Three Sigma Labs

RAGNARÖK

Code Audit

Ronin Zero NFTs

Ragnarok Meta

Disclaimer

Table of Contents

Summary

Three Sigma Labs audited Ragnarok's Ronin Zero NFT smart contract in a 3 person week engagement. The audit was conducted from 03-04-2022 to 20-04-2022.

Ragnarok is a metaRPG NFT project that enables users to claim ownership of their in-game character by representing ownership with NFTs. In the game, players will be able to battle monsters, loot objects, craft NFTs, trade, earn, and own digital real estate.

Remarks

The audit uncovered significant flaws that would have resulted in unexpected behavior and compromise of the smart contract. Namely, the following issues were classified as critical in terms of severity:

- (3S-RAG-01) Maximum mint amount exceeded during batch mint.
- (3S-RAG-02) Inconsistent counter increment during team sale.
- (3S-RAG-17) Wrong increment of the total minted variable.

These issues were acknowledged and promptly fixed by the Ragnarok team.

Additionally, Three Sigma Labs presented a series of suggestions targeting gas optimizations. A comparison summary of the gas units used by the original and optimized contract is presented in the table below:

	Original (gas used)	Optimized (gas used)	Improvement
Mint 1	216,616	60,391	72.1 %
Mint 3	402,849	92,089	77.1 %

(Gas usage of mint functions measured using Foundry's gas reports)

Scope

The audit reviewed file Ragnarok.sol which encompasses all the sale process and inherits from OppenZeppelin's ERC1155Pausable.

The NFT minting logic implements the following specification:

	Name	Duration	NFT Amount	Price
Phase 1	Public Dutch Auction	1 Day	3900	0.77 ETH
Phase 2	Pixel Whitelisted Mint	1 Day	3000	Phase 1 final price
Phase 3	Pill Whitelisted Mint	1 Day	600	Half of Phase 1 final price
Phase 4	Team Mint	1 Day	277	None
Phase 5	Final Public Sale	Until every NFT is sold	Remaining up to 7777	Phase 1 final price

Dutch auction specification:

- Starting price is 0.77 ETH.
- Price decreases every 7 minutes by 0.01925 ETH.
- A maximum of 3 NFTs can be minted per address.
- The cost of the last sold NFT during the Dutch auction will serve as the base price for subsequent phases.
- All Dutch auction participants will be refunded the difference between their purchase price and the final price.

UPDATE 18-04-2022:
On the 18th of April 2022 the Ragnarok team decided to change the refund mechanism such that the refund was to be initiated by the team instead of being claimable by each user directly from the contract (with the purpose of reducing gas costs). These changes were introduced in commit **62868626d0c025428f639b1c63fc35186de1d3ac**.

Assumptions

Throughout this code audit, it was assumed that the Ragnarok team honors the whitelisted addresses and does not tamper in any way with the contract once it has been deployed.

UPDATE 18-04-2022:
The Ragnarok team is also expected to repay bidders of the Dutch auction if there is a price difference between the amount they paid and the auction's closing price.

Methodology

To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. After that we thoroughly examined the code for known security flaws and attack vectors. Following that logic, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form. Finally, we considered several different gas optimizations and guided the Ragnarok team on implementing them.

Taxonomy

In this audit we report our findings using as a guideline Immunefi's vulnerability taxonomy, which can be found at immunefi.com/severity-updated/.

Level	Description
Critical	- Empty or freeze the contract's holdings. - Cryptographic flaws.
High	- Token holders temporarily unable to transfer holdings. - Users spoof each other. - Theft of yield. - Transient consensus failures.
Medium	- Contract consumes unbounded gas. - Block stuffing. - Griefing denial of service. - Gas griefing.
Low	- Contract fails to deliver promised returns, but doesn't lose value.
None	- Best practices. - Gas optimizations.

Project Dashboard

Application Summary

Name	Ragnarok Meta
Commit	5bf362a6
Language	Solidity
Platform	Ethereum

Engagement Summary

Timeline	3 April to 20 April, 2022
Nº of Auditors	2
Review Time	3 person weeks

Vulnerability Summary

Nº Critical Severity Issues	3
Nº High Severity Issues	0
Nº Medium Severity Issues	1
Nº Low Severity Issues	2
Nº None Severity Issues	23
Nº Informational Severity Issues	Several (Slither)

Category Breakdown

Gas Optimization	10
Functional Correctness	5
Access Control	1
Best Practice	13 + Several (Slither)

Code Maturity Evaluation

Category	**Evaluation**
Access Controls	**Satisfactory.** The codebase has strong access control mechanisms.
Arithmetic	**Satisfactory.** The codebase uses Solidity version 0.8.13.
Centralization	**Weak.** The Ragnarok team has significant privileges over the contract.
Code Stability	**Weak.** The code was constantly altered during the audit.
Upgradeability	**Moderate.** Certain parameters of the contract can be modified after deployment.
Function Composition	**Moderate.** Certain components are similar, and the codebase would benefit from increased code reuse.
Front-Running	**Moderate.** Transactions minting NFTs can be frontrun.
Monitoring	**Satisfactory.** Events are correctly emitted.
Specification	**Moderate.** Numerous behaviors were excluded from the available documentation, and the codebase will further benefit from more thorough documentation.
Testing and Verification	**Weak.** There were no tests.

Automated Testing and Verification

To enhance coverage of certain areas of the codebase we complement our manual analysis with automated testing techniques:

- **Slither:** A Solidity static analysis framework with native support for multiple vulnerability detectors. We used Slither to scan the entire codebase against common vulnerabilities and programming malpractices.

Despite augmenting our security analysis, automated testing techniques still present some limitations and should not be used in isolation. Slither may fail to identify vulnerabilities, either due to the lack of specific detectors or whenever certain properties fail to hold after Solidity code is compiled to EVM bytecode. In order to mitigate these risks, we supplemented our automated testing efforts with a careful manual review of the contracts in scope.

Slither Results

During the engagement Slither was executed whenever there was a change in the codebase. All true positive results were communicated to the Ragnarok team and fixed.

Findings

3S-RAG-01

Maximum mint amount exceeded during batch mint

Id	3S-RAG-01
Severity	Critical
Difficulty	Low
Category	Functional Correctness

There is a validation error in the `_firstPublicSaleBatchMint` and `_lastPublicSaleBatchMint` functions, which makes it possible to bypass the mint limits. These functions do not check if the supply cap is exceeded after the user mints a new batch, therefore allowing a user to mint the maximum batch amount of 3 NFTs when only one is available.

Recommendation

Add explicit checks to these functions that account for this scenario.

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-02

Inconsistent counter increment during team sale

Id	3S-RAG-02
Severity	Critical
Difficulty	N/A
Category	Functional Correctness

When calling the `singleMint` function the `_tokenIds` counter value is always incremented by a single unit even when 277 NFTs are minted during the team sale. This causes an internal inconsistency, which results in the subsequent last public sale to fail.

Recommendation

Increment the `_tokenIds` variable by 277 during the team sale.

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-03

Unused merkle root update event

Id	3S-RAG-03
Severity	None
Difficulty	N/A
Category	Best Practice

The `UpdatedMerkleRootOfTeamMint` event is never emitted.

Recommendation

Remove the declaration of the `UpdatedMerkleRootOfTeamMint` event.

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-04

Redundant event parameterization

Id	3S-RAG-04
Severity	None
Difficulty	N/A
Category	Best Practice

The `claimedStatus` parameter of the `ReimbursementClaimedOFPublicSale` event is always `true`. Additionally, the following events, emitted when minting NFTs during distinct sale phases, take a redundant `saleType` parameter already implied in the event naming:

- `NewNFTMintedOnFirstPublicSale`
- `NewNFTBatchMintedOnFirstPublicSale`
- `NewNFTBatchMintedOnLastPublicSale`
- `NewNFTMintedOnPixelSale`
- `NewNFTMintedOnPillSale`
- `NewNFTMintedOnTeamSale`
- `NewNFTMintedOnLastPublicSale`

Recommendation

Refactor the `ReimbursementClaimedOFPublicSale` event to remove the `claimedStatus` parameter. Whether remove the `saleType` parameter from these events, as it is already implied in the event naming, or keep the parameterization and rely on a single event.

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-05

Wrong error parameterization

Id	3S-RAG-05
Severity	Low
Difficulty	N/A
Category	Functional Correctness

Whenever the `_firstPublicSaleBatchMint` and `_lastPublicSaleBatchMint` functions revert with an `InvalidBuyNFTPrice` error, the error contains an incorrect `invalidInputPrice` parameter value. The `msg.value` should not be multiplied by the `tokenCount`. The `msg.value` corresponds to the total ether value sent with the transaction, not the amount sent per minted NFT.

Recommendation

In `_firstPublicSaleBatchMint` and `_lastPublicSaleBatchMint` replace:

```
revert InvalidBuyNFTPrice(
    SafeMath.mul(getPriceOFNFT, tokenCount),
    SafeMath.mul(msg.value, tokenCount)
);
```

with:

```
revert InvalidBuyNFTPrice(
    SafeMath.mul(getPriceOFNFT, tokenCount),
    msg.value
);
```

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-06

SafeMath with Solidity version 0.8.13

Id	3S-RAG-06
Severity	None
Difficulty	N/A
Category	Gas Optimization

The Ragnarok.sol contract is coded targeting Solidity 0.8.13. All versions since 0.8.0 provide checked arithmetic by default. Therefore, the usage of `SafeMath` is redundant and incurs extra gas costs.

Recommendation

Refactor the codebase to remove `SafeMath` and all of its references.

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-07

Tight pack pricing struct

Id	3S-RAG-07
Severity	None
Difficulty	N/A
Category	Gas Optimization

The parameters of the `PublicPricing` struct can be packed together in order to reduce the number of occupied storage slots.

Recommendation

Rearrange the `PublicPricing` struct as:

```
struct PublicPricing {
    uint256 buyPrice;
    address payable ownerAddress;
    bool isClaimed;
}
```

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-08

Unchanged state variables not marked as constant

Id	3S-RAG-08
Severity	None
Difficulty	N/A
Category	Gas Optimization

The contract declares multiple state variables, whose value is set at compile time and never changed during execution, that are not marked as `constant`. By declaring a state variable as `constant`, the compiler does not reserve a storage slot for it but instead replaces each occurrence with its value, therefore removing the gas costs associated with storage reads. Additionally, these state variables are declared as `internal` by default, which makes it difficult for users to access import information regarding the sale details.

Recommendation

Change the visibility of the following variables to `public` and add the `constant` keyword:

- `DEFAULT_MAX_MINTING_SUPPLY`
- `DEFAULT_MAX_FIRST_PUBLIC_SUPPLY`
- `DEFAULT_NFT_PRICE`
- `DEFAULT_DECREASE_NFT_PRICE_AFTER_TIME_INTERVAL`
- `DEFAULT_TIME_INTERVAL`
- `MAX_DECREASE_ITERATIONS`
- `DEFAULT_INITIAL_PUBLIC_SALE`
- `DEFAULT_PIXELMINT_SALE`
- `DEFAULT_PILLMINT_SALE`
- `DEFAULT_TEAMMINT_SALE`
- `LIMIT_IN_PUBLIC_SALE_PER_WALLET`

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-09

Possible to mint zero NFTs using the batch mint function

Id	3S-RAG-09
Severity	Low
Difficulty	N/A
Category	Best Practice

In the `mintBatchToAddress` function there is no explicit check that `tokenCount > 0`. Despite not compromising the safety of the contract it can lead to unexpected side effects when `tokenCount == 0`.

Recommendation

Add a check to the `mintBatchToAddress` function explicitly enforcing that `tokenCount >= 0`.

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-10
OpenZeppelin library bloat

Id	3S-RAG-10
Severity	None
Difficulty	N/A
Category	Gas Optimization

The contract relies on OpenZeppelin libraries to perform simple actions that could alternatively be achieved using native Solidity functionalities. These libraries incur extra gas costs. Additionally, the OpenZeppelin `Address` library is imported and declared for the `address` type but never used.

Recommendation
Replace invocations of `_msgSender` with `msg.sender` when appropriate. Refactor the code to replace the type of `_tokenIds` from `Counters.Counter` with a `uint256`. Remove the `Address` library import and declaration.

Status
Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-11

Unnecessary on-chain computation

Id	3S-RAG-11
Severity	None
Difficulty	N/A
Category	Gas Optimization

The contract provides two functions `singleMint` and `mintBatchToAddress` that users can call in order to participate in the sale. These functions work as a router, which in turn invokes the corresponding internal function depending on the current sale period. In order to perform this routing, the contract must invoke the `checkSaleType` function and perform multiple storage accesses. Additionally, `checkSaleType` is called at every branch of the router if statement.

Recommendation

Move the routing computation off-chain and have the frontend application invoke the correct sale functions. Change the visibility of the `internal` mint functions to `external` and add checks to each one in order to ensure the calls are made during the expected sale phases.

Status

Fixed in commit **5dc1e1e666d9bad5326ce83f668a014431860580**.

3S-RAG-12

Unnecessary mapping variable

Id	3S-RAG-12
Severity	None
Difficulty	N/A
Category	Best Practice

The `teamMintWhitelistedAddresses` variable is used to check whether the team allocation has been minted or not. Using a `mapping` imposes unnecessary trust assumptions on the team as it can mint its allocation multiple times by repeatedly invoking `updatePlatformWalletAddress`.

Recommendation

Refactor `teamMintWhitelistedAddresses` as a `bool`.

Status

Fixed in commit **f03f1ddc304faf7f8ddfb86d8753ddd4ee5ab2b0**.

3S-RAG-13

Unexpected revert behavior in NFT price getter

Id	3S-RAG-13
Severity	None
Difficulty	N/A
Category	Best Practice

The `getCurrentNFTMintingPrice` function contains a condition that reverts in case the sale hasn't started. Since upon deployment, `DEFAULT_SALE_START_TIME` is always set to `block.timestamp`, this condition will never evaluate to `true`. Nonetheless, even if the revert was possible such behavior is discouraged in `view` functions.

Recommendation

Remove the `getCurrentNFTMintingPrice` condition that checks whether the sale has started or not.

Status

Fixed in commit **f03f1ddc304faf7f8ddfb86d8753ddd4ee5ab2b0**.

3S-RAG-14

Empty constructor and declaration of hardcoded values

Id	3S-RAG-14
Severity	None
Difficulty	N/A
Category	Best Practice

The following parameters should be passed as arguments in the constructor instead of being hardcoded:

- `DEFAULT_PLATFORM_ADDRESS`
- `merkleRootOfPixelMintWhitelistAddresses`
- `merkleRootOfPillMintWhitelistAddresses`
- `newUri`

Additionally, the constructor should emit the respective setter events.

Recommendation

Replace the current empty constructor with the following:

```
constructor(
    address payable _defaultPlatformAddress,
    bytes32 _merkleRootOfPixelMintWhitelistAddresses,
    bytes32 _merkleRootOfPillMintWhitelistAddresses,
    string memory _newUri
) public ERC1155(_newUri) {
    // set storage values
    DEFAULT_PLATFORM_ADDRESS = _defaultPlatformAddress;
    merkleRootOfPixelMintWhitelistAddresses = _merkleRootOfPixelMintWhitelistAddresses;
    merkleRootOfPillMintWhitelistAddresses = _merkleRootOfPillMintWhitelistAddresses;

    // emit setter events
    emit UpdatedPlatformWalletAddress(_defaultPlatformAddress, msg.sender);
    emit UpdatedMerkleRootOfPixelMint(_merkleRootOfPixelMintWhitelistAddresses, msg.sender);
    emit UpdatedMerkleRootOfPillMint(_merkleRootOfPillMintWhitelistAddresses, msg.sender);
    emit NewURI(_newUri, msg.sender);
}
```

Status

Fixed in commit **f03f1ddc304faf7f8ddfb86d8753ddd4ee5ab2b0**.

3S-RAG-15

Wasted gas due to immediate transfers

Id	3S-RAG-15
Severity	None
Difficulty	N/A
Category	Gas Optimization

Transferring funds to `DEFAULT_PLATFORM_ADDRESS` upon every sale adds an unnecessary gas overhead, which negatively impacts user experience.

Recommendation

Add a `withdraw` function callable only by `DEFAULT_PLATFORM_ADDRESS` or the `owner` to withdraw deposited funds from multiple purchases in a single call.

Status

Fixed in commit **f03f1ddc304faf7f8ddfb86d8753ddd4ee5ab2b0**.

3S-RAG-16

Unnecessary address validation and error parameterization

Id	3S-RAG-16
Severity	None
Difficulty	N/A
Category	Gas Optimization

The `singleMint` and `mintBatchToAddress` functions revert with an error of `InvalidAddress` whenever `msg.sender == address(0)`. This is an unnecessary check for a scenario that should never occur. Additionally, `InvalidAddress` is always parameterized with the same `invalidAddress` value of zero.

Recommendation

Remove checks for `msg.sender == address(0)` from both the `singleMint` and `mintBatchToAddress` functions.

Status

Fixed in commit **f03f1ddc304faf7f8ddfb86d8753ddd4ee5ab2b0**.

3S-RAG-17

Wrong increment on the total minted variable

Id	3S-RAG-17
Severity	Critical
Difficulty	Low
Category	Functional Correctness

Starting with the value of 1, the `_tokenIds` variable corresponds to the number of NFTs minted. During the first and last public sales, when checking whether the supply limit has been reached the latest value is compared with `DEFAULT_MAX_FIRST_PUBLIC_SUPPLY` and `DEFAULT_MAX_MINTING_SUPPLY` respectively. The comparison is made using the `>=` operator, which expects `_tokenIds` to start at 0. This causes one less NFT to be mintable during each public sale, 3899 and 7776 instead of 3900 and 7777 respectively.

Recommendation

Replace the `>=` operator with `>` in the previous conditions.

Status

Fixed in commit **ee9f4077af977214f6f1aa403030134fe11a8187**.

3S-RAG-18

Constructor visibility

Id	3S-RAG-18
Severity	None
Difficulty	N/A
Category	Best Practice

The Ragnarok.sol constructor has a `public` visibility modifier. Starting from Solidity version 0.7.0 visibility modifiers are obsolete for constructors.

Recommendation

Remove the `public` modifier from the constructor.

Status

Fixed in commit **b1f32ecb7aac61cb4d0e2e7e51cca36e830a4468**.

3S-RAG-19

Events are emitted out of order

Id	3S-RAG-19
Severity	None
Difficulty	N/A
Category	Best Practice

The `_mint` function of ERC1155.sol performs an unsafe external call to the minter address if the minter is a contract. In Ragnarok.sol the `_mint` function is invoked before emitting the corresponding mint event. This can lead to mint events with out of order NFT ids in case the function reenters, which might result in issues for third party applications.

Recommendation

Emit the mint event before calling the `_mint` function in all `external` mint functions of Ragnarok.sol, according to the following example:

```
// omitted ...
emit NewNFTMintedOnFirstPublicSale(
    _tokenIds,
    msg.sender,
    msg.value
);
_mint(msg.sender, _tokenIds, 1, "");
return true;
```

Status

Fixed in commit **b1f32ecb7aac61cb4d0e2e7e51cca36e830a4468**.

3S-RAG-20

External call inside a loop

Id	3S-RAG-20
Severity	Medium
Difficulty	Low
Category	Functional Correctness

The `reimbursementAirdrop` function sends a refund to all addresses specified in the `addresses` parameter by iterating through the recipients inside a loop. If one of the recipients has a `fallback`/`receive` function that reverts or consumes infinite gas `reimbursementAirdrop` will always revert.

Recommendation

Reimburse all addresses manually, or execute independent calls reimbursing the addresses.

Status

The team acknowledged the issue and decided to perform a manual address triage prior to invoking the `reimbursementAirdrop` function.

3S-RAG-21

Redundant whenNotPaused modifier

Id	3S-RAG-21
Severity	None
Difficulty	N/A
Category	Best Practice

The `whenNotPaused` modifier was added to all the mint functions of Ragnarok.sol in commit **f4060ad41df007d8c3495ee7de40320853750a8c.** These functions call `_mint` internally, which already performs an equivalent `whenNotPaused` check.

Recommendation

Remove the `whenNotPaused` modifier from:

- `firstPublicMintingSale`
- `pixelMintingSale`
- `pillMintingSale`
- `teamMintingSale`
- `lastPublicMintingSale`
- `firstPublicSaleBatchMint`
- `lastPublicSaleBatchMint`

Status

Fixed in commit **b1f32ecb7aac61cb4d0e2e7e51cca36e830a4468**.

3S-RAG-22

Costly operations inside a loop

Id	3S-RAG-22
Severity	None
Difficulty	N/A
Category	Gas Optimization

The `teamMintingSale` function increments the `_tokenIds` storage variable in a large loop, which incurs unnecessary gas costs.

Recommendation

Use a local variable to hold the loop computation result.

Status

Fixed in commit **b1f32ecb7aac61cb4d0e2e7e51cca36e830a4468**.

3S-RAG-23

Dead code

Id	3S-RAG-23
Severity	None
Difficulty	N/A
Category	Best Practice

The `SaleNotStartedYet` and `InvalidAddress` errors are declared but never used.

Recommendation

Remove dead code.

Status

Fixed in **6978d4074eba1b4dec91970bcaf22218f8f9ba6b**.

3S-RAG-24

Improper custom error naming and parameterizations

Id	3S-RAG-24
Severity	None
Difficulty	N/A
Category	Best Practice

In the `InvalidTokenCount` error the `tokenCount` parameter is always 0. Both the `status` and `data` error parameterizations of `AmountReimbursementFailed` and `TransactionFailed` are unnecessary. The `MaximumMintLimitReached` error is only used once, in the same circumstances as `MaximumMintLimitReachedByUser`, and doesn't accurately describe the behavior that triggers it.

Recommendation

Rename `InvalidTokenCount`, remove unused error parameters, and rename `MaximumMintLimitReached` to `MaximumMintLimitReachedByUser`.

Status

Fixed in **6978d4074eba1b4dec91970bcaf22218f8f9ba6b**.

3S-RAG-25

Improper use of constants

Id	3S-RAG-25
Severity	None
Difficulty	N/A
Category	Best Practice

The `LIMIT_IN_PUBLIC_SALE_PER_WALLET` constant is never used.
The team allocation amount is hardcoded in multiple places with the value 227.

Recommendation

Replace the hardcoded value 3 with references to the declared constant `LIMIT_IN_PUBLIC_SALE_PER_WALLET`. The team allocation amount should be declared as a `constant` value and be referenced instead of using the hardcoded value.

Status

Fixed in **6978d4074eba1b4dec91970bcaf22218f8f9ba6b.**

3S-RAG-26

Unnecessary condition checking

Id	3S-RAG-26
Severity	None
Difficulty	N/A
Category	Gas Optimization

In the mint functions, invocations of `checkSaleType` validate unnecessary conditions and execute redundant logic.

Recommendation

Invocations of `checkSaleType` should be replaced with a single condition check, following:

```
function firstPublicMintingSale() external payable returns (bool) {
    // check if first public sale is ongoing
    if (
        (block.timestamp < DEFAULT_SALE_START_TIME) ||
        (block.timestamp ≥ DEFAULT_SALE_START_TIME +
DEFAULT_INITIAL_PUBLIC_SALE)
    ) {
        revert UnAuthorizedRequest();
    }
    // ...
}
```

Status

Fixed in **6978d4074eba1b4dec91970bcaf22218f8f9ba6b**.
Note that the Ragnarok team opted for keeping the `checkSaleType` function, due to frontend requirements, despite replacing its invocations in the mint functions.

3S-RAG-27

Effectless function behavior

Id	3S-RAG-27
Severity	None
Difficulty	N/A
Category	Best Practice

The `claimAmountFirstPublicSale` accepts reimbursement claims before the start of the first public sale. Despite not compromising the safety of the contract, as there are no funds deposited prior to the beginning of the sale, the function does not revert as expected.

Recommendation

Add a check that prevents calling `claimAmountFirstPublicSale` before the start of the sale.

Status

Fixed in **f4060ad41df007d8c3495ee7de40320853750a8c**.

3S-RAG-28

Explicit whitelist mint limits

Id	3S-RAG-28
Severity	None
Difficulty	N/A
Category	Access Control

There are no explicit mint limit checks for the pixel and pill whitelist sales.

Recommendation

Add explicit checks in the code that enforce whitelisted mint limits.

Status

It is assumed that the merkle trees enforce the proper NFT mint limits per phase.

3S-RAG-29

Redundant condition check

Id	3S-RAG-29
Severity	None
Difficulty	N/A
Category	Gas Optimization

In the `firstPublicSaleBatchMint` and `lastPublicSaleBatchMint` functions the `firstPublicSale[msg.sender] >= 3` check is redundant and can be safely removed.

Recommendation

Remove the previous redundant condition check.

Status

Fixed in **b1f32ecb7aac61cb4d0e2e7e51cca36e830a4468.**