CERTIK

Security Assessment

# NFTCall

CertiK Verified on Mar 16th, 2023

CertiK Verified on Mar 16th, 2023

# NFTCall

The security assessment was prepared by CertiK, the leader in Web3.0 security.

## Executive Summary

| TYPES | ECOSYSTEM | METHODS |
|---|---|---|
| NFT, Option | Ethereum | Manual Review, Static Analysis |
| **LANGUAGE** | **TIMELINE** | **KEY COMPONENTS** |
| Solidity | Delivered on 03/16/2023 | N/A |

CODEBASE

https://github.com/NFTCall-xyz/nftcall-core

...View All

COMMITS

base: 28bd8a20f1364629bdaeb75cd7b7fd1bc926c42c

update1: ec7e6daa3e837e5d6e2403c843dba747b69fbaab

update2: 7dab503916b376a346b1a82cfb391fd19626f8e4

...View All

## Vulnerability Summary

| 18 | 18 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|
| Total Findings | Resolved | Mitigated | Partially Resolved | Acknowledged | Declined | Unresolved |

| | | | |
|---|---|---|---|
| 0 | Critical | | Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks. |
| 0 | Major | | Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project. |
| 1 | Medium | 1 Resolved | Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform. |
| 6 | Minor | 6 Resolved | Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions. |
| 11 | Informational | 11 Resolved | Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code. |

# TABLE OF CONTENTS | NFTCALL

# CODEBASE | NFTCALL

## Repository

https://github.com/NFTCall-xyz/nftcall-core

## Commit

base: 28bd8a20f1364629bdaeb75cd7b7fd1bc926c42c
update1: ec7e6daa3e837e5d6e2403c843dba747b69fbaab
update2: 7dab503916b376a346b1a82cfb391fd19626f8e4
update3: 0883956c96aaeee4d4a89de1e6fcb028972f3230
update4: febd1cbe5741aaeb7b576e683b4b7f1490590d44

# AUDIT SCOPE | NFTCALL

26 files audited ● 1 file with Acknowledged findings ● 10 files with Resolved findings ● 15 files without findings

| ID | Repo | Commit | File | SHA256 Checksum |
|---|---|---|---|---|
| ENF | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/Errors.sol | 32b3d30c0533ff6acfadd2e283d69e3223e9402dc2a3223cf41a1329ba881419 |
| ICE | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/pool/ICallPoolEvents.sol | b535de5dad0df11a64ffe6105f0969f6e8fe4a53f5c7be4872a263cc9951d8ed |
| IPG | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/IPriceOracleGetter.sol | ac38c5abda19ee6217c8a67f329698322ab768a2413f7c6433ee4876dd0a0777 |
| CFN | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/CallFactory.sol | 47cd8a6126fe2947e2445a39b5479db55d393e7b7ece915a6c2ca835ed3e2b99 |
| CPN | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/CallPool.sol | 9da275fc251e3d41d665e9d0377d7c4f10ba03cc51b06e2082b94db4c2ad3f7c |
| CPD | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/CallPoolDeployer.sol | 8eb6fa1142d6f359f3bfaa45c444e17340e69f95762493af14a3938bfe477f75 |
| CTN | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/CallToken.sol | c51db0c41e51c43bbb1224ad2ab4274ef33156a039f00e274f9805e36547b866 |
| DTN | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/DataTypes.sol | 60e919166f5af623b655db2a90827c9a9479f7ab1d6987622d1a9e42e947c9a6 |
| NTN | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/NToken.sol | f809b0e197f941b28de2f0583ef9b50781e8c592d1c6474c85d7921db37c0659 |
| NDC | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/NoDelegateCall.sol | a46fd51c77dff6d116cd4f90d66ccac74fe334f6bbc5659b5a6dd874d52de2cf |
| PNF | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/Premium.sol | 81ea9e880251aea80f208ee495c0345fe2db4080e9fbc181dc36b482830f2009 |
| ICA | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/pool/ICallPoolActions.sol | 911dfbf71304d0be9dbe7dad9d1070c4eac8b75976d2becc5cfe6c287eca14fd |
| ICS | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/pool/ICallPoolDerivedState.sol | cdcdd7afc159448694fc61b15c8469db4407213cdddac91eae2b6887b97ce28f |

| ID | Repo | Commit | File | SHA256 Checksum |
|---|---|---|---|---|
| ICI | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/pool/ICallPoolImmutables.sol | efeecee24d0296bc9f3e8973993c4d92cb5d0b7b0dd0b3e13631573bb065828a |
| ICO | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/pool/ICallPoolOwnerActions.sol | 19b00797874de53f9f4b11e777a870c13c76edc0db99593f844d993c24b60f84 |
| ICN | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/pool/ICallPoolState.sol | 7e93426856672a7050e253ad39b515d690567296ed110ded93ca2f67f9e43b91 |
| ICF | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/ICallFactory.sol | 9afef3237ae5e0e15f37b1f4da8564e28a88ef9aeac9d11c399cf22991392f14 |
| ICP | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/ICallPool.sol | 61a55536d7db09317500ef8101936f9de0237d506aa5df8851cb0b3d78e2f821 |
| ICD | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/ICallPoolDeployer.sol | 680739ea1e49f16cda3f1a5b2d16a9e7c8b402b74ce2e46604a06e32741c658e |
| ICT | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/ICallToken.sol | d3c02300c5d5950051ab20001309e580cbbc1ec55e1069d6db32ac169a19c3d0 |
| INT | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/INToken.sol | 6290af92fc917f8c023a607278c2da584eb930460eee8b3030b2c3de90963ee3 |
| IPN | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/IPremium.sol | 2a91a47fc018285ee108bf9b8ec14db47dc703b1fa35f60c3f47aa1825f909f1 |
| IPO | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/interfaces/IPriceOracle.sol | 10d4bed00babf4e19c1b7ae56d69b512dd1385139458bd5672e79548a3573a19 |
| CTF | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/CallTokenFactory.sol | 629de0c7827a40abf16b353d248a58d660636dcfe5a358d5526f0b2351913424 |
| NFS | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/NFTStatus.sol | c8377adade6848783ba4ba52f928dc5ffa6b105f6706815f34f499e36e5534eb |
| NTF | NFTCall-xyz/nftcall-core | 28bd8a2 | contracts/NTokenFactory.sol | e23860af9493ec885361460dc4702bb954ffa43e2bcf81f093e8ccf555c6a02b |

# APPROACH & METHODS | NFTCALL

This report has been prepared for NFTCall to discover issues and vulnerabilities in the source code of the NFTCall project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

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.

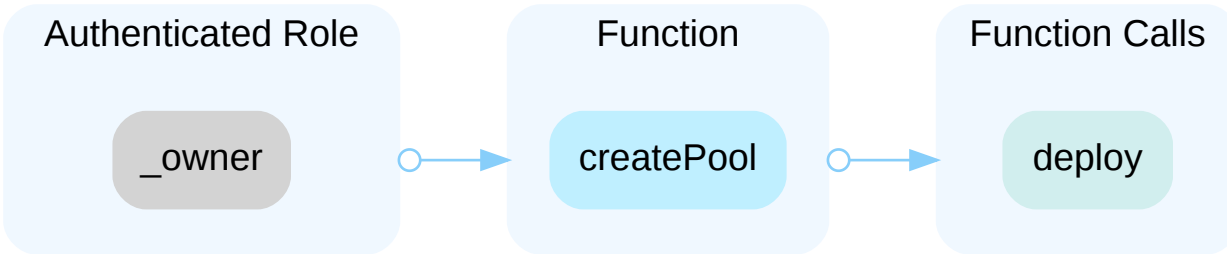
The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# DECENTRALIZATION EFFORTS | NFTCALL

## Description

In the contract `CallFactory`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and create a pool for a malicious ERC721 token with a malicious oracle and/or a malicious premium.




In addition, in the contract `CallPool`, the factory owner mentioned above also has authority over the following functions:

- pause();
- unpause();
- activate();
- deactivate();
- collectProtocol();
- transferERC721();

Any compromise to the factory owner may allow the hacker to take advantage of this authority and do the following:

- pause/unpause the protocol, which disables/enables the use of any function;
- activate/deactivate the protocol, which enables/disables the ability to deposit NFTs, re-list NFTs, open calls, exercise calls, and change preferences;
- collect the accumulated premiums for reserve for themselves;
- transfer any ERC721 tokens accidentally sent to the `NToken` contract that are not part of an open call to a wallet they control.

## Recommendations

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

### Short Term

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

### Long Term

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

### Permanent

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

# FINDINGS | NFTCALL

| 18 | 0 | 0 | 1 | 6 | 11 |
|---|---|---|---|---|---|
| Total Findings | Critical | Major | Medium | Minor | Informational |

This report has been prepared to discover issues and vulnerabilities for NFTCall. Through this audit, we have uncovered 18 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| CPN-02 | Able To Change Preference During Exercise Window | Logical Issue | Medium | Resolved |
| CFN-01 | `createPool()` Will Always Return Zero Address | Logical Issue | Minor | Resolved |
| CPN-01 | Cannot Open Call If `premiumToOwner` Equals `minimumPremiumToOwner` | Logical Issue | Minor | Resolved |
| CPN-08 | Missing Zero Address Validation | Volatile Code | Minor | Resolved |
| NFT-02 | Third-Party Dependencies | Volatile Code | Minor | Resolved |
| NFT-03 | Potential Reentrancy (Out-Of-Order Events) | Volatile Code | Minor | Resolved |
| NTN-01 | NFTs Can Be Locked | Logical Issue | Minor | Resolved |
| CFN-03 | Missing Emit Events | Coding Style | Informational | Resolved |
| CPF-01 | Possible Reentrancy | Logical Issue | Informational | Resolved |
| CPN-05 | Usage Of Magic Numbers | Coding Style | Informational | Resolved |
| CPN-06 | Ambiguous Or Missing Emitted Error Code | Coding Style | Informational | Resolved |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| CPN-07 | Oracle Must Return Price With 18 Decimals | Logical Issue | Informational | Resolved |
| ICE-01 | Unused Event | Coding Style | Informational | Resolved |
| IPG-01 | Unused Interface | Coding Style | Informational | Resolved |
| NFC-01 | Missing Error Messages | Coding Style | Informational | Resolved |
| NFT-01 | Typos | Inconsistency, Coding Style | Informational | Resolved |
| NFT-04 | `tokenURI` Returns NFT URI | Coding Style | Informational | Resolved |
| PNF-01 | Out Of Scope Dependencies | Volatile Code | Informational | Resolved |

# CPN-02 | ABLE TO CHANGE PREFERENCE DURING EXERCISE WINDOW

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Medium | contracts/CallPool.sol (base): 462 | Resolved |

## Description

The exercise window of an option includes the `endTime`. The function `changePreference()` should only be able to be called after an option has ended, however, it makes the following check:

```
462  require(block.timestamp >= uint256(nftStatus[tokenId].getEndTime()),
Errors.CP_NFT_ON_MARKET_OR_UNABAILABLE);
```

This check allows the preferences to be changed if the `block.timestamp` equals the `endTime`. This allows the preferences to be changed and the call to be exercised in the same block. This scenario can cause confusion and have the user calling `changePreference()` spend gas unnecessarily.

## Scenario

The following scenario can occur, allowing a user to change the preferences before a call is executed in the same block:

1. Bob calls `deposit()` on their NFT;
2. Alice opens a call option for a duration of 14 days;
3. After exactly 14 days, Bob calls `changePreference()` with new parameters;
4. In the same block, Alice calls `exerciseCall()`.

## Recommendation

We recommend checking that the `block.timestamp` is greater than the `endTime` to ensure that the preferences cannot be changed during a calls exercise window.

## Alleviation

`[CertiK]`: The client made the recommended changes in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# CFN-01 | `createPool()` WILL ALWAYS RETURN ZERO ADDRESS

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | contracts/CallFactory.sol (base): 21, 26 | Resolved |

## Description

In the function `createPool()`, the return variable `pool` is never assigned as there is a new temorary variable `pool` created in the following line of code:

```
26  address pool = deploy(address(this), erc721token, oracle, premium);
```

## Recommendation

We recommend removing `address` from this line of code to use the return variable `pool`.

## Alleviation

`[CertiK]`: The client made the recommended changes in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# CPN-01 | CANNOT OPEN CALL IF `premiumToOwner` EQUALS `minimumPremiumToOwner`

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | contracts/CallPool.sol (base): 347 | Resolved |

## Description

The value of `minimumPremiumToOwner` should be the smallest premium that will be accepted, however, in the function `_previewOpenCall()` this will revert if the calculated `premiumToOwner` is equal to the `minimumPremiumToOwner`.

## Recommendation

We recommend making the inequality strict to allow calls to be opened when the calculated `premiumToOwner` equals the `minimumPremiumToOwner`.

## Alleviation

`[CertiK]` : The client made the recommended changes in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# CPN-08 | MISSING ZERO ADDRESS VALIDATION

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Minor | contracts/CallPool.sol (base): 401 | Resolved |

## Description

The function `collectProtocol()` should check if the `recipient` is the zero address.

## Recommendation

We recommend adding a check that the passed-in address is not `address(0)` to prevent Ether being accidentally sent to the zero address.

## Alleviation

`[CertiK]` : The client made the recommended changes in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# NFT-02 | THIRD-PARTY DEPENDENCIES

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Minor | contracts/CallFactory.sol (base): 18, 19; contracts/CallPool.sol (base): 23 | Resolved |

## Description

The contract is serving as the underlying entity to interact with third-party `Oracles` and `NFT's` The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets.

## Recommendation

We recommend that the project team constantly monitor the functionality of all `Oracles` and `NFT's`, to mitigate any side effects that may occur when unexpected changes are introduced.

## Alleviation

`[CertiK]`: The client provided the following quote regarding how they will handle the third parties:

`[NFTCall]`:

1. To ensure the reliability of our platform, we will use trusted oracles. For example, we have selected BendDao's oracle for blue chip NFTs due to its high TVL, which is a strong indicator of trustworthiness.
2. We will utilize and monitor multiple oracles to minimize the risk of relying on a single source. If any issues arise with the current oracle, we will quickly switch to another to maintain the integrity of our platform.
3. If an oracle problem arises, it may affect the strike price and premium of a position, but the loss of premium can be disregarded. To prevent openers from taking advantage of such a problem, we can suspend the pool. We have at least 36 hours until exercise time, providing ample time to suspend the pool and address any issues.

# NFT-03 | POTENTIAL REENTRANCY (OUT-OF-ORDER EVENTS)

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Minor | contracts/CallPool.sol (base): 148, 191, 192, 396; contracts/CallToken.sol (base): 199, 204; contracts/NToken.sol (base): 40, 45 | Resolved |

## Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker manipulates the following contract by re-entering, they could potentially throw off any protocol listening for these events. Additionally, `exerciseCall()` does not conform to check effect interaction patterns by burning before all state changes have been completed.

**`withdrawETH()` can send ETH to an external contract which can trigger a re-entry on fallback which can cause events to be out of order:**

```
        _safeTransferETH(to, amount);
        emit WithdrawETH(_msgSender(), to, amount);
        emit BalanceChangedETH(user, _balanceOf[user]);
```

**`withdraw()` and `exerciseCall()` externally calls `Ntoken.burn()` which can cause events to be out of order:**

```
function burn(address user, address receiverOfUnderlying, uint256 tokenId)
public override onlyOwner{
  _burn(tokenId);
  IERC721(nft).safeTransferFrom(address(this), receiverOfUnderlying, tokenId);
  emit Burn(user, receiverOfUnderlying, tokenId);
}
```

**`deposit()` and `depositWithPreference()` externally calls `Ntoken.mint()` which can cause events to be out of order:**

```
function mint(address user, uint256 tokenId) public override onlyOwner{
  _safeMint(user, tokenId);
  emit Mint(user, tokenId);
    }
```

## Scenario

### `withdrawETH()`

1. Bob opens a call option with a higher than intended msg.value.
2. Bob re-enters `withdrawETH()` through a fallback function.
3. The emitted events at the end will be out of order.

### `withdraw()` and `exerciseCall()`

1. Bob has two call options ready to expire.
2. Bob calls to execute one of the options.
3. During the burn of nToken, the `safeTransfer()` calls back to his contract which triggers `onReceived`.
4. This allows re-entrance where another option can be executed.
5. This would put the emitted events out of order.

### `deposit()` and `depositWithPreference()`

1. Bob wants to open two call options.
2. Bob calls to open one of the options.
3. During the mint of nToken, the `_safeMint()` calls back to his contract which triggers `onReceived`.
4. This allows re-entrance where another option can be executed.
5. This would put the emitted events out of order.

## Recommendation

We recommend applying OpenZeppelin's ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent any potential issues from re-entrancy.

## Alleviation

`[CertiK]`: The client made the recommended changes and added `nonReentrant` modifiers to the functions in commits:

- ec7e6daa3e837e5d6e2403c843dba747b69fbaab;
- 16f30a772ee53e8e5080d9a604d994d6e8bf3bec;
- febd1cbe5741aaeb7b576e683b4b7f1490590d44.

# NTN-01 | NFTS CAN BE LOCKED

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | contracts/NToken.sol (base): 51~58 | Resolved |

## Description

If NFTs are accidentally sent directly to the `NToken` contract, then they will become locked and unable to be recovered. The contract implements the `onERC721Received()` function, so even if `safeTransferFrom()` is used a user will be able to send NFTs directly to the contract.

## Recommendation

We recommend adding functionality for the factory owner to return tokens that are accidentally sent directly to the `NToken` contract or providing clear documentation to your users explaining that any NFTs sent directly to the `NToken` contract will be locked forever.

## Alleviation

`[CertiK]` : The client added the function `transferERC721()` which can recover tokens that are accidentally sent to the contract in the following commits:

- ec7e6daa3e837e5d6e2403c843dba747b69fbaab;
- 7dab503916b376a346b1a82cfb391fd19626f8e4.;
- 0883956c96aaeee4d4a89de1e6fcb028972f3230.

# CFN-03 | MISSING EMIT EVENTS

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | Informational | contracts/CallFactory.sol (base): 29 | Resolved |

## Description

When `createPool()` is called in the `CallFactory` contract it creates a new pool, but in addition creates new `NToken` and `CallToken` contracts. There is never an event emitted for the newly created `NToken` or `CallToken` contracts.

## Recommendation

We recommend emitting the addresses of the newly created `NToken` and `CallToken` contracts in the `PoolCreated` event.

## Alleviation

`[CertiK]` : The client made the recommended changes in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# CPF-01 | POSSIBLE REENTRANCY

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Informational | contracts/CallPool.sol (batchOperations): 127, 147, 200, 490 | Resolved |

## Description

In the batch functions that mint or transfer ERC721 tokens, the `onERC721Received()` hook can be used for reentrancy.

## Recommendation

Considering the amount of possible ways a function can be re-entered due to the batch functions calling the `onERC721Received()` hook multiple times, we recommend adding a lock or applying OpenZeppelin's ReentrancyGuard library - `nonReentrant` modifier for any function that calls the `onERC721Received()` hook.

## Alleviation

`[CertiK]` : The client added the `nonReentrant` modifier to the functions in the following commits:

- 51058a1a47dad7b0116a288824281d776ea8346b;
- 16f30a772ee53e8e5080d9a604d994d6e8bf3bec;
- febd1cbe5741aaeb7b576e683b4b7f1490590d44.

# CPN-05 | USAGE OF MAGIC NUMBERS

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | Informational | contracts/CallPool.sol (base): 37, 38 | Resolved |

## Description

The `STRIKE_PRICE_SCALE` and `MAXIMUM_STRIKE_PRICE` are determine based on the decimals of the price returned by the oracle and the decimals of the strike price. The implementation assumes that the decimals of the price returned by the oracle is 18 and the decimals of the strike price is 9.

In addition, the `MAXIMUM_STRIKE_PRICE` is the maximum `uint64` multiplied by the `STRIKE_PRICE_SCALE`.

## Recommendation

We recommend declaring and using constants for the `STRIKE_PRICE_DECIMALS` and `DECIMALS` to improve code maintainability and readability. In addition we recommend adjusting the `MAXIMUM_STRIKE_PRICE` formula to use the `STRIKE_PRICE_SCALE`.

## Alleviation

`[CertiK]`: The client made the recommended changes in commits: ec7e6daa3e837e5d6e2403c843dba747b69fbaab and f1cb2c859d032e8986ebe309268a05782de846bb.

# CPN-06 | AMBIGUOUS OR MISSING EMITTED ERROR CODE

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | Informational | contracts/CallPool.sol (base): 230 | Resolved |

## Description

Inside of the function `openCall()` the following line is returned but is not emitted.

```
vars.errorCode
```

This can cause confusion if the error code is nonzero.

- In `openCall()` this will revert with the error message `Errors.CP_CAN_NOT_OPEN_CALL`. This can make it hard to determine the source of the revert.

## Scenario

1. Bob calls `deposit()` on an NFT.
2. Bob calls `openCall()` on his own NFT.
3. Bob receives the error `CP_CAN_NOT_OPEN_CALL` instead of `CP_CAN_NOT_OPEN_A_POSITION_ON_SELF_OWNED_NFT`.

## Recommendation

We recommend adding an emitted event for these situations to provide more information to users as to why their call was reverted.

## Alleviation

`[CertiK]`: The client changed the code to emit a more descriptive error code in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# CPN-07 | ORACLE MUST RETURN PRICE WITH 18 DECIMALS

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Informational | contracts/CallPool.sol (base): 351, 372, 382, 385 | Resolved |

## Description

The `strikePrice` stored when opening a call is calculated using the oracles returned price. If this does not have 18 decimals, then in `exerciseCall` the `strikePrice` that is compared against the `msg.value` will not have 18 decimals.

## Recommendation

We recommend ensuring the oracle used will return the price using 18 decimals.

## Alleviation

`[CertiK]` : The client stated that the oracle is designed to return the price using 18 decimals.

# ICE-01 | UNUSED EVENT

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | Informational | contracts/interfaces/pool/ICallPoolEvents.sol (base): 7, 8, 18 | Resolved |

## Description

```
7     event Activate(address account);
```

- `Activate` is declared in `ICallPoolEvents` but never emitted.

```
8     event Deactivate(address account);
```

- `Deactivate` is declared in `ICallPoolEvents` but never emitted.

```
18     event DepositETH(address indexed user, address indexed receiver, uint256
amount);
```

- `DepositETH` is declared in `ICallPoolEvents` but never emitted.

## Recommendation

We recommend emitting the `Activate` and `Deactivate` events and either implementing or removing the `DepositETH` event.

## Alleviation

`[CertiK]` : The client made the recommended changes in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# IPG-01 | UNUSED INTERFACE

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | Informational | contracts/interfaces/IPriceOracleGetter.sol (base): 9 | Resolved |

## Description

```
9 interface IPriceOracleGetter {
```

- `IPriceOracleGetter` is declared but never used.

## Recommendation

We recommend removing or implementing the unused interface.

## Alleviation

`[CertiK]` : The client made the recommended changes in commit: 1bc041671c16d92c49e5b0fdfc92c45d7e7ae1f9.

# NFC-01 | MISSING ERROR MESSAGES

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | Informational | contracts/CallPool.sol (update3): 424; contracts/CallFactory.sol (base): 22, 23, 24, 25; contracts/CallPool.sol (base): 174; contracts/CallPoolDeployer.sol (base): 38, 44; contracts/NoDelegateCall.sol (base): 12 | Resolved |

## Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

---

In addition there is an error message that may be misleading. In `exerciseCall()` there is the following check:

```
require(remainValue == 0, Errors.CP_DID_NOT_SEND_ENOUGH_ETH);
```

A user may send more than the strike price in the `msg.value` and this will revert with an error stating they did not send enough ETH, however they sent too much ETH.

## Recommendation

We recommend adding error messages to the linked **require** statements and changing the error emitted in `exerciseCall()`.

## Alleviation

`[CertiK]`: The client added an error message for require statement in the `CallPool` contract. The client opted to not add error messages for the other require statements as they are only possible during the creation of a new pool, which can only be done by the owner of the `CallFactory`. Considering these errors would not be needed by users of the protocol and would only be useful for the developers, we mark this finding as resolved. The client also changed the error message emitted in `exerciseCall()`. These changes were made in commits: ec7e6daa3e837e5d6e2403c843dba747b69fbaab and dbd0df014253e4119d8aca148e7ee4d7c136a865.

# NFT-01 | TYPOS

| Category | Severity | Location | Status |
|---|---|---|---|
| Inconsistency, Coding Style | Informational | contracts/CallPool.sol (base): 34, 157, 160, 236, 287, 336, 385, 389, 440, 462; contracts/DataTypes.sol (base): 22; contracts/Errors.sol (base): 15, 462 | Resolved |

## Description

In the contract, `CallPool` and `Errors`, the following typos were found:

- `CP_NFT_ON_MARKET_OR_UNABAILABLE` should be corrected to `CP_NFT_ON_MARKET_OR_UNAVAILABLE`;
- `CP_DID_NOT_SEND_ENOUGHT_ETH` should be corrected to `CP_DID_NOT_SEND_ENOUGH_ETH`.

In the contract, `DataTypes`, the following typo was found:

- bit 128-192: minimumStrikePrice should be corrected to bit 128-191: minimumStrikePrice.

In the contract, `CallPool`, the following typos were found:

- In the comments above `STRIKE_PRICE_SCALE`, it should use "greater" as opposed to "great";
- In the function `withdraw()`, the comment `// Burn NToken` should be moved to just before the `NToken` is burned;
- In the function `exerciseCall()`, the comment `// Burn NToken and transfer underlying NFT` should be moved to just before the `NToken` is burned.

## Recommendation

We recommend fixing the typos mentioned above.

## Alleviation

`[CertiK]`: The client made the recommended changes in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# NFT-04 | `tokenURI` RETURNS NFT URI

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | Informational | contracts/CallToken.sol (base): 33~35; contracts/NToken.sol (base): 27~31 | Resolved |

## Description

Both contracts `NToken` and `CallToken` have the function `tokenURI()` return the `tokenURI` of the NFT and not a unique URI to the tokens.

## Recommendation

We recommend sharing the reason behind this design and if there is no design necessity, to change the tokenURI for the `NToken` and `CallToken`.

## Alleviation

`[Certik]`: The client stated that the `NToken` is a wrapper and by design have its `tokenURI` return the `tokenURI` of the original NFT. They made the recommended changes for the `CallToken` in commits: 0883956c96aaeee4d4a89de1e6fcb028972f3230 and fb33b0ecc87985989042438a7a9df5dda108902d.

# PNF-01 | OUT OF SCOPE DEPENDENCIES

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Informational | contracts/Premium.sol (base): 8, 11 | Resolved |

## Description

The contract, `Premium` serves as the underlying entity to interact with the pricing of premiums on call options. However, the scope of the audit assumes the input `_premiumMesh` is functionally correct and treats it as a black box.

## Recommendation

We recommend the team carefully considers any `_premiumMesh` to be implemented and ensures that it provides accurate premiums for any situation.

## Alleviation

`[CertiK]` : The client provided the following quote regarding how they will handle the out of scope dependencies:

`[NFTCall]` : "The `_premiumMesh` is generated using the Black-Scholes model, and will be immutable on chain."

# OPTIMIZATIONS | NFTCALL

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| ENF-01 | Can Use Custom Errors | Gas Optimization | Optimization | Acknowledged |
| NTN-02 | Overriding Functions Are Unnecessary | Gas Optimization | Optimization | Resolved |

# ENF-01 | CAN USE CUSTOM ERRORS

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Optimization | contracts/Errors.sol (base): 1~66 | Acknowledged |

## Description

From Solidity `v0.8.4` , there are more gas-efficient ways to explain to users why an operation failed than through strings. Using custom errors can significantly reduce the size of the deployed bytecode and reduce the gas cost when calls revert.

## Recommendation

We recommend considering the use of custom errors to reduce gas costs. For more information see: https://blog.soliditylang.org/2021/04/21/custom-errors/.

## Alleviation

`[CertiK]` : The client acknowledged the finding, but opted to not make any changes to the current version.

# NTN-02 | OVERRIDING FUNCTIONS ARE UNNECESSARY

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Optimization | contracts/NToken.sol (base): 32~38 | Resolved |

## Description

In the contract `NToken`, the functions `_beforeTokenTransfer()` and `supportsInterface()` override their respective functions from the `ERC721` contract. However, when overriding, they only call super on that function, which will simply call the function from the ERC721 contract.

## Recommendation

We recommend removing these functions.

## Alleviation

`[CertiK]`: The client made the recommended changes in commit: ec7e6daa3e837e5d6e2403c843dba747b69fbaab.

# APPENDIX | NFTCALL

## Finding Categories

| Categories | Description |
|---|---|
| Gas Optimization | Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction. |
| Logical Issue | Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works. |
| Volatile Code | Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability. |
| Coding Style | Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable. |
| Inconsistency | Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function. |

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

# DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

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 CertiK 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. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK'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 CertiK 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.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.