing.

Resolution

PleasrDAO Team: The issue was resolved in commit 63e2585.

M-02 | contractURI Metadata Is Not Queryable From ERC721 Mirror

Category	Severity	Location	Status
Logical Error	Medium	Token.sol: 253	Resolved

Description

The `Token` contract implements a `contractURI` function which is intended to include collection level metadata about the ERC721 counterpart of the DN404 pair.

However, unlike the `tokenURI`, the `contractURI` cannot be queried from the `DN404Mirror` contract and therefore this metadata is not available for integrators who would query the ERC721 compatible contract for it.

Recommendation

Create a contract which inherits from the `DN404Mirror` contract and adds a `contractURI` function which uses `_readString` to query the DN404 base contract, similar to the `tokenURI` function. Then override the `dn404Fallback` function to add the corresponding selector functionality for a `_contractURI` function.

Resolution

PleasrDAO Team: The issue was resolved in commit 7d1d3b7.

M-03 | Malicious Bid Gas Griefing

Category	Severity	Location	Status
Griefing	Medium	Global	Resolved

Description

The DN404 contract has a public `setSkipNFT` function where the `msg.sender` can assign a true or false skip status to themselves.

A malicious actor may abuse this functionality to place bids on listed NFTs which would gas grief the lister upon acceptance with the following steps:

• Call `setSkipNFT` to set their skip status to true.
• Accumulate many ERC20 DN404 tokens by minting through the token contract, but no NFTs since they have a skip status of true.
• Call `setSkipNFT` to set their skip status to false.
• Place bids on listed DN404 NFTs, such that if they are accepted the lister would have to expend a significant amount of gas to mint the NFTs corresponding to the accounts pre-existing ERC20 balance.

This can result in an unexpected loss of funds for the lister through gas expenditure, or even allow for the creation of bids which cannot be accepted as their execution cost would exceed the block gas limit.

Recommendation

Be aware of this risk and document it for users. Consider overriding and disabling the `setSkipNFT` function to remove this potential griefing vector.

Resolution

PleasrDAO Team: The issue was resolved in commit cd3cd28.

M-04 | Purchase Price May Significantly Differ Based On The Currency

Category	Severity	Location	Status
Logical Error	Medium	Global	Acknowledged

Description

Users can buy Album NFTs with ETH or USDC based on their preference. These token prices are predetermined (0.004 ETH or 15 USDC) and immutable. However, since this is a long term project, there will definitely be significant price movements in terms of ETH/USDC. Users will always choose to buy with the lower price.

None of the users will buy with ETH when the ETH price increases in the long term and the protocol will still get $15 per token in that scenario. However, in the other scenario when ETH price goes down, users will buy with ETH at a much cheaper price. Even if ETH goes to $2500, NFT price per token will be $10 and it is 33% discount in expected sale price.

Recommendation

One option is giving the owner the right to arrange prices based on market movements. The other option is determining a minimum token price in terms of USD value, and charging users at least corresponding amount of ETH if it requires more than 0.004 ETH.

Consider choosing an option based on the protocol's intentions since the former increases the owner power and the latter requires an oracle implementation and increases complexity.

Resolution

PleasrDAO Team: Acknowledged.

M-05 | Contract URI Does Not Conform To ERC-7572

Category	Severity	Location	Status
Logical Error	Medium	Token.sol: 261	Partially Resolved

Description

In contractURI, the field for external_url should be named external_link instead to conform with ERC-7572 (https://eips.ethereum.org/EIPS/eip-7572). Additionally, ERC-7572 requires an event ContractURIUpdated, which is currently not implemented.

This is also the standard that Opensea uses, see metadata. Not adhering to ERC-7572 could result in improper display of information on secondary marketplaces like Opensea where NFTs are traded.
It should be noted however that in tokenURI, external_url is the correct naming.

Recommendation

Rename the media field from `external_url` to `external_link`.
Create a separate update function for `contractURI` and emit the event `ContractURIUpdated`.

Resolution

PleasrDAO Team: The issue was resolved in commit 1cf921e.

M-06 | Circumvented Token Transfer Restrictions

Category	Severity	Location	Status
Validation	Medium	Token.sol: 362	Resolved

Description

ERC20 tokens are non transferable when the contract is deployed. This is enforced with the `transferable` flag, preventing `_transfer` and `_transferFromNFT` to be executed when the `from` address is not the zero address, to allow token minting.

The DN404 contract does not validate if the `from` address is the zero address in the `transferFrom` function. Therefore, users may call the function as follows: `transferFrom(address(0), BOB, 0)`.

The call will not revert, the `transferable` condition is circumvented, and a `Transfer` event will be emitted. Although there is no impact on user's balance, this is an unexpected behavior which may trick off-chain services.

Recommendation

Consider overriding the DN404 `transferFrom` function to add the zero address check on the `from` address.

Resolution

PleasrDAO Team: The issue was resolved in commit 5b49193.

M-07 | Zero Amount Purchases Allowed Before And After Minting Period

Category	Severity	Location	Status
Logical Error	Medium	Token.sol: 158	Resolved

Description

Both `purchase()` and `purchaseWithUSDC()` do not validate if `nftAmount_` parameters is greater than zero. Even though there is no change in the contract state, there will be unexpected events emitted like `Transfer`, `IntervalsReduced` and `Minted`.

Furthermore, purchases with zero amounts are possible before `startTime` and after `endTime` due to modifier `checkAndUpdateReducedIntervals` calculating `currentIntervalsLeft` as zero instead of reverting. Again, although there is no impact of contract state, there could be unexpected behavior with off-chain monitoring systems due to this issue.

Recommendation

Require that `nftAmount_` parameter is non zero in both purchase functions. Also, in `checkAndUpdateReducedIntervals` revert instead of returning zero if before or after minting period.

```
uint256 currentIntervalsLeft = block.timestamp < startTime ||
            block.timestamp >= endTime
            ? revert BeforeAfterMintingPeriod()
            : _initialIntervals - ((block.timestamp - startTime) / interval);
```

Resolution

PleasrDAO Team: The issue was resolved in commit 883d02c.

L-01 | Excess ETH Not Refunded

Category	Severity	Location	Status
Logical Error	Low	Token.sol: 133	Acknowledged

Description

Users can purchase NFTs with ETH or USDC, where `purchase()` is a payable function in charge of receiving ETH for tokens. It validates if a user has sent enough funds with the `checkPrice` modifier. In case a user sends more ETH than the NFT value, these funds will not be refunded, and will be sent to the `fundsRecipient` instead.

Recommendation

Consider adding a refund function where the excess ETH is first sent to the user and the remaining sent to the funds recipient.

Resolution

PleasrDAO Team: Acknowledged.

L-02 | USDC Token Purchase DoS'ed By Blacklisted fundsRecipient

Category	Severity	Location	Status
DoS	Low	Token.sol: L179	Acknowledged

Description

User purchasing tokens with USDC will call `purchaseWithUSDC()`. This function will collect the USDC from the user and transfer it to the `fundsRecipient`.

The USDC token in Base has a blacklist functionality. In case `fundsRecipient` address gets blacklisted, `purchaseWithUSDC()` will revert for all users. Due to the fact that there is no way to update the `fundsRecipient`, users will only be able to buy tokens using ETH.

Recommendation

Consider adding an admin function to update the `fundsRecipient` address.

Resolution

PleasrDAO Team: Acknowledged.

L-03 | Invalid secondsReduced Emitted

Category	Severity	Location	Status
Logical Error	Low	Token.sol: 145	Acknowledged

Description

The `checkAndUpdateReducedIntervals` function uses round down division to reduce the currentIntervalsLeft by the intervals that have passed. Therefore the final interval in the mint period can be less than 15 minutes, however the `Mint` event assumes that an entire interval of time was removed from the countdown.

Consider the following scenario:

• Intervals are 10 seconds
• The period is 100 seconds in total
• Time is currently at 95 seconds
• `currentIntervalsLeft = 10 - 95 / 10 = 1`
• Bob mints the last interval, technically this only removes 5 seconds from the countdown, but the Mint event emits that it took off the entire 10 seconds

This will misinform consumers of the Mint event and potentially cause issues with integrating off-chain applications.

Recommendation

Consider using round up division to compute the `currentIntervalsLeft`, thus not allowing for partial interval mints.

Resolution

PleasrDAO Team: Acknowledged.

L-04 | Countdown May Not Reach Zero

Category	Severity	Location	Status
Logical Error	Low	Token.sol: 106	Acknowledged

Description

The `Token` constructor validates the `MintConfig` params, and initializes the `_initialIntervals` immutable param with the following formula: `_initialIntervals = (mintConfig_.endTime -mintConfig_.startTime) / mintConfig_.interval;`

This param will be used to calculate the amount of intervals a user can purchase. When `interval` is not a multiple of the difference between `endTime` and `startTime`, the division will round down, and `_initialIntervals` value will be 1 interval short. Therefore, when a user purchases all intervals available to buy, the countdown will not reach zero.

Consider the following scenario:

• startTime: 100, endTime: 200, interval: 17
• _initialIntervals: 5 (200 - 100 / 17)
• user buys all available intervals at t=100
• final countdown = 15 seconds

`endTime - intervalsReduced * interval - block.timestamp = 200 - 5*17 - 100 = 15`

Recommendation

Ensure the difference in time between end and start time is an even multiple of interval. Otherwise, consider using a round up division to make sure countdown reaches zero when all available intervals are purchased. Keep in mind this will cause countdown to be negative in some cases.

Resolution

PleasrDAO Team: Acknowledged.

L-05 | reduceIntervals Lacking Input Validation

Category	Severity	Location	Status
Validation	Low	Token.sol:: 199	Resolved

Description

`reduceIntervals()` is an important admin function used to reduce intervals without minting NFTs. The initial number of intervals is estimated at 2.1 mil (41 mil minutes / 20 min intervals). Given this large number, admin input error is possible when trying to reduce the number of intervals.

If an incorrect `intervalAmount_` input is used, it could greatly reduce the amount of time left for minting in an irreversible way.

Recommendation

Consider some form of input validation, for example by adding a `max interval amount` check.

Resolution

PleasrDAO Team: The issue was resolved in commit 7087a65.

L-06 | tokenURI Fields Not Fully Configurable

Category	Severity	Location	Status
Logical Error	Low	Token.sol: 219	Acknowledged

Description

The specification document states that update of `tokenURI` metadata is a mandatory feature.
While there is a `setMedia` function implemented, it does not allow for all `tokenConfig` fields to be updated.

`tokenConfig` has a total of 14 fields while `setMedia` can only update 5 of those fields. Fields such as `image` and `encrypted_media_url` are among those excluded and cannot be updated.
Allowing these fields to be updated is important for possible features to be added in the future.

Recommendation

Allow `setMedia` to configure all of `tokenConfig` fields.

Resolution

PleasrDAO Team: Acknowledged.

L-07 | Lack Of Indexed Parameter In Event

Category	Severity	Location	Status
Events	Low	Token.sol: 17	Acknowledged

Description

The `Minted` event lacks an indexed parameter for `to` address, so off-chain services will not be able to filter them by user address.

Recommendation

Add the indexed `to` parameter:

```
event Minted(address indexed to, uint256 amount, uint256 secondsReduced);
```

Resolution

PleasrDAO Team: Acknowledged.

L-08 | Liquidity Pool Considerations

Category	Severity	Location	Status
Unexpected Behavior	Low	Global	Acknowledged

Description

While ERC-721 royalties cannot be enforced, royalties could be earned indirectly through fees from staking in Liquidity Pools (LPs), because of DN404's dual nature as an ERC20.

Recommendation

Therefore, the protocol should consider:

1) Setting aside an amount of tokens to create LP
2) Adding admin ability to `setSkipNFT` for the liquidity pool contract to prevent it from minting NFTs each time a swap is done so as to improve gas efficiency.

Resolution

PleasrDAO Team: Acknowledged.

L-09 | Typo

Category	Severity	Location	Status
Typo	Low	Token.sol: 88	Resolved

Description

In the `Token` constructor the royalty recipient parameter is misspelled as `royaltRecipient_`.

Recommendation

Correct the `royaltRecipient_` to `royaltyRecipient_`.

Resolution

PleasrDAO Team: The issue was resolved in commit 63e2585.

L-10 | Unused Custom Error

Category	Severity	Location	Status
Optimization	Low	Token.sol: 33	Resolved

Description

The `InvalidMint` custom error is defined but never used.

Recommendation

Remove unused custom error.

Resolution

PleasrDAO Team: The issue was resolved in commit 5b49193.

L-11 | Memory Parameters Can Be Calldata

Category	Severity	Location	Status
Optimization	Low	Token.sol: 219	Resolved

Description

The `string memory` parameters for the `setMedia` function are never mutated and therefore can be declared as `calldata`.

Recommendation

Convert the `string memory` parameters to `string calldata` parameters.

Resolution

PleasrDAO Team: The issue was resolved in commit a8700e5.

L-12 | Variables Can Be Declared Immutable

Category	Severity	Location	Status
Mutability	Low	Token.sol: 74-76	Resolved

Description

In the `Token` contract the `startTime`, `endTime`, and `interval` storage variables are assigned to only once in the constructor and are never reassigned, therefore they can be declared as `immutable`.

Recommendation

Declare the `startTime`, `endTime`, and interval storage variables as `immutable`.

Resolution

PleasrDAO Team: The issue was resolved in commit e97717d.

L-13 | Mirror Contract Owner Should Be Synced In The Constructor

Category	Severity	Location	Status
Logical Error	Low	Token.sol	Resolved

Description

DN404 and Mirror contracts are synced with `_initializeDN404`, and the owner of the DN404 is updated with `_initializeOwner` in the constructor. However, this action do not update the owner of the Mirror contract.

After the initialization of these contracts, the owner of the Mirror contract is still `address(0)` even though the owner of the DN404 is the `owner`. Anyone can call the `pullOwner` function and sync them, but there will be a mismatch until this function is called.

Recommendation

Call the `pullOwner` function in the constructor as a last step after calling `_initializeDN404`.

Resolution

PleasrDAO Team: The issue was resolved in commit 43c8bc5.

L-14 | Inconsistent Media Field Names

Category	Severity	Location	Status
Best Practices	Low	Token.sol: 295	Resolved

Description

Throughout the `Token` contract snake case is used to represent the names of url media fields, however the animation url field is named `animationURL`, which does not follow the snake case standard for url fields.

Recommendation

Consider if the `animationURL` field should be renamed as `animation_url`. Additionally, ensure the other fields have the appropriate expected formatting.

Resolution

PleasrDAO Team: The issue was resolved in commit c7c9871.

L-15 | Unimplemented Feature

Category	Severity	Location	Status
Optimization	Low	Token.sol: 159-172	Acknowledged

Description

According to protocol specs, the `purchase` function “Should try to mint max to, or up to that much if there is less supply left”.

However, the function reverts when the requested amount is more than the remaining supply. This might cause a big purchase to revert when there are still tokens to be minted, and the user may lose their chance while trying again with a lower amount due to race condition if the demand is high.

Recommendation

Consider updating the function in a way to mint the remaining supply or document this behaviour.

Resolution

PleasrDAO Team: Acknowledged.

L-16 | NFTMetadataRenderer Library Collection Size Is 0

Category	Severity	Location	Status
Optimization	Low	Token.sol: 290	Acknowledged

Description

`_tokenURI` function uses `NFTMetadataRenderer` library to create the metadata for NFTs. According to this library, last parameter of the `tokenURIMetadata` function should be the size of entire edition. However, the function is called with the value `0` and the metadata is not rendered with the true collection size as expected.

Recommendation

This may be the expected behavior, however if it is not use the current NFT supply as the size while rendering metadata, with the knowledge that this size can fluctuate.

Resolution

PleasrDAO Team: Acknowledged.

L-17 | Users Can Game The Leaderboard System

Category	Severity	Location	Status
Gaming	Low	Global	Resolved

Description

Users can climb the leaderboard by minting Album tokens and will get incentives based on their leaderboard ranking. The leaderboard is determined solely based on the `mintCount`.

`mintCount` is only updated when a user purchases and mints directly but it is not updated after transfers or secondary sales. This is done to prevent gaming like buying a lot of tokens from secondary market just before the sale ends.

A user could potentially mint many tokens, sell them all, and use the proceeds to mint more tokens in order to move up the leaderboard while only expending a limited amount of initial capital.
This risk is however not applicable if token transfers are disabled during the entire period of the sale.

Recommendation

Be sure to keep token transfers disabled during the entire period of the sale.

Resolution

PleasrDAO Team: Resolved.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits