Taker Controller

Audit Report

Tue Dec 31 2024

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

Taker Controller Audit Report

1 Executive Summary

1.1 Project Information

Description	Taker Controller is the asset management, exchange management and veTaker management in the taker protocol
Type	DeFi
Auditors	ScaleBit
Timeline	Fri Dec 20 2024 - Tue Dec 31 2024
Languages	Solidity
Platform	Taker
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/takerprotocol/token-controller
Commits	f1e6ebc3f194b0aa5fae269564101859037 02da3 ca03675907dcb5842c003c4b18ef3335594d004a 86ed6c7ec1b42bd38c7871eaca43ff02f83f28d8

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
ECO	contracts/ExchangeController.sol	1a6f5f1ba422f7ab2ac5f66b8c39b4de920b39f1
PAU	contracts/Pausable.sol	797c4e2b42bc55869b4db094d1b7b7e54338f0ae
VET	contracts/VETaker.sol	9a491a86ab998a4a679e8de48bc0d178f84a7fa7
PAD	contracts/PrecompiledAdapter.sol	0e774acdd4fde30ebcc67120f2c09ddf14003a8f
AMA	contracts/AssetsManager.sol	9763c0a812b536a2c222444ec8662e5aa45e8d50

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	3	0
Informational	0	0	0
Minor	2	2	0
Medium	0	0	0
Major	1	1	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Taker Controller to identify any potential issues and vulnerabilities in the source code of the Taker Controller smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 3 issues of varying severity, listed below.

ID	Title	Severity	Status
AMA-1	Lack of Events Emit	Minor	Fixed
AMA-2	Lack of Reentrancy Protection	Minor	Fixed
PAD-1	Lack of Permission Control	Major	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Taker Controller Smart Contract :

Admin

- Admin can set `tToken` , `vToken` , `tGas` , `tStaking` , `tGasRatio` , `isMulRatio` .
- Admin can set `live` through the `toggleLive` function.
- Admin can add or remove `whiteList` .

Minter

- Minter can mint or burn `VETaker` .

User

- User can use `exchangeVToTToken` , `exchangeTToVToken` , `exchangeTgasFromTToken` , `exchangeTgasFromVToken` functions to exchange tokens.
- User can use `claimTToken` to exchange tokens `tToken` to `vToken` .
- User can use `stakingWithNominate` to exchange tokens `vToken` to `tToken` and call `bondAndNominate` remotely through `tStaking` .
- User can use `stakingExtra` to exchange tokens `vToken` to `tToken` and call `bondExtra` remotely through `tStaking` .
- User can use `stakingExtra` to exchange tokens `vToken` to `tToken` and call `bondExtraAndNominate` remotely through `tStaking` .
- User can use `stakingExtra` to exchange tokens `vToken` to `tToken` and call `bondAndValidate` remotely through `tStaking` .
- User can use `stakingExtra` to exchange tokens `vToken` for `tToken` and remotely call `bondExtraAndValidate` through `tStaking` .
- User can use `burnT/mintT/mintTGas` functions to call the `burn/mintTo` interface of `ITToken` and `ITgas` .

4 Findings

AMA-1 Lack of Events Emit

Severity: Minor

Status: Fixed

Code Location:

contracts/AssetsManager.sol#46

Descriptions:

The contract lacks appropriate events for some key functions such as: `setTToken()` , `setVToken()` , `setTgas()` , `setTStaking()` , `setTgasRatio()` . The lack of event records for these functions may cause inconvenience in the subsequent tracking and contract status changes.

Suggestion:

It is recommended to emit events for the functions.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

AMA-2 Lack of Reentrancy Protection

Severity: Minor

Status: Fixed

Code Location:

contracts/AssetsManager.sol

Descriptions:

In the _exchangeVToTToken function, the burn operation is executed before mint, and the burn method may trigger the callback function of the token. If the burn method of _vToken supports the callback mechanism, the attacker can call exchangeVToTToken again in the callback function, thereby repeatedly triggering the mintT operation, causing the system to be abused to mint tokens.

Attack flow as:

1. The attacker calls exchangeVToTToken and provides an initial _amount.
2. In the callback of burn, the attacker calls exchangeVToTToken again, resulting in repeated minting of tokens.
3. The attacker mints more tokens than his actual assets through multiple nested calls.

```
function exchangeVToTToken(uint256 _amount) public onlyWhenLive {
    _exchangeVToTToken(vToken, tToken, _amount);
}

function _exchangeVToTToken(
    address _vToken,
    address _tToken,
    uint256 _amount
) internal {
    require(
        _amount <= IERC20(_vToken).balanceOf(msg.sender),
        "Insufficient VToken balance"
    );
    IVETaker(_vToken).burn(msg.sender, _amount);

    mintT(_tToken, msg.sender, _amount);
    emit ExchangeVToT(msg.sender, _amount);
}
```

Suggestion:

It is recommended to add a decorator to prevent reentrancy and use the official library `nonReentrant`. Since the mock contract is BRC20 and the intended design is BRC20, errors will occur when using tokens with callbacks, such as ERC1155 standard tokens.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

PAD-1 Lack of Permission Control

Severity: Major

Status: Fixed

Code Location:

contracts/PrecompiledAdapter.sol#11

Descriptions:

Functions lack permission control. In the PrecompiledAdapter contract, `burnT()`, `mintT`, and `mintTGas` can be called arbitrarily.

```
function burnT(address _tToken, address _addr, uint256 _amount) public {
    ITToken(_tToken).burn(_addr, _amount);
}

function mintT(address _tToken, address _to, uint256 _amount) public {
    ITToken(_tToken).mintTo(_to, _amount);
}

function mintTGas(address _tGas, address _to, uint256 _amount) public {
    ITgas(_tGas).mintTo(_to, _amount);
}
```

On the other hand, the `token` address of the function can be passed in arbitrarily, which means that the user can arbitrarily manipulate the address called by the PrecompiledAdapter contract.
Finally, the caller is checked. In the node, caller==msg.sender is checked. That is, msg.sender is the adapter contract, which means that the public function does not check the user's permissions.

Suggestion:

It is recommended to add permission control, for example, onlyOwner or onlyNode or change to internal call.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer