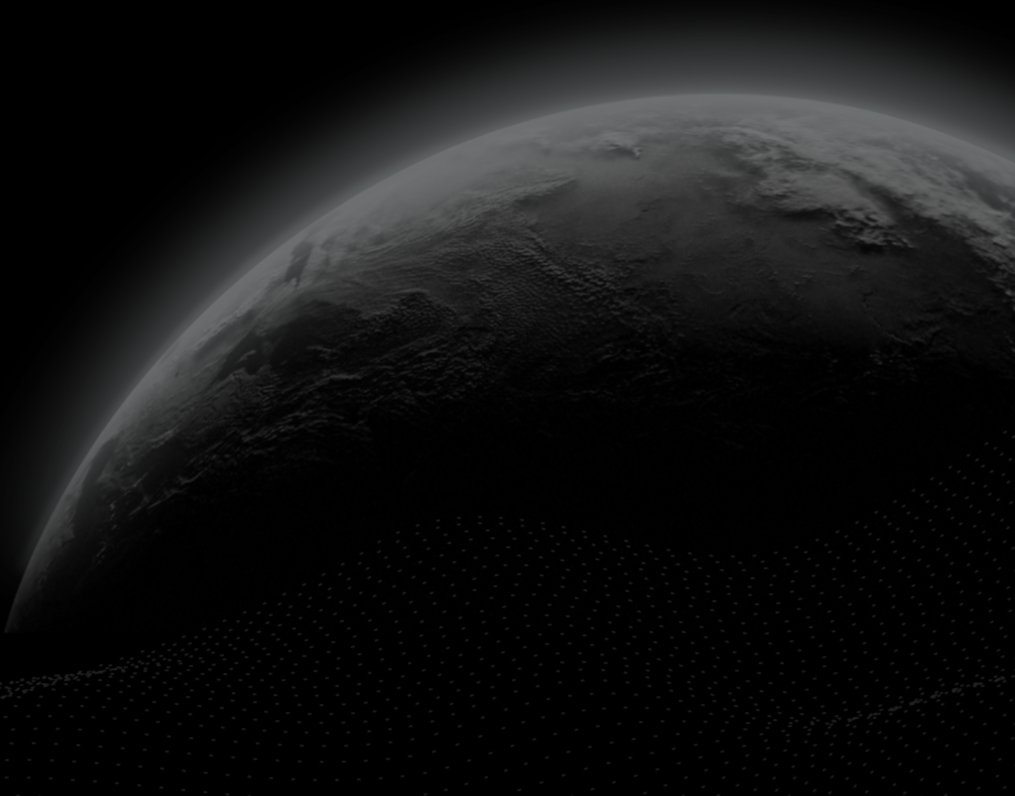
CERTIK

Security Assessment

Double

CertiK Verified on Jan 20th, 2023



CertiK Verified on Jan 20th, 2023

Double

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

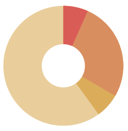
Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Binance Smart Chain (BSC)	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 01/20/2023	N/A

CODEBASE
https://github.com/emojidao/GaslessMarket/tree/1bf449689d4e4df7e17ff4c1b1313c6b018a0c97
...View All

COMMITS
1bf449689d4e4df7e17ff4c1b1313c6b018a0c97
f6cf6aa12c5b2e11c477217e5f086352b4287d81
...View All

Vulnerability Summary



15	12	0	0	3	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥	1	Critical	1 Resolved	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.
🟧	4	Major	2 Resolved, 2 Acknowledged	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.
🟨	9	Minor	8 Resolved, 1 Acknowledged	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.
🟦	0	Informational		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 | DOUBLE

CODEBASE | DOUBLE

Repository

https://github.com/emojidao/GaslessMarket/tree/1bf449689d4e4df7e17ff4c1b1313c6b018a0c97

Commit

1bf449689d4e4df7e17ff4c1b1313c6b018a0c97
f6cf6aa12c5b2e11c477217e5f086352b4287d81

AUDIT SCOPE | DOUBLE

19 files audited ● 7 files with Acknowledged findings ● 3 files with Resolved findings ● 9 files without findings

ID	File	SHA256 Checksum
● BA1	contracts/bank/Bank1155.sol	7739485c8d1bc9fe84eb0bd6306b24815f592de71b6179710ad30171c59b17ee
● BA7	contracts/bank/Bank721.sol	4da0424109ef945a9ac8cbfc2a9f9c3616108a2d7127c494b4548d2ce2c3229b
● BBG	contracts/bank/BaseBank721.sol	8365386eba1675da514ce28248cb1337a8fd7c1aad072526a61895b6a82a551a
● WGM	contracts/bank/W4907Factory.sol	b833b142fa22c280c6043ef9062e6400870efdf061fc84290cb5eb23d057a00f
● FGM	contracts/bank/W5006Factory.sol	89cc6542716c7753f52a1a583f46cc0b2cfaef563af6e6cfff47a8dabb89b0b2
● REN	contracts/market/RentalMarket1155.sol	5d9fe591d6d8775cbc1047938162c7f5ea8095c45a8e88a7a832b4e902a93c5a
● RET	contracts/market/RentalMarket721.sol	aabdf5c430f92903cdc95bd14d3ab08d6f82609f53c51901ab925ef27c549671
● BAB	contracts/bank/Bank.sol	60e7a348f078f9bf4a5eb9058b452437e1426452c9f3738b75c8c71dd8ab3ba1
● BRG	contracts/market/BaseRentalMarket.sol	55e00418de6ff7961ba21217e77b86a26c7cb505b49c9684167ec5b695649c9d
● EIG	contracts/validater/EIP712.sol	c5b3dfc2d2cedfa64b8a0a71f175896342eaa7983f7cdca4cd55bc70ff615707
● IBA	contracts/bank/IBank.sol	70754420c3a50af1e0f34c5b5f5913bad4e0d1d3c76d3bc38a394c5da8596d47
● IBN	contracts/bank/IBank1155.sol	268271068b8f35819dc41f848e724e2aec94bb4543148fd416482ff3176a3a25
● IBK	contracts/bank/IBank721.sol	46aa2f9fa568dfa2dc8680dbbcaca2c270e4971236fd0c4f42befb502bfdab7e
● BSM	contracts/constant/BaseStructs.sol	e33a4132deef7cc07c60a79a63e43b79a975594580432b45e5f3fb3c469e1ad5

ID	File	SHA256 Checksum
● RSM	contracts/constant/RentalStructs.sol	dd96aba3825558c75626de8686d290082e63966a7ffa70804124fc8cc932a981
● TEM	contracts/constant/TokenEnums.sol	bd93d99eb75611cb158f51c0d3e141adc8963c10ec7468c0986ecf3e50609c26
● IMM	contracts/market/IRentalMarket.sol	1cc6555423fda7d11034b48b2332bad0e86f850a9fcaa721fa5c91d461ecff17
● RGM	contracts/market/IRentalMarket1155.sol	d2e32557b1de82017879eb10c48abc72ee66a43b970b997fd2e93d8ff5ff67e5
● MGM	contracts/market/IRentalMarket721.sol	b5e79161061609fd66f31e535fcf702b014778582853df803858a76b6dd5adcb

APPROACH & METHODS | DOUBLE

This report has been prepared for Double to discover issues and vulnerabilities in the source code of the Double 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.

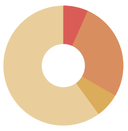
The auditing process pays special attention to the following considerations:

- 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.

FINDINGS | DOUBLE



15	1	4	1	9	0
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Double. Through this audit, we have uncovered 15 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
BA1-01	Irredeemable NFT	Logical Issue	Critical	● Resolved
BA1-02	Lack Of Access Control	Control Flow	Major	● Resolved
BA1-03	Missing Expire Validation	Logical Issue	Major	● Resolved
GMB-01	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	● **Acknowledged**
GMB-02	**Centralized Control Of Contract Upgrade**	**Centralization / Privilege**	**Major**	● **Acknowledged**
GMB-03	Potential Reentrancy Attack	Logical Issue	Medium	● Resolved
BA7-01	Insufficient If-Else Clause	Logical Issue	Minor	● Resolved
BAB-01	Lack Of Storage Gap In Upgradeable Contract	Logical Issue	Minor	● Resolved
FGM-01	Missing Input Validation	Volatile Code	Minor	● Resolved
GMB-04	Missing Zero Address Validation	Volatile Code	Minor	● Resolved

ID	Title	Category	Severity	Status
GMB-05	Unsafe Integer Cast	Logical Issue	Minor	● Resolved
GMB-06	Unprotected Initializer	Coding Style	Minor	● Resolved
GMH-01	Potential Risks Of `deployW4907()`	Logical Issue	Minor	● Resolved
GMH-02	Third Party Dependency	Volatile Code	Minor	● Acknowledged
GMI-01	Unused Return Value	Volatile Code	Minor	● Resolved

BA1-01 | IRREDEEMABLE NFT

Category	Severity	Location	Status
Logical Issue	● Critical	contracts/bank/Bank1155.sol: 76~77, 126	● Resolved

Description

When the token type is ERC5006, the NFT will be locked in the bank.

When creating user records and the token type is ERC5006, the record owner is set as `address(this)` which is the Bank1155 contract.

```
75          if (param.tokenType == TokenType.ERC5006) {
76              recordId = IERC5006(addr5006).createUserRecord(
77                  address(this), // this is record owner
78                  param.user,
79                  param.oNFTId,
80                  uint64(param.oNFTAmount),
81                  uint64(param.expiry)
82              );
83  ...
```

When deleting user records and the token type is ERC5006, the NFT should be sent back to the real owner who stakes it to the bank. But it is transferred from `address(this)` to `record.owner`. The NFT is still in the bank as the `record.owner` is also bank address.

```
123         } else {
124             IERC5006(addr5006).deleteUserRecord(param.recordId);
125             IERC1155(param.oNFT).safeTransferFrom(
126                 address(this),
127                 record.owner, // record.owner is bank
128                 record.tokenId,
129                 record.amount,
130                 ""
131             );
132         }
133 ...
```

Recommendation

We recommend refactoring the code and fixing this issue. We think the correct receiver should be `param.lender`.

Alleviation

[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: 502a0a0439d685efdc33ff52d547f342ccd10d14.

BA1-02 | LACK OF ACCESS CONTROL

Category	Severity	Location	Status
Control Flow	● Major	contracts/bank/Bank1155.sol: 64, 140	● Resolved

Description

The functions `createUserRecord()` and `deleteUserRecords()` can be called by anyone as it has no access restriction. This enables anyone to call the function `deleteUserRecords()` and delete unexpired rental records. Similarly, anyone can call `createUserRecord()` and create rental records without payment if the NFT owner happens to allow the bank to transfer the NFT. Both risks could seriously harm the interests of renter and lender.

Recommendation

We advise the client to add `OnlyMarket` modifier to the functions so that only the market can call the functions.

Alleviation

[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: 3b4aa4644eb6a53e501121d124ab56e595ab5c45.

BA1-03 | MISSING EXPIRE VALIDATION

Category	Severity	Location	Status
Logical Issue	● Major	contracts/bank/Bank1155.sol: 107	● Resolved

Description

The bank contract does not check whether the rental records are indeed expired before deleting them. If unexpired records are deleted, the interests of renter are compromised.

Recommendation

We recommend adding a validation to ensure the deleted rental records are indeed expired, as shown below:

```
if (record.expiry > block.timestamp) return;
```

Alleviation

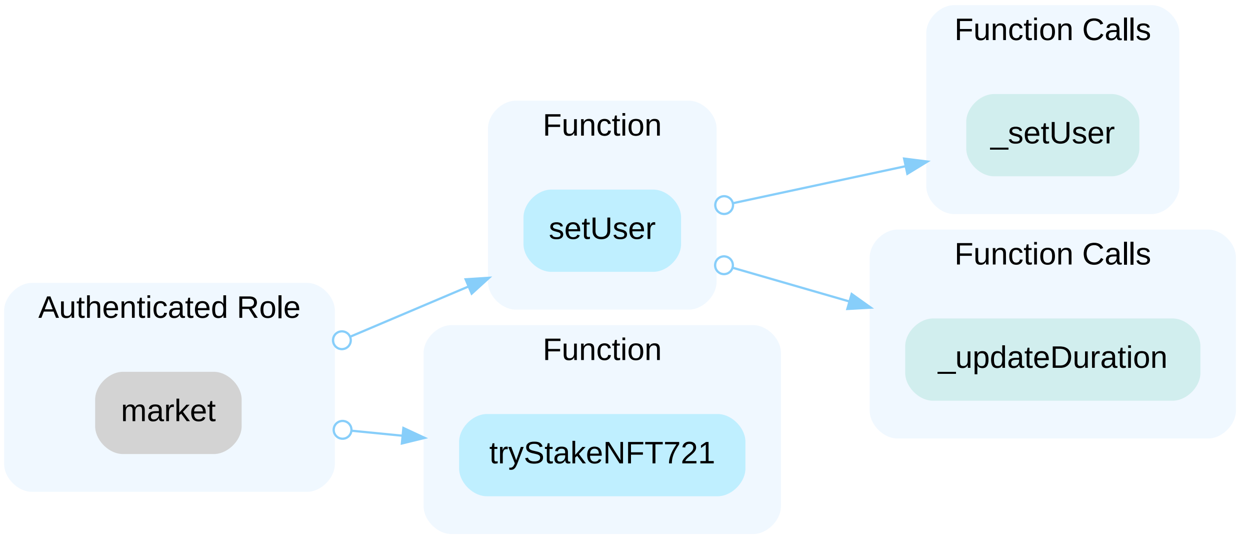
[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: 3b4aa4644eb6a53e501121d124ab56e595ab5c45.

GMB-01 | CENTRALIZATION RELATED RISKS

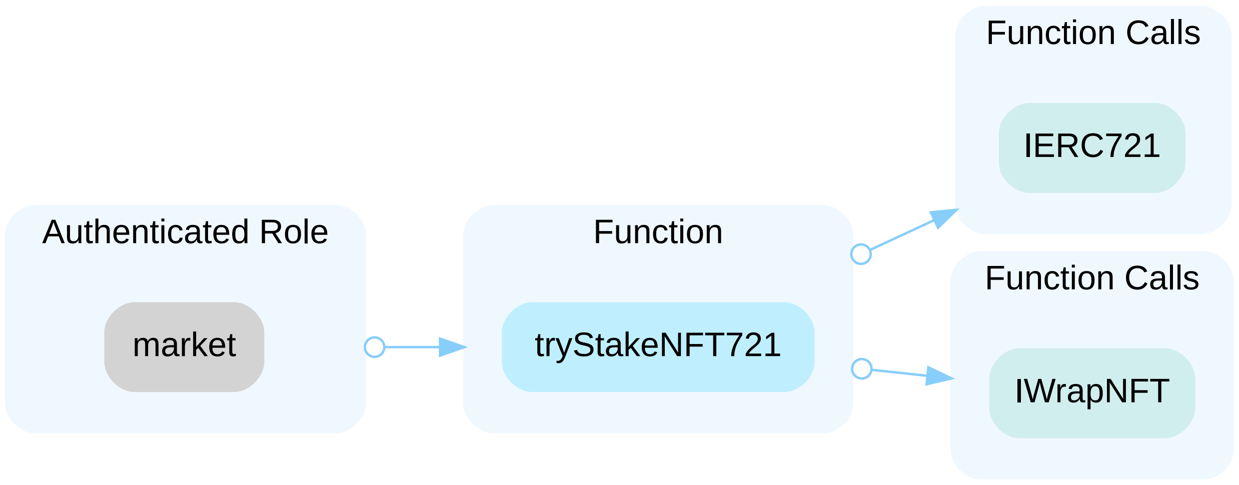
Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/bank/Bank721.sol: 19; contracts/bank/BaseBank721.sol: 15, 28; contracts/bank/W4907Factory.sol: 27, 68; contracts/bank/W5006Factory.sol: 21, 38; contracts/market/RentalMarket1155.sol: 19; contracts/market/RentalMarket721.sol: 20**	● **Acknowledged**

Description

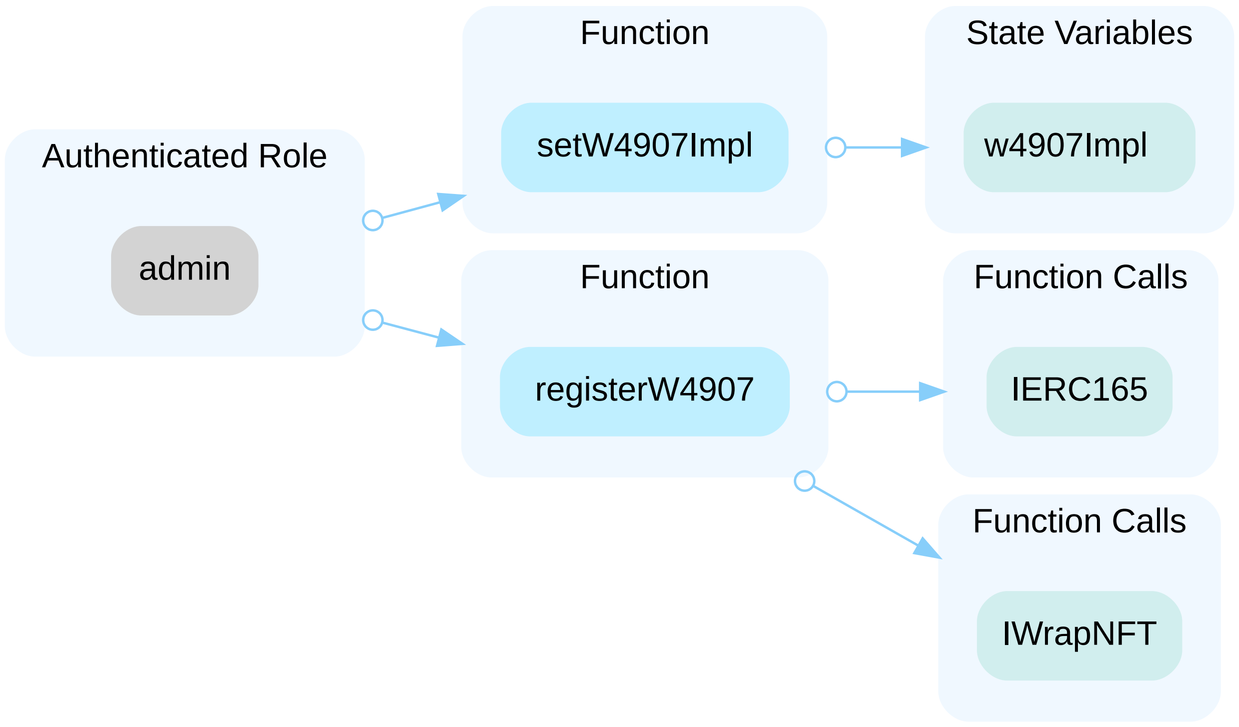
In the contract `BaseBank721` the role `market` has authority over the functions shown in the diagram below. Any compromise to the `market` account may allow the hacker to take advantage of this authority.



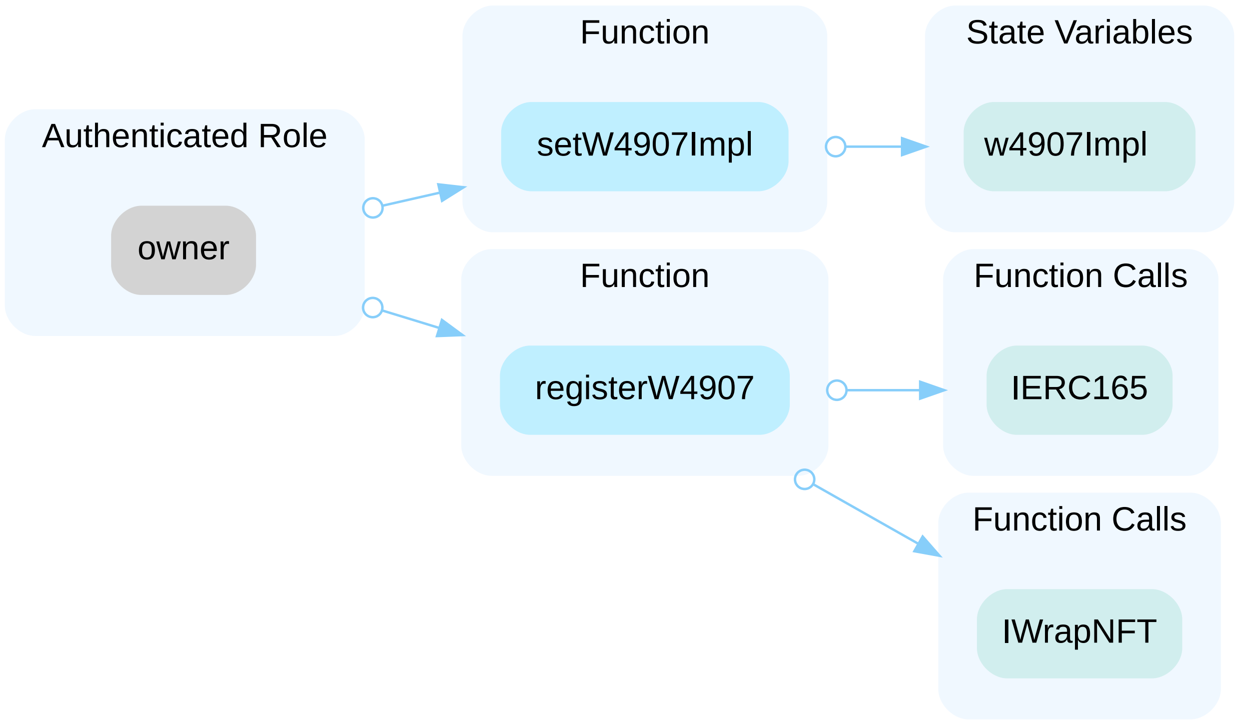
In the contract `Bank721` the role `market` has authority over the functions shown in the diagram below. Any compromise to the `market` account may allow the hacker to take advantage of this authority.



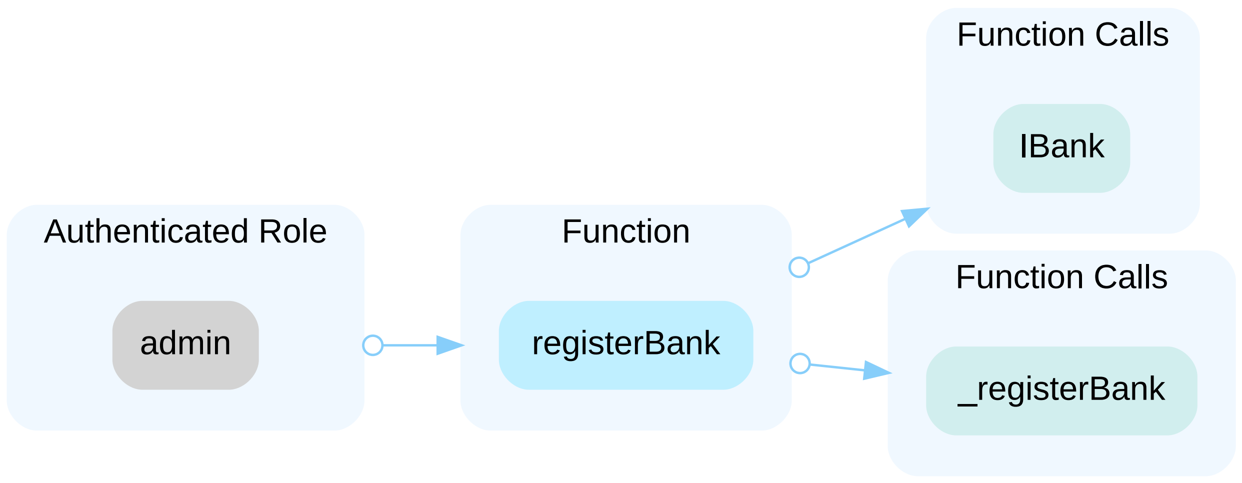
In the contract `W4907Factory` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority.



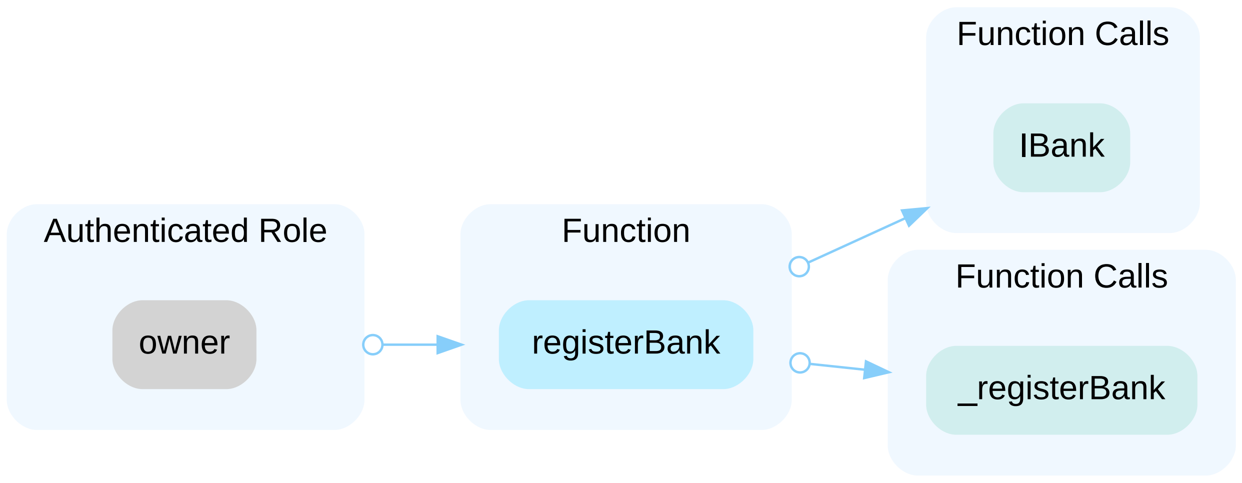
In the contract `W4907Factory` the role `owner` has authority over the functions shown in the diagram below. Any compromise to the `owner` account may allow the hacker to take advantage of this authority.



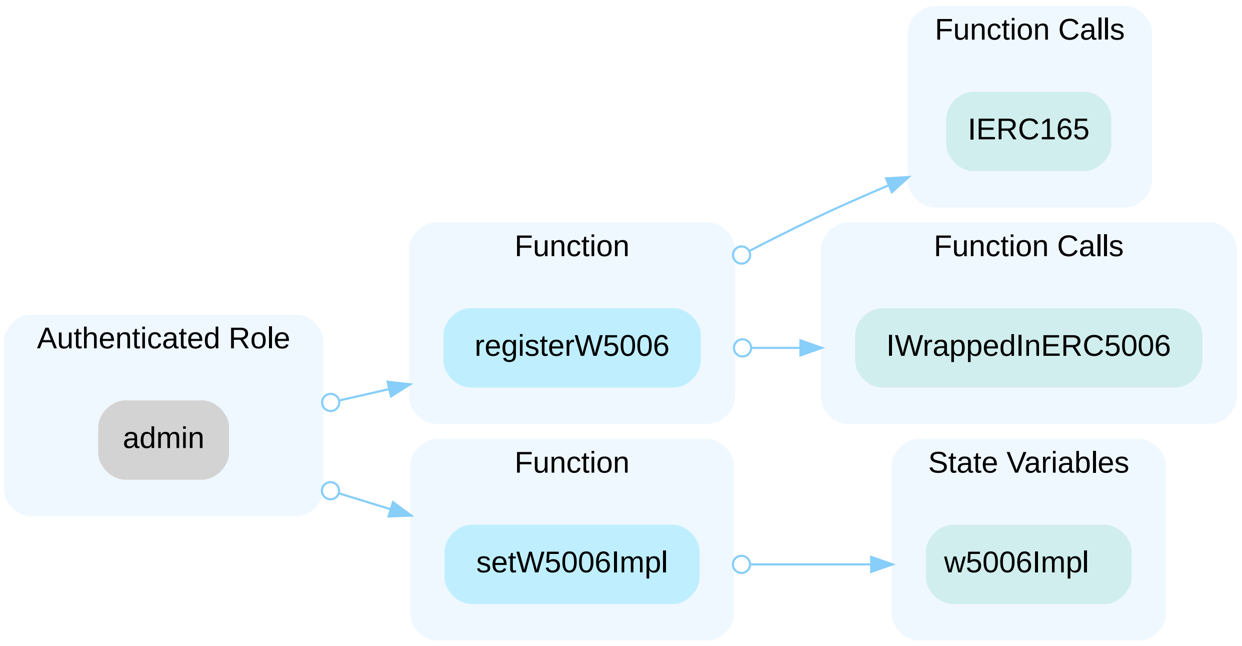
In the contract `RentalMarket721` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority.



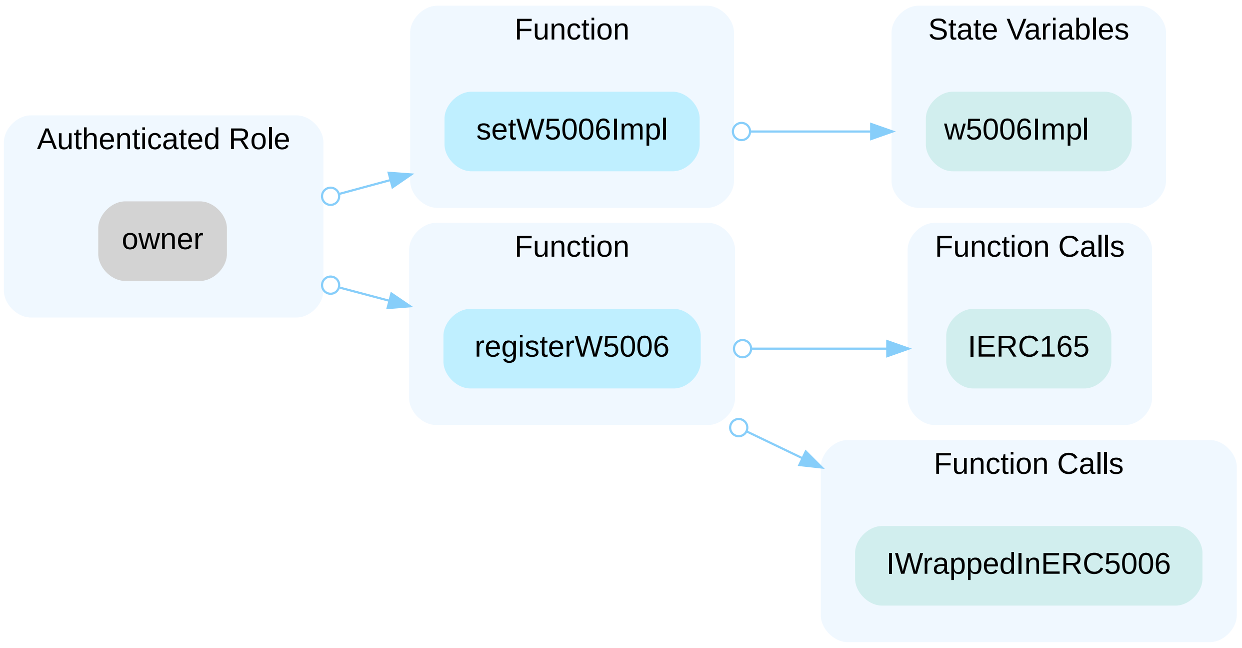
In the contract `RentalMarket721` the role `owner` has authority over the functions shown in the diagram below. Any compromise to the `owner` account may allow the hacker to take advantage of this authority.



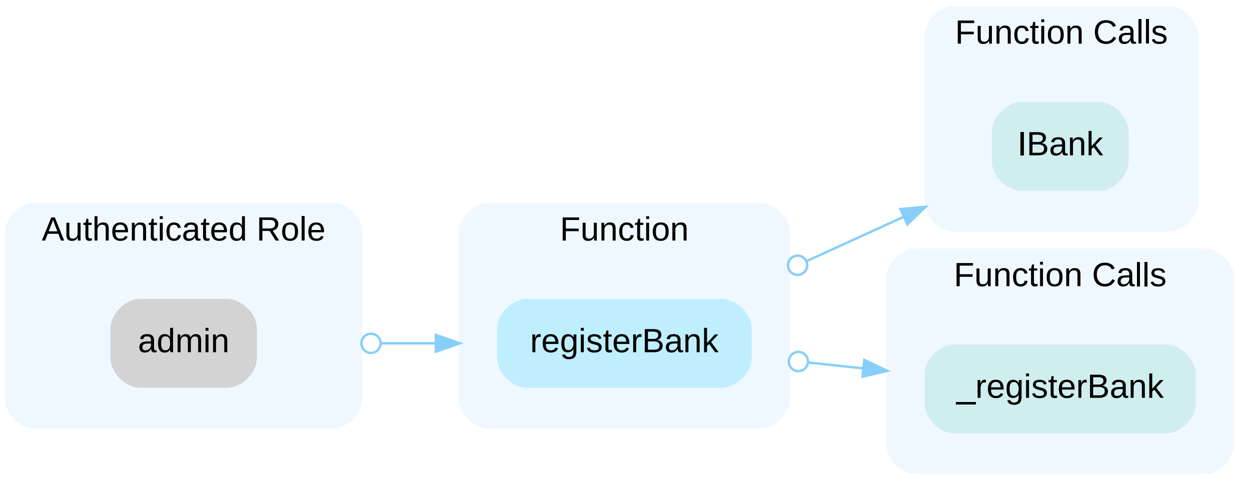
In the contract `W5006Factory` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority.



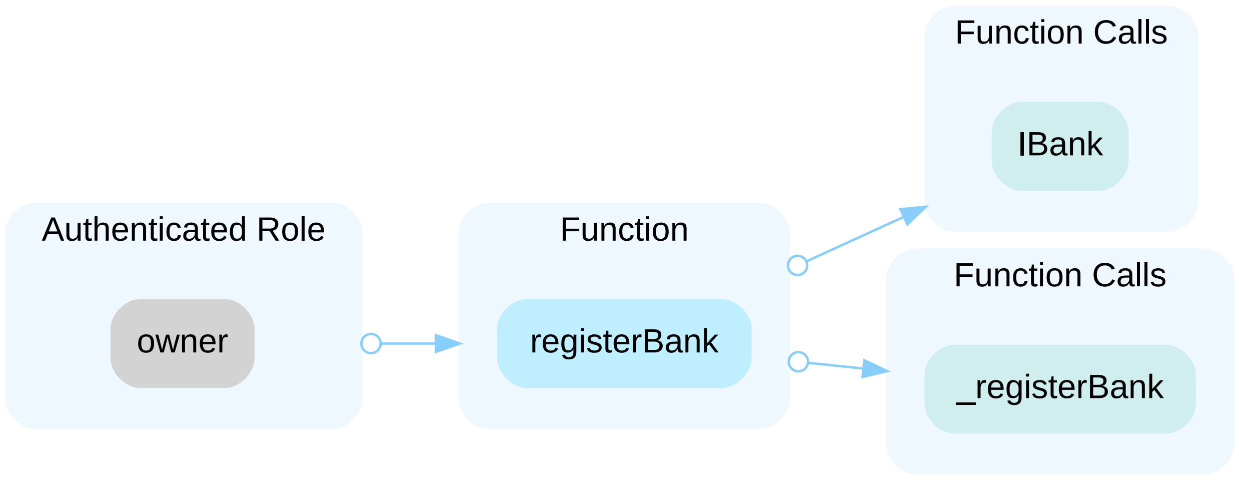
In the contract `W5006Factory` the role `owner` has authority over the functions shown in the diagram below. Any compromise to the `owner` account may allow the hacker to take advantage of this authority.



In the contract `RentalMarket1155` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` account may allow the hacker to take advantage of this authority.



In the contract `RentalMarket1155` 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.



Recommendation

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.

Alleviation

[`Double`]: Issue acknowledged.

1. The ProxyAdmin will be a `TimeLockMultiSign Wallet`.
2. The owner will be a `GnosisSafe Wallet`.
3. The admin of contracts will initially be a personal wallet, which will also be changed to `GnosisSafe Wallet` once all are stable.

GMB-02 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/bank/Bank1155.sol: 16; contracts/bank/Bank721.sol: 10; contracts/market/RentalMarket1155.sol: 10; contracts/market/RentalMarket721.sol: 11**	● **Acknowledged**

Description

The cited contracts are upgradeable contracts, the owner can upgrade the contract without the community's commitment. If an attacker compromises the account, he/she can change the implementation of the contract and may damage the contracts.

Recommendation

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.

Alleviation

[`Double`]: Issue acknowledged. The ProxyAdmin will be a `TimeLockMultiSign Wallet`.

GMB-03 | POTENTIAL REENTRANCY ATTACK

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/bank/Bank1155.sol: 140; contracts/bank/Bank721.sol: 37; contracts/market/RentalMarket1155.sol: 30, 84; contracts/market/RentalMarket721.sol: 29, 78	● 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 can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: f6cf6aa12c5b2e11c477217e5f086352b4287d81.

BA7-01 | INSUFFICIENT IF-ELSE CLAUSE

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/bank/Bank721.sol: 31, 47, 68, 83	● Resolved

Description

The cited if-else clause is not insufficient. If the token type is not ERC721, it may also not be ERC4907.

Recommendation

We recommend adding a real else clause as below:

```
    if (tokenType == TokenType.ERC721) {
        ...
     } else if (tokenType == TokenType.ERC4907) {
        ...
     } else {
        revert("invalid token type");
     }
```

Alleviation

[CertiK]: The team heeded our advice and resolved this finding in the commit hash: 3b4aa4644eb6a53e501121d124ab56e595ab5c45.

BAB-01 | LACK OF STORAGE GAP IN UPGRADEABLE CONTRACT

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/bank/Bank.sol: 7	● Resolved

Description

There is no storage gap preserved in the logic contract. Any logic contract that acts as a base contract that needs to be inherited by other upgradeable child should have a reasonable size of storage gap preserved for the new state variable introduced by the future upgrades.

Recommendation

We recommend having a storage gap of a reasonable size preserved in the logic contract in case that new state variables are introduced in future upgrades. For more information, please refer to: https://docs.openzeppelin.com/contracts/3.x/upgradeable#storage_gaps.

Alleviation

[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: e86b3218ca54152b404ff08e29f686cfb99d5a80.

FGM-01 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/bank/W5006Factory.sol: 25	● Resolved

Description

It should be checked that the parameter `oNFT` is ERC1155 and is not ERC5006.

Recommendation

We recommend adding checks for the parameter `oNFT` to ensure it is ERC1155 and is not ERC5006.

Alleviation

[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: 3b4aa4644eb6a53e501121d124ab56e595ab5c45.

GMB-04 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/bank/Bank.sol: 17; contracts/bank/Bank1155.sol: 21; contracts/bank/Bank721.sol: 15, 45; contracts/bank/W4907Factory.sol: 28; contracts/bank/W5006Factory.sol: 22; contracts/market/BaseRentalMarket.sol: 37	● Resolved

Description

Addresses should be checked before assignment or an external call to make sure they are not zero addresses.

Bank.sol

```
17          market = market_;
```

- `market_` is not zero-checked before being used.

W4907Factory.sol

```
28          w4907Impl = w4907Impl_;
```

- `w4907Impl_` is not zero-checked before being used.

W5506Factory.sol

```
22          w5006Impl = w5006Impl_;
```

- `w5006Impl_` is not zero-checked before being used.

Bank721.sol

```
15           _initOwnable(owner_, admin_);
```

- `owner_` is not zero-checked before being used.

```
        address w4907 = oNFT_w4907[oNFT];
```

- `w4907` is not zero-checked before being used.

Bank1155.sol

```
        _initOwnable(owner_, admin_);
```

- `owner_` is not zero-checked before being used.

BaseRentalMarket.sol

```
        _initOwnable(owner_, admin_);
```

- `owner_` is not zero-checked before being used.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[`CertiK`]: The team heeded our advice and resolved the finding in the commit hash: 3b4aa4644eb6a53e501121d124ab56e595ab5c45.

GMB-05 | UNSAFE INTEGER CAST

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/bank/Bank1155.sol: 80, 81, 86, 88; contracts/bank/Bank721.sol: 67, 69; contracts/bank/BaseBank721.sol: 99; contracts/validater/EIP712.sol: 189	● Resolved

Description

```
80                  uint64(param.oNFTAmount),
```

- The type conversion `uint64(param.oNFTAmount)` from larger type uint256 to smaller type uint64 may truncate data.

```
81                  uint64(param.expiry)
```

- The type conversion `uint64(param.expiry)` from larger type uint256 to smaller type uint64 may truncate data.

```
86                  uint64(param.oNFTAmount),
```

- The type conversion `uint64(param.oNFTAmount)` from larger type uint256 to smaller type uint64 may truncate data.

```
88                  uint64(param.expiry)
```

- The type conversion `uint64(param.expiry)` from larger type uint256 to smaller type uint64 may truncate data.

```
67              IERC4907(w4907).setUser(nft.tokenId, renter, uint64(expiry));
```

- The type conversion `uint64(expiry)` from larger type uint256 to smaller type uint64 may truncate data.

```
69              IERC4907(nft.token).setUser(nft.tokenId, renter, uint64(expiry));
```

- The type conversion `uint64(expiry)` from larger type uint256 to smaller type uint64 may truncate data.

```
        durations[key].start = uint40(start);
```

- The type conversion `uint40(start)` from larger type uint256 to smaller type uint40 may truncate data.

```
            v = uint8(uint256(vs >> 255)) + 27;
```

- The type conversion `uint8(uint256(vs >> 255))` from larger type uint256 to smaller type uint8 may truncate data.

Recommendation

We advise checking the bounds of integer values before casting, so the values will not be truncated or flip the sign. Alternatively, the SafeCast library from OpenZeppelin can be used in place of type casting.

Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/71aaca2d9db4655602137403920 44b2cd3853a3b/contracts/utils/math/SafeCast.sol

Alleviation

[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: 3b4aa4644eb6a53e501121d124ab56e595ab5c45.

GMB-06 | UNPROTECTED INITIALIZER

Category	Severity	Location	Status
Coding Style	● Minor	contracts/bank/Bank1155.sol: 16; contracts/bank/Bank721.sol: 10; contracts/market/RentalMarket1155.sol: 10; contracts/market/RentalMarket721.sol: 11	● Resolved

Description

One or more logic contracts do not protect their initializers. An attacker can call the initializer and assume ownership of the logic contract, whereby she can perform privileged operations that trick unsuspecting users into believing that she is the owner of the upgradeable contract.

```
contract Bank1155 is Bank, W5006Factory, ERC1155Receiver, IBank1155 {
```

- `Bank1155` is an upgradeable contract that does not protect its initializer.

```
    function initialize(
```

- `initialize` is an unprotected initializer function.

```
contract Bank721 is BaseBank721, W4907Factory  {
```

- `Bank721` is an upgradeable contract that does not protect its initializer.

```
    function initialize(
```

- `initialize` is an unprotected initializer function.

```
contract RentalMarket1155 is BaseRentalMarket, IRentalMarket1155 {
```

- `RentalMarket1155` is an upgradeable contract that does not protect its initializer.

```
    function initialize(
```

- `initialize` is an unprotected initializer function.

```
contract RentalMarket721 is BaseRentalMarket, IRentalMarket721 {
```

- `RentalMarket721` is an upgradeable contract that does not protect its initializer.

```
    function initialize(
```

- `initialize` is an unprotected initializer function.

Recommendation

We advise calling `_disableInitializers` in the constructor or giving the constructor the `initializer` modifier to prevent the intializer from being called on the logic contract.

Reference: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract

Alleviation

[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: 3b4aa4644eb6a53e501121d124ab56e595ab5c45.

GMH-01 | POTENTIAL RISKS OF `deployW4907()`

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/bank/W4907Factory.sol: 58; contracts/bank/W5006Factory.sol : 31	● Resolved

Description

According to the design, the function `deployW4907()` has no access restrictions, and anyone can establish a mapping relationship between a compliant ERC721 contract and the `w4907Impl` contract.

However, since the mapping relationship cannot be modified, if an ERC721 contract is mapped to `w4907Impl` contract by someone, the admin cannot establish a mapping relationship between this ERC721 contract and other ERC4907 contracts through the function `registerW4907()` .

This may present potential uncontrolled risks because the user may set incorrect name and symbol data in the deployment of the w4907 contract.

Recommendation

We advise the client to confirm this part of the logic.

Alleviation

[`Double`]: Issue acknowledged. Deploying the `w4907Impl` contract is essentially cloning, and the deployer has no rights to change the token's data. We removed the input of `name` and `symbol` in the commit 6fef4632c605987733c891c186087da1329ea6c7.

GMH-02 | THIRD PARTY DEPENDENCY

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/bank/Bank1155.sol: 64, 105; contracts/bank/Bank721.sol: 21, 39; contracts/bank/W4907Factory.sol: 20, 70; contracts/bank/W5006Factory.sol: 31, 38	● Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assume their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
64      function createUserRecord(RecordParam memory param) external {
```

- The function `Bank1155.createUserRecord` interacts with third party contract with `IERC1155` interface via `param`.

```
105      function _deleteUserRecord(RentingRecord calldata param) internal {
```

- The function `Bank1155._deleteUserRecord` interacts with third party contract with `IERC1155` interface via `param`.

```
21          address oNFT,
```

- The function `Bank721.tryStakeNFT721` interacts with third party contract with `IERC721` interface via `oNFT`.

```
39          address oNFT,
```

- The function `Bank721.redeemNFT721` interacts with third party contract with `IERC721` interface via `oNFT`.

```
20      mapping(address => address) public oNFT_w4907;
```

- The contract `W4907Factory` interacts with third party contract with `IWrapNFT` interface via `oNFT_w4907`.

```
        address w4907
```

- The function `W4907Factory.registerW4907` interacts with third party contract with `IWrapNFT` interface via `w4907`.

```
    function deployW5006(address oNFT) public {
```

- The function `W5006Factory.deployW5006` interacts with third party contract with `IERC1155` interface via `oNFT`.

```
    function registerW5006(address oNFT, address w5006) public onlyAdmin {
```

- The function `W5006Factory.registerW5006` interacts with third party contract with `IWrappedInERC5006` interface via `w5006`.

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[`Double`]: Issue acknowledged.

GMI-01 | UNUSED RETURN VALUE

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/market/BaseRentalMarket.sol: 147; contracts/market/RentalMarket721.sol: 38, 86	● Resolved

Description

According to the design, when the renter or lender fulfills the order, the contract will validate the metadata of the NFTs included in the order via the function `_validateMetadata()` .

BaseRentalMarket.sol

```
function _validateMetadata(NFT calldata nft, Metadata calldata metadata)
        internal
        view
    {
        if (metadata.checker != address(0)) {
            IMetadataChecker(metadata.checker).check(
                nft.token,
                nft.tokenId,
                metadata.metadataHash
            );
        }
    }
```

However, the outcome of the external call is not handled by the function `_validateMetadata()` . The external call is useless if the result is not validated.

Recommendation

We recommend that the function `_validateMetadata()` return the outcome of the external call and verify the result in the `RentalMarket721` contract. **BaseRentalMarket.sol**

```
function _validateMetadata(NFT calldata nft, Metadata calldata metadata)
        internal view returns (bool)
    {
        if (metadata.checker != address(0)) {
            return IMetadataChecker(metadata.checker).check(
                nft.token,
                nft.tokenId,
                metadata.metadataHash
            );
        }
        return true;
    }
```

RentalMarket721.sol

```
  bool success = _validateMetadata(lendOrder.nft, lendOrder.metadata);
  require(success, "nft or metadata error!");
```

Alleviation

[`CertiK`]: The team heeded our advice and resolved this finding in the commit hash: 3b4aa4644eb6a53e501121d124ab56e595ab5c45.

APPENDIX | DOUBLE

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
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.
Control Flow	Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.
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.

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.