



PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Avalaunch
(NFT Marketplace)

19 May 2023



paladinsec.co



info@paladinsec.co

Table of Contents

Table of Contents	2
Disclaimer	3
1 Overview	4
1.1 Summary	4
1.2 Contracts Assessed	4
1.3 Findings Summary	5
1.3.1 AvalaunchNFTFactory	6
1.3.2 AvalaunchNFT	6
1.3.3 AvalaunchNFTMarketplace	7
1.3.4 AvalaunchNFTErrors	7
1.3.5 BatchReveal	8
2 Findings	9
2.1 AvalaunchNFTFactory	9
2.1.1 Privileged Functions	9
2.1.2 Issues & Recommendations	10
2.2 AvalaunchNFT	14
2.2.1 Privileged Functions	14
2.2.2 Issues & Recommendations	15
2.3 AvalaunchNFTMarketplace	24
2.3.1 Privileged Functions	24
2.3.2 Issues & Recommendations	25
2.4 AvalaunchNFTErrors	39
2.4.1 Issues & Recommendations	39
2.5 BatchReveal	40
2.5.1 Issues & Recommendations	41

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 advocacy 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 the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Avalaunch's NFT marketplace contracts 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	Avalaunch
URL	https://avalaunch.app/
Platform	Avalanche
Language	Solidity
Preliminary	https://github.com/avalaunch-app/nft-marketplace/tree/3fdadad782482dd31cb6bfa55dd57139430ddd33/contracts

1.2 Contracts Assessed

Name	Contract	Live Code Match
AvalaunchNFTFactory	0xbd27ECE0637518C519aa4323D93281AD179CA65b	UNVERIFIED
AvalaunchNFT	0xE29561fdc8A759742403D537DD1fadCaba0B346A	UNVERIFIED
AvalaunchNFTMarketplace	0x4c7cCD59Df34F6F86311C4ad5cE922f9C83c6C3c	UNVERIFIED
AvalaunchNFTErrors	NA	N/A

The Avalaunch team has decided to keep these contracts unverified on the blockchain, so users should take note of this when interacting with the contracts. **We are unable to verify whether the deployed code match the audited code.**

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	8	8	-	-
● Medium	8	6	-	2
● Low	9	5	-	4
● Informational	9	6	1	2
Total	34	25	1	8

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 with 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 AvalaunchNFTFactory

ID	Severity	Summary	Status
01	INFO	Lack of validation for initialize values	ACKNOWLEDGED
02	INFO	Unnecessary Ownable() constructor call	✓ RESOLVED
03	INFO	getAllSales should be getAllDeployments	✓ RESOLVED
04	INFO	Typographical errors	ACKNOWLEDGED

1.3.2 AvalaunchNFT

ID	Severity	Summary	Status
05	HIGH	A valid signature can be used multiple times	✓ RESOLVED
06	HIGH	Minting can happen without a set coordinator	✓ RESOLVED
07	MEDIUM	VRFCoordinator can only be set once	✓ RESOLVED
08	MEDIUM	AVAX refund is flawed	✓ RESOLVED
09	MEDIUM	Cross-domain signature replay	ACKNOWLEDGED
10	LOW	Royalty percentage can be changed without timelock	ACKNOWLEDGED
11	LOW	Lack of validation for withdrawEarnings	✓ RESOLVED
12	LOW	Pausing functionality is unused	✓ RESOLVED
13	LOW	Incorrect tokenUri will be returned for non-existing/not minted NFTs	ACKNOWLEDGED
14	INFO	Typographical issues	✓ RESOLVED

1.3.3 AvalaunchNFTMarketplace

ID	Severity	Summary	Status
15	HIGH	claimAuctionItem can be called an arbitrary number of times	✓ RESOLVED
16	HIGH	Refund of AVAX might revert	✓ RESOLVED
17	HIGH	Users can DoS any auction by extending the endTime	✓ RESOLVED
18	HIGH	Multiple re-entrancy attack vectors allow anyone to drain AvalaunchNFTMarketplace's native tokens balance	✓ RESOLVED
19	HIGH	Malicious buyer can frontrun acceptAsk	✓ RESOLVED
20	HIGH	Contract owner has the ability to DoS	✓ RESOLVED
21	MEDIUM	Seller can blackmail the highest bidder	✓ RESOLVED
22	MEDIUM	Funds can potentially get stuck within claimAuctionItem	✓ RESOLVED
23	MEDIUM	Owner can frontrun fee change, resulting in a loss of user funds	ACKNOWLEDGED
24	MEDIUM	Lack of validation for timeExtensionPerBid	✓ RESOLVED
25	LOW	onlyExistingAuction modifier uses standard items as check	✓ RESOLVED
26	LOW	Owner can circumvent accepting offers	ACKNOWLEDGED
27	LOW	Multiple functions can be abused by sellers	✓ RESOLVED
28	INFO	Ownable initialization is unnecessary	✓ RESOLVED
29	INFO	setMinTimeExtensionsPerBid is unused	✓ RESOLVED
30	INFO	Contract does not adhere to CEI pattern	PARTIAL
31	INFO	Typographical issues	✓ RESOLVED

1.3.4 AvalaunchNFTErrors

No issues found.

1.3.5 BatchReveal

ID	Severity	Summary	Status
32	MEDIUM	Possible DoS in case of unsuccessful callback	✓ RESOLVED
33	LOW	_buildJumps might run out of gas	ACKNOWLEDGED
34	LOW	Wrong storage __gap size	✓ RESOLVED

2 Findings

2.1 AvalaunchNFTFactory

AvalaunchNFTFactory is a factory contract for the purpose of deploying of AvalaunchNFT contracts. The contract owner can deploy multiple instances (minimal proxy contracts) pointing to the implementation. Each deployment is made with the create opcode which deploys the proxy independent from the bytecode under the following address:

```
address = keccak256(rlp([sender_address, sender_nonce]))[12:]
```

The assembly code used for the deployment is from the OpenZeppelin library Clones.sol (<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/proxy/Clones.sol>)

The deployed proxy then simply interacts with the implementation contract via delegatecalls; however, it is not upgradeable.

2.1.1 Privileged Functions

- transferOwnership
- renounceOwnership
- setImplementation
- deploy

2.1.2 Issues & Recommendations

Issue #01	Lack of validation for initialize values
Severity	 INFORMATIONAL
Description	After the clone has been deployed, it is initialized directly. However, there is no validation for the variables used for initialization — for example, <code>_collectionName</code> is not checked to be unique which might confuse users.
Recommendation	Consider being careful with the selection of <code>_collectionName</code> and <code>_collectionSymbol</code> during deployment. The team could also consider implementing logic that does not allow deploying clones with an already existent <code>_collectionSymbol</code>. However, we do not think a contract adjustment is necessary here as long as the developer is mindful of this.
Resolution	 ACKNOWLEDGED

Issue #02	Unnecessary Ownable() constructor call
Severity	 INFORMATIONAL
Description	It is unnecessary to call the constructor of the ownable contract as it is automatically called.
Recommendation	Consider removing this call.
Resolution	 RESOLVED

Issue #03	getAllSales should be getAllDeployments
Severity	 INFORMATIONAL
Description	The function actually returns all deployments and not sales. This name might confuse third-parties.
Recommendation	Consider renaming this function.
Resolution	 RESOLVED



DescriptionLine 4

```
import "@openzeppelin/contracts/proxy/Clones.sol";
```

The `Clones.sol` library from OpenZeppelin is imported but not used. Instead, the code from `Clones.clone` is copied and used in `AvalaunchNFTFactory.deploy`.

Consider either removing the `import` statement or using `Clones.clone`.

Line 24-25

```
require(owner != address(0));  
transferOwnership(owner);
```

In the constructor of `AvalaunchNFTFactory`, a zero address check is made but `transferOwnership` is used which also checks for the zero address and that the `msg.sender` is owner of this contract. These checks are unnecessary. Consider using `_transferOwnership` instead of `transferOwnership`.

Line 69-76

```
assembly {  
    // Cleans the upper 96 bits of the implementation word,  
    then packs the first 3 bytes  
    // of the implementation address with the bytecode before  
    the address.  
    mstore(0x00, or(shr(0xe8, shl(0x60, imp)),  
0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000))  
    // Packs the remaining 17 bytes of implementation with  
    the bytecode after the address.  
    mstore(0x20, or(shl(0x78, imp),  
0x5af43d82803e903d91602b57fd5bf3))  
    clone := create(0, 0x09, 0x37)  
}
```

The `AvalaunchNFTFactory` deploys EIP1167 minimal proxies which are non-upgradeable and cheaper in terms of deployment gas cost. However, proxies add ~2300 additional gas per each function call due to the `delegatecall` opcode used each time a function is invoked. Consider whether the `AvalaunchNFT` contract will have to be deployed frequently or it would be more efficient to save runtime gas cost per each function call by simply deploying normal `AvalaunchNFT` contracts (not behind a proxy).

Recommendation Consider fixing the typographical errors.

Resolution

 ACKNOWLEDGED



2.2 AvalaunchNFT

AvalaunchNFT is the implementation contract used for the minimal proxy deployment within AvalaunchNFTFactory. It is a customized ERC721A contract which implements the ERC2981 royalty standard and the BatchReveal logic using Chainlink VRF. The BatchReveal logic is out of scope and therefore only partially included in this audit, however, the general assumption is that this contract is bug-free.

Users can mint NFTs using a valid signature which was signed previously by the owner of this contract. This signature determines the price per unit and the quantity.

Finally, after users have minted NFTs with AVAX, the contract owner can withdraw all AVAX in the contract via `withdrawEarnings`. The royalty can be set by the owner up to 15% within the `setDefaultRoyalty` function.

2.2.1 Privileged Functions

- `setVRF`
- `setBaseURI`
- `setUnrevealedURI`
- `setDefaultRoyalty`
- `withdrawEarnings`

2.2.2 Issues & Recommendations

Issue #05	A valid signature can be used multiple times
Severity	 HIGH SEVERITY
Description	The signature determines how many tokens to which price can be minted. However, there is no check if this signature has already been used, which means a malicious user can use the same signature repeatedly to mint until the <code>collectionSize</code> is reached.
Recommendation	Add a nonce to the message payload that has been signed by the owner and increment it each time after signature verification has passed.
Resolution	 RESOLVED A hash usage marker was added.

Issue #06**Minting can happen without a set coordinator****Severity** HIGH SEVERITY**Description**

At first glance, it does not seem it is an issue that minting can be done even when the Chainlink coordinator is not set. However, the following exploit can occur:

1. Malicious user John inspects the contract deployment.
2. John mints a whole batch size of tokens.
3. John now calls `reveal` while manipulating the pseudo randomness.
4. Since the `reveal` function automatically uses pseudo randomness when the coordinator is not set, John can successfully manipulate it.

An attacker can manipulate the received `tokenIds` in the above manner.

Recommendation

Consider implementing a boolean which must be set to `true` by the owner before any minting can happen.

Resolution RESOLVED

`reveal` can only be called by the owner.

Issue #07	VRFCoordinator can only be set once
Severity	 MEDIUM SEVERITY
Description	VRFCoordinator can only be set once. While this is normally not a problem, the subscription methodology might cause an issue. If the subscription contract decides to delist the AVALaunchNFT contract as a consumer contract, batches cannot be revealed anymore. Another scenario, while unlikely to happen, which can be a problem is the removal of the used keyHash from the Coordinator contract.
Recommendation	Consider either allowing for the subscriptionID to be changed or simply allow the contract owner to change isUsingVRF so they are able to react to such events.
Resolution	 RESOLVED The owner can disable the functionality now.



Issue #08	AVAX refund is flawed
Severity	 MEDIUM SEVERITY
Description	The refund method when transferring the leftover amount back to the user is flawed: <pre>if (msg.value > total) { (bool success,) = msg.sender.call{value: total - msg.value}(""); if (!success) { revert AvalaunchNFT__AVAXTransferFailed(); } However, since msg.value is larger than total, this function will revert due to an underflow.</pre>
Recommendation	Consider calculating it in the proper direction: <pre>msg.value - total;</pre>
Resolution	 RESOLVED

Issue #09**Cross-domain signature replay****Severity** MEDIUM SEVERITY**Location**L177-185

```
bytes32 hash = keccak256(
    abi.encodePacked(
        msg.sender,
        address(this),
        quantity,
        pricePerUnit,
        sigExpTime
    )
);
```

Description

AvalaunchNFT.mint allows users to mint NFTs if they can provide a valid signature from the owner of the contract. The message that has to be signed simply follows packed values: msg.sender, the address of the contract, the quantity to mint, price per token, and signature expiration time.

If similar types of variables with the same absolute values are used in a different protocol and the owner of the contract has signed this message for a different purpose, the signature can be reused in AvalaunchNFT.mint.

EIP712 introduces the concept of a domain separator that includes several arguments unique for each protocol to prevent such types of vulnerabilities.

Recommendation

Consider using an EIP712 domain separator for the message payload and signature verification in AvalaunchNFT.mint.

Resolution ACKNOWLEDGED

Issue #10 Royalty percentage can be changed without timelock	
Severity	 LOW SEVERITY
Description	The royalty percentage can be changed by the owner to a maximum value of 15%. However, the owner can change this value at any time, which can lead to unexpected losses for sellers on compatible marketplaces.
Recommendation	Consider implementing a timelock feature for this function as well as openly communicating all changes.
Resolution	 ACKNOWLEDGED The Avalaunch team will communicate such changes to the community with an appropriate lead time.

Issue #11 Lack of validation for withdrawEarnings	
Severity	 LOW SEVERITY
Description	withdrawEarnings lacks a proper validation for the to address. In the worst case scenario, this can be address(0) which results in a loss of all AVAX.
Recommendation	Consider validating this parameter accordingly.
Resolution	 RESOLVED

Issue #12	Pausing functionality is unused
Severity	 LOW SEVERITY
Description	The contract imports the PausableUpgradeable library from OZ. However, it is not actively used in the contract.
Recommendation	Consider implementing it correctly or simply removing it.
Resolution	 RESOLVED The extension was removed.

Issue #13	Incorrect tokenUri will be returned for non-existing/not minted NFTs
Severity	 LOW SEVERITY
Location	<u>Line 263-264</u> <pre>if (id >= lastTokenRevealed) { return unrevealedURI;</pre>
Description	tokenURI is overridden and returns either the unrevealedURI which is set by the contract owner or the normal URI which composes of the baseURI, shuffled token id and suffix. According to the EIP721 specification, the tokenURI function “throws if _tokenId is not a valid NFT” while in the current code it just returns the unrevealedURI. This may cause unexpected behavior in front-end apps or third-party contracts.
Recommendation	Consider reverting tokenURI in case the passed id is not yet minted or exceeds the collectionSize.
Resolution	 ACKNOWLEDGED

DescriptionLine 84`__Ownable_init();`

* This call is unnecessary because the owner will be transferred to owner anyways.

Line 103

* @param _keyHash is

Missing NatSpec comment for _keyHash in setVRF.

Line 105

* @param _callbackGasLimit gas limit for the coordinator callbacks

"gas" is misspelled.

Line 172

* @param pricePerUnit is single nft price

"single" is misspelled.

Line 212`_mint(_msgSender(), quantity, "", false);`

Inconsistently using msg.sender and _msgSender() in mint.

Line 230

* @notice function to set base uri

setBaseUri also sets the suffix.

Line 256

* @notice function to retrieve token id

Wrong @notice comment for tokenURI

Line 271

`bytes(suffix).length > 0 ? suffix : ""`

Redundant check in tokenURI for suffix length.

Recommendation Consider fixing the listed typographical issues.

Resolution



2.3 AvalaunchNFTMarketplace

AvalaunchNFTMarketplace is the implementation contract used by the Avalaunch team for the proxy contract as NFT marketplace.

The marketplace has three main business logics:

1. Regular selling of items: An address can list an item for sale to a specific address and the corresponding address can purchase this item for the agreed sale price. The seller can also list an item with address(0) as buyer which then allows anyone to purchase it.
2. Ask for items on sale: An address can create an ask for a listed item and suggest a price-offer and the owner can then eventually accept this offer. This will only work for publicly-listed items.
3. A regular auction: An address can list an item as auction and other addresses can then bid on this auction.

The marketplace revenue is generated during a direct sale, an accepted offer and a finalized auction and is taken in AVAX. The contract owner can then withdraw these funds via the `withdrawFees` function and has the privilege to set this fee up to 20%.

2.3.1 Privileged Functions

- `setFeeParameters`
- `setMinTimeExtensionPerBid`
- `transferOwnership`
- `renounceOwnership`

2.3.2 Issues & Recommendations

Issue #15	claimAuctionItem can be called an arbitrary number of times
Severity	 HIGH SEVERITY
Description	claimAuctionItem does not check whether the item is still available, which opens the possibility for an exploit where the buyer can use a secondary address in order to purchase their own NFT, then the seller can call claimAuctionItem for an arbitrary number of times in order to drain all AVAX from the contract (of course he would need to manually transfer the NFT to the marketplace again in order to make the function call successful). Additionally, a buyer can call claimAuctionitem in the future when he realizes that the NFT has been listed again in an attempt to steal it.
Recommendation	Consider ensuring that claimAuctionItem can only be called when the item is still available.
Resolution	 RESOLVED A check if the item is still available has been implemented.

Issue #16	Refund of AVAX might revert
Severity	 HIGH SEVERITY
Description	The bid function refunds AVAX from the previous highest bidder to their address. However, it is possible to use a smart contract and bid without a fallback function to receive AVAX. This would effectively prevent all future bidders from successful bidding, allowing the attacker to claim the item for the start price. <i>*The smart contract can still get seeded via a simple deposit payable function.</i>
Recommendation	Consider preventing contracts from bidding or switching the refund logic.
Resolution	 RESOLVED
	Only EOAs are allowed to bid.



Issue #17**Users can DoS any auction by extending the endTime****Severity** HIGH SEVERITY**Description**

endTime is extended every time a new bid occurs. A malicious user can therefore DoS the auction by always bidding shortly before the auction can be claimed. However, this means that the user will need to lock his own funds for the time being and the owner could counter-bid the attacker with a higher value in order to force the attacker to bid even higher.

Recommendation

This fix is non-trivial, consider switching to a different logic than extending the endTime per bid.

Resolution RESOLVED

The minimum bid must be more than 10% higher than the previous bid — this effectively prevents the attack vector.



Description

There are multiple re-entrancy and cross-function re-entrancy vulnerabilities that allow bidders and sellers to drain all AVAX tokens stored in the marketplace.

Line 289-290

```
// Return AVAX taken for previously set ask  
safeFundTransfer(msg.sender, ask.value, false);
```

`AvalaunchNFTMarketplace.createAsk` transfers the previously asked value to the `msg.sender`, but neither has the `nonReentrant` modifier nor follows the CEI pattern as `ask.value` is updated after the external call (and is not set to 0 before the call).

This exposes at least 3 re-entrancy attack vectors:

- asker can re-enter `.createAsk` and drain the contract
- asker can re-enter into `.removeAsk` and receive `ask.value` twice at the cost of just 1 wei (the `msg.value`)
- attacker contract can list an NFT item, then call `createAsk` from another address (let's say `addressAsker`), `addressAsker` calls `.createAsk` a second time (`msg.value = 1 wei`), `safeFundTransfer` makes the untrusted call to `addressAsker`, `addressAsker` invokes a function on the attacker contract which calls `acceptAsk`. The result is that the attacker receives their NFT and `ask.value` back, but also stole additional `ask.value` similar to double-spending.

Lines 312-313

```
// Return AVAX taken ask  
safeFundTransfer(msg.sender, ask.value, false);
```

`AvalaunchNFTMarketplace.removeAsk` transfers the currently asked value to the asker, but neither has the `nonReentrant` modifier nor follows the CEI pattern as `ask.active` is updated after the external call.

This again exposes at least 3 re-entrancy attack vectors:

- asker can re-enter `.removeAsk` and drain the contract
- asker can re-enter into `.createAsk` and receive `ask.value` twice at the cost of just 1 wei (the `msg.value`)
- attacker contract can list an NFT item, then call `createAsk` from another address (let's say `addressAsker`), `addressAsker` calls `.removeAsk`, `safeFundTransfer` makes the untrusted call to `addressAsker`, `addressAsker` invokes a function on the attacker contract which calls `acceptAsk`. The result is that the attacker received their NFT and `ask.value` back, but also stole additional `ask.value` similar to double-spending.

Lines 442-445

```
// Return funds to the previous bidder
if (item.highestBid > 0) {
    safeFundTransfer(item.bidder, item.highestBid, false);
}
```

`AvalaunchNFTMarketplace.bid` returns the previous highest bid to the previous highest bidder when they are outbid, but neither has the `nonReentrant` modifier nor follows the CEI pattern as `item.bidder`, `item.highestBid` and `item.endTime` are updated after the external call.

Note that both `.bid` and `.claimAuctionItem` can be called when `block.timestamp == item.endTime`.

`AvalaunchNFTMarketplace.bid`:

```
// Check if auction is in progress
if (block.timestamp > item.endTime && item.endTime > 0) {
    revert AvalaunchNFTMarketplace__AuctionEnded();
}
```

`AvalaunchNFTMarketplace.claimAuctionItem`:

```
// Check if auction is in progress
if (block.timestamp < item.endTime || item.endTime == 0) {
    revert AvalaunchNFTMarketplace__AuctionInProgress();
}
```

This opens another cross-function re-entrancy vector where a malicious bidder (potentially an account controlled by the seller) can `.bid` and wait for another bidder to bid exactly at the last second of the auction: `block.timestamp = item.endTime`.

If the above case occurs, the outbidded malicious bidder/seller can re-enter into `.claimAuctionItem` and claim their NFT as well as transfer the bid amount to the seller.

`AvalaunchNFTMarketplace.bid` will continue to execute and set the `item.bidder` to the victim and consume/lock the sent bid (`msg.value`). After the extended `.endTime` passes, the victim will be able to call `claimAuctionItem`, but the function will revert since the NFT has already been claimed.

Recommendation Add the `nonReentrant` modifier to the following functions and follow the Checks-Effects-Interactions pattern:

- `createAsk`
- `removeAsk`
- `bid`

Resolution



Issue #19	Malicious buyer can frontrun acceptAsk
Severity	 HIGH SEVERITY
Description	It is possible for a buyer to change the value from an already existing ask. A sophisticated hacker can therefore make a generous offer for an existing item, inspect the acceptAsk function in the mempool and frontrun it with a createAsk call with a very low value such as 1 wei in order to trick the seller into accepting it. The hacker now successfully bought the NFT for 1 wei.
Recommendation	Consider removing or timelocking the ability to change the ask value.
Resolution	 RESOLVED
	The check has been added and the expirationTimestamp logic has been removed.



Issue #20	Contract owner has the ability to DoS
Severity	 HIGH SEVERITY
Description	The contract owner can change the fees as already mentioned up to 20%. However, it is not validated that <code>feePrecision != 0</code>, which can lead to a division by zero error. If that is fixed, there is another possible method of DoS: The owner can set the <code>feePrecision</code> to $2^{256} - 1$ which allows the <code>feePercent</code> to be set to $(2^{256} - 1) / 5$ which results in an overflow during the fee calculation depending on the amount: $\text{fee} = (\text{amount} * \text{feePercent}) / \text{feePrecision};$
Recommendation	Consider validating that <code>feePrecision</code> can never be zero, and also validate that <code>feePrecision</code> can only have a reasonable upper limit.
Resolution	 RESOLVED

Issue #21	Seller can blackmail the highest bidder
Severity	 MEDIUM SEVERITY
Description	Within <code>AvalaunchNFTMarketplace.claimAuctionItem</code>, the NFT item is sent to the highest bidder and the highest bid paid for it is sent to the seller of the NFT. The seller can make the transfer of native tokens always revert in order to blackmail the bidder for a higher price.
Recommendation	Consider implementing the withdrawal pattern instead of sending the bid directly to the seller.
Resolution	 RESOLVED

Issue #22	Funds can potentially get stuck within <code>claimAuctionItem</code>
Severity	 MEDIUM SEVERITY
Location	<u>Line 500</u> <pre>(bool success,) = _to.call{value: _amount}(_msg ? MSG : new bytes(0x00));</pre>
Description	Whenever <code>safeFundTransfer</code> is called from <code>claimAuctionItem</code>, <code>acceptAsk</code> and <code>buyItem</code>, the following message will be sent to the receiver (the seller) as a calldata: <pre>bytes public constant MSG = bytes("Your item got sold! Greetings from Avalaunch team :)");</pre> This will cause sending funds to proxy contracts such as Gnosis Safe wallets to always revert since this will be interpreted as a call to a function with a selector <code>0x596f7572</code>, which will most likely not be a present function. This can be critical in <code>AvalaunchNFTMarketplace.claimAuctionItem</code> since if, for example, the <code>item.seller</code> is a Gnosis Safe wallet, the transaction will always revert, meaning that the NFT and the highest bid will be stuck in the marketplace.
Recommendation	Consider removing MSG.
Resolution	 RESOLVED



Issue #23	Owner can frontrun fee change, resulting in a loss of user funds
Severity	 MEDIUM SEVERITY
Description	The owner can change the fee via setFeeParams at any time to frontrun a huge purchase and set the fee to 20% in order to gain unauthorized funds. This issue is only of medium severity because the fees have an upper limit of 20%.
Recommendation	Consider timelocking this feature and also consider announcing any changes to the community.
Resolution	 ACKNOWLEDGED

Issue #24	Lack of validation for timeExtensionPerBid
Severity	 MEDIUM SEVERITY
Description	There is no validation for timeExtensionPerBid, which means that an auction owner can use a very high value which can result in a DoS state, although this would also lock the owners NFT.
Recommendation	Consider using an appropriate upper limit for this value.
Resolution	 RESOLVED



Issue #25**onlyExistingAuction modifier uses standard items as check****Severity** LOW SEVERITY**Description**

The onlyExistingAuction modifiers should ensure that no call (removeAuctionItem, bid and claimAuctionItem) can happen on existing items. However, it uses the regular listed items for this check:

```
modifier onlyExistingAuction(uint256 itemId) {  
    _onlyExisting(itemId);  
    -;  
}  
  
function _onlyExisting(uint256 _itemId) private view {  
    if (items.length <= _itemId) {  
        revert AvalaunchNFTMarketplace__InvalidItemId();  
    }  
}
```

These modifiers are also not necessary since the array access would revert anyway.

Recommendation

Consider removing these modifiers.

Resolution RESOLVED

Issue #26 Owner can circumvent accepting offers	
Severity	 LOW SEVERITY
Description	In traditional auctions, the owner is forced to accept the latest offer; however, within this contract, the owner can simply counter-bid the undesired bid in and then claim his own NFT. This issue is only marked as low because the owner will need to pay the fee for doing so.
Recommendation	A fix is non-trivial, there is no solution to ensure any other EOA is not the owner.
Resolution	 ACKNOWLEDGED

Issue #27 Multiple functions can be abused by sellers	
Severity	 LOW SEVERITY
Description	Functions such as buyItem and claimAuctionItem can be made to always revert or either reverting the transaction in their fallback function or consuming all the gas sent in specific cases that the seller may find profitable.
Recommendation	Consider implementing the withdrawal pattern.
Resolution	 RESOLVED

Issue #28 Ownable initialization is unnecessary	
Severity	 INFORMATIONAL
Description	Currently, the initialize function calls <code>__Ownable_init()</code> . However, ownership is transferred to <code>_owner</code> anyway, therefore this call is unnecessary.
Recommendation	Consider removing this call.
Resolution	 RESOLVED

Issue #29 setMinTimeExtensionsPerBid is unused	
Severity	 INFORMATIONAL
Description	The above function and the whole associated logic is unused.
Recommendation	Consider removing this function and all logic associated with it if it's not desired, however, we strongly recommend implementing this functionality correctly.
Resolution	 RESOLVED



Issue #30	Contract does not adhere to CEI pattern
Severity	 INFORMATIONAL
Description	The contract has numerous sections that do not adhere to the CEI-pattern. We recommend every client to strictly follow the CEI-pattern (https://fravoll.github.io/solidity-patterns/checks_effects_interactions.html) in order to keep their contracts safe from exploits. This issue is only rated as informational because detailed reentrancy exploits will be listed as other issues.
Recommendation	Consider strictly following the CEI pattern.
Resolution	 PARTIALLY RESOLVED

Issue #31	Typographical issue
Severity	 INFORMATIONAL
Description	<u>Lines 239-240</u> <pre>// Check msg.value if (!item.available) {</pre> Wrong comment.
Recommendation	Consider fixing the listed typographical issue.
Resolution	 RESOLVED

2.4 AvalaunchNFTErrors

AvalaunchNFTErrors is a simple library contract which stores all custom errors.

2.4.1 Issues & Recommendations

No issues found.



2.5 BatchReveal

BatchReveal is inherited by the AvalaunchNFT contract and is responsible for shuffling tokenIds. This will be either achieved by using Chainlink VRF or pseudo randomness. A privileged caller within the AvalaunchNFT contract can call reveal which then calls the internal `_reveal` function within this contract in order to reveal the next batch. It is only possible to reveal a batch if this batch was already minted.

During each reveal iteration, a random number is created which then is used to set a seed between 0 and leftover - 1 (the remaining number of NFTs) for the respective batch number.

The `getShuffledTokenId` then returns the shuffled tokenId for each corresponding ID, whereas initially unshuffled, each tokenId returns the original tokenId.



2.5.1 Issues & Recommendations

Issue #32	Possible DoS in case of unsuccessful callback
Severity	 MEDIUM SEVERITY
Description	<code>_getAvailableBatch</code> executes the following check: <pre>if (_totalSupply < lastTokenRevealed + revealBatchSize vrfRequested[_batchNumber]) { revert BatchReveal__NoBatchAvailable(); }</pre> The issue is in the highlighted check — if the Chainlink callback function is unsuccessful, <code>lastTokenRevealed</code> does not increase, essentially preventing the <code>_batchNumber</code> to be increased and resulting in a revert for this check for all future reveals.
Recommendation	Consider implementing logic to disable the VRF method and switching to pseudo-randomness in an effort to keep the reveal process progressing.
Resolution	 RESOLVED The team implemented a function within the <code>AvalaunchNFT</code> contract to disable the VRF logic.

Issue #33**_buildJumps might run out of gas****Severity** LOW SEVERITY**Description**

The _buildJumps function loops over all batches until lastBatch and then calls _getFreeTokenId which again loops over up to the whole rangeLength.

Depending on the rangeLength, this might result in a DoS due to exceeding the block gas limit.

This issue is only assessed as low because there are no expensive operations during these loops and the block gas limit is 15M on Avalanche (<https://github.com/ava-labs/coreth/commit/5ca42a440e8cc4fde3127af4bada7bb32c2e1ec7#diff-bbd41dae8298b4173fc48c404645d2fc29ad98b0419f8f0d30b0b7658a593de8>)

We also refer to the comment on the original tubby-cats implementation:

Total amount of batches (calculated as $TOKEN_LIMIT / REVEAL_BATCH_SIZE$) should be equal or lower than 117, otherwise you might run into the gas limit

Recommendation

Since the algorithm is highly complex and the block gas limit is very high on Avalanche, we do not see the necessity for any changes, however, we think it is important to keep an eye on that and eventually execute some tests before NFTs with an ultra high collectionSize / range / batches are deployed.

Resolution ACKNOWLEDGED

Issue #34	Wrong storage __gap size
Severity	 LOW SEVERITY
Description	BatchReveal is an upgradeable contract and therefore allocates a certain amount of empty slots using a uint256[] empty array of a fixed length. A flaw is that the contract occupies a total of 10 slots while the __gap array length is set to 49 (50 reserved slots - assuming 1 used slot). This would not break any functionality as the contract has not yet been deployed, but in a future upgrade if the contract storage is modified and the storage gap size is again not handled correctly, it might lead to overwriting storage variables that are already in use.
Recommendation	Change the __gap array length to 40.
Resolution	 RESOLVED





PALADIN
BLOCKCHAIN SECURITY