# Smart Contract Security Audit Report

# Table Of Contents

**1 Executive Summary**

**2 Audit Methodology**

**3 Project Overview**

3.1 Project Introduction

3.2 Vulnerability Information

**4 Code Overview**

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

**5 Audit Result**

**6 Statement**

# 1 Executive Summary

On 2022.04.18, the SlowMist security team received the Coordinape team's security audit application for Coordinape protocol and coordinape vesting, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

| Test method | Description |
|---|---|
| Black box testing | Conduct security tests from an attacker's perspective externally. |
| Grey box testing | Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses. |
| White box testing | Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc. |

The vulnerability severity level information:

| Level | Description |
|---|---|
| Critical | Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities. |
| High | High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities. |
| Medium | Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities. |
| Low | Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed. |
| Weakness | There are safety risks theoretically, but it is extremely difficult to reproduce in engineering. |

| Level | Description |
|---|---|
| Suggestion | There are better practices for coding or architecture. |

# 2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

| Serial Number | Audit Class | Audit Subclass |
|---|---|---|
| 1 | Overflow Audit | - |
| 2 | Reentrancy Attack Audit | - |
| 3 | Replay Attack Audit | - |
| 4 | Flashloan Attack Audit | - |
| 5 | Race Conditions Audit | Reordering Attack Audit |
| 6 | Permission Vulnerability Audit | Access Control Audit |
|  |  | Excessive Authority Audit |

| Serial Number | Audit Class | Audit Subclass |
|---|---|---|
| 7 | Security Design Audit | External Module Safe Use Audit |
| | | Compiler Version Security Audit |
| | | Hard-coded Address Security Audit |
| | | Fallback Function Safe Use Audit |
| | | Show Coding Security Audit |
| | | Function Return Value Security Audit |
| | | External Call Function Security Audit |
| | | Block data Dependence Security Audit |
| | | tx.origin Authentication Security Audit |
| 8 | Denial of Service Audit | - |
| 9 | Gas Optimization Audit | - |
| 10 | Design Logic Audit | - |
| 11 | Variable Coverage Vulnerability Audit | - |
| 12 | "False Top-up" Vulnerability Audit | - |
| 13 | Scoping and Declarations Audit | - |
| 14 | Malicious Event Log Audit | - |
| 15 | Arithmetic Accuracy Deviation Audit | - |
| 16 | Uninitialized Storage Pointer Audit | - |

# 3 Project Overview

## 3.1 Project Introduction

**Audit Version**

https://github.com/coordinape/coordinape-protocol/tree/feat/fix_from_audit/contracts/ApeProtocol

commit: 29823a493eb5b9b47b6ddd33a0a933b5f2a3b787

https://github.com/OwlOfMoistness/coordinape-vesting-contracts

commit: 97f4fb9316d476596467cbb990fd87993d662788

Audit scope: Audit does not include the token directory in ApeProtocol

**Fixed Version**

https://github.com/coordinape/coordinape-protocol

commit: 59df785cf0d8fba3e27e038cc09a514e4956523e

## 3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

| NO | Title | Category | Level | Status |
|---|---|---|---|---|
| N1 | Risk of replay attack | Replay Vulnerability | Suggestion | Fixed |
| N2 | Missing event records | Others | Suggestion | Fixed |
| N3 | Coding optimization | Authority Control Vulnerability | Suggestion | Fixed |
| N4 | Business logic is not clear | Others | High | Fixed |
| N5 | Coding standards issues | Others | Suggestion | Confirmed |
| N6 | The external call does not judge the return value | Unsafe External Call Audit | Medium | Fixed |

| NO | Title | Category | Level | Status |
|---|---|---|---|---|
| N7 | Excessive authority issue | Authority Control Vulnerability | High | Confirmed |
| N8 | Lack of permission checks | Authority Control Vulnerability | Low | Confirmed |

# 4 Code Overview

## 4.1 Contracts Description

The main network address of the contract is as follows:

**The code was not deployed to the mainnet.**

## 4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

| ApeBeacon | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | BeaconProxy |
| proxyOwner | Public | - | - |
| transferProxyOwnership | External | Can Modify State | - |
| setBeaconDeploymentPrefs | External | Can Modify State | - |

| ApeRegistryBeacon | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |

| ApeRegistryBeacon | | | |
|---|---|---|---|
| <Constructor> | Public | Can Modify State | TimeLock |
| implementation | Public | - | - |
| implementation | Public | - | - |
| setDeploymentPrefs | External | Can Modify State | - |
| pushNewImplementation | Public | Can Modify State | itself |

| ApeUgradeableBeacon | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | UpgradeableBeacon TimeLock |
| upgradeTo | Public | Can Modify State | itself |

| OwnableImplementation | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| owner | Public | - | - |
| renounceOwnership | Public | Can Modify State | onlyOwner |
| transferOwnership | Public | Can Modify State | onlyOwner |

| ApeVaultWrapperImplementation | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| init | External | Can Modify State | - |
| shareValue | Public | - | - |

| ApeVaultWrapperImplementation | | | |
|---|---|---|---|
| sharesForValue | Public | - | - |
| profit | Public | - | - |
| apeWithdrawSimpleToken | Public | Can Modify State | onlyOwner |
| apeWithdraw | External | Can Modify State | onlyOwner |
| exitVaultToken | External | Can Modify State | onlyOwner |
| apeMigrate | External | Can Modify State | onlyOwner |
| tap | External | Can Modify State | onlyDistributor |
| _tapOnlyProfit | Internal | Can Modify State | - |
| _tapBase | Internal | Can Modify State | - |
| _tapSimpleToken | Internal | Can Modify State | - |
| syncUnderlying | External | Can Modify State | onlyOwner |
| addFunds | External | Can Modify State | onlyRouter |
| updateCircleAdmin | External | Can Modify State | onlyOwner |
| updateAllowance | External | Can Modify State | onlyOwner |

| ApeBeacon | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | BeaconProxy |
| proxyOwner | Public | - | - |
| transferProxyOwnership | External | Can Modify State | - |

| ApeBeacon | | | |
|---|---|---|---|
| setBeaconDeploymentPrefs | External | Can Modify State | - |

| ApeVaultFactoryBeacon | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| createApeVault | External | Can Modify State | - |

| BaseWrapperImplementation | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| setRegistry | External | Can Modify State | - |
| bestVault | Public | - | - |
| allVaults | Public | - | - |
| _updateVaultCache | Internal | Can Modify State | - |
| totalVaultBalance | Public | - | - |
| totalAssets | Public | - | - |
| _deposit | Internal | Can Modify State | - |
| _withdraw | Internal | Can Modify State | - |
| _migrate | Internal | Can Modify State | - |
| _migrate | Internal | Can Modify State | - |
| _migrate | Internal | Can Modify State | - |

| ApeAllowanceModule | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| setAllowance | External | Can Modify State | - |
| _isTapAllowed | Internal | Can Modify State | - |
| _updateInterval | Internal | Can Modify State | - |

| ApeDistributor | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| _tap | Internal | Can Modify State | - |
| uploadEpochRoot | External | Can Modify State | - |
| sum | Internal | - | - |
| tapEpochAndDistribute | External | Can Modify State | - |
| updateCircleAdmin | External | Can Modify State | - |
| isClaimed | Public | - | - |
| _setClaimed | Internal | Can Modify State | - |
| claim | Public | Can Modify State | - |
| claimMany | External | Can Modify State | - |

| ApeRegistry | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | TimeLock |

| ApeRegistry | | | |
|---|---|---|---|
| setFeeRegistry | External | Can Modify State | itself |
| setRouter | External | Can Modify State | itself |
| setDistributor | External | Can Modify State | itself |
| setFactory | External | Can Modify State | itself |
| setTreasury | External | Can Modify State | itself |

| ApeRouter | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | TimeLock |
| delegateDepositYvTokens | External | Can Modify State | - |
| delegateDeposit | External | Can Modify State | - |
| delegateWithdrawal | External | Can Modify State | - |
| removeTokens | External | Can Modify State | onlyOwner |
| setRegistry | External | Can Modify State | itself |

| FeeRegistry | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| activateFee | External | Can Modify State | itself |
| shutdownFee | External | Can Modify State | itself |
| staticFee | External | - | - |
| getVariableFee | External | Can Modify State | - |

| **TimeLock** | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| changeMinDelay | External | Can Modify State | itself |
| hashOperation | Internal | - | - |
| isPendingCall | Public | - | - |
| isDoneCall | Public | - | - |
| isReadyCall | Public | - | - |
| schedule | External | Can Modify State | onlyOwner |
| cancel | External | Can Modify State | onlyOwner |
| execute | External | Can Modify State | onlyOwner |
| _call | Internal | Can Modify State | - |

| **CoordinapeCircle** | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | ERC721 |
| invite | External | Can Modify State | onlyRole |
| revoke | External | Can Modify State | onlyRole |
| setupRole | External | Can Modify State | onlyRole |
| setMinimumVouches | External | Can Modify State | onlyOwner |

| CoordinapeCircle | | | |
|---|---|---|---|
| vouch | External | Can Modify State | onlyRole |
| enter | External | Can Modify State | - |
| state | External | - | - |
| members | External | - | - |
| activeMembersCount | Public | - | - |
| inviteOf | Public | - | - |
| permissionsOf | External | - | - |
| hasRole | Public | - | - |
| vouchesOf | External | - | - |
| minimumVouches | External | - | - |
| totalSupply | Public | - | - |
| _epochInProgress | Internal | - | - |
| _issueInvite | Internal | Can Modify State | - |
| _revokeInvite | Internal | Can Modify State | - |
| updateURI | Public | Can Modify State | onlyOwner |
| _baseURI | Internal | - | - |
| transferFrom | Public | Can Modify State | onlyOwner |
| safeTransferFrom | Public | Can Modify State | onlyOwner |
| safeTransferFrom | Public | Can Modify State | onlyOwner |
| _beforeTokenTransfer | Internal | Can Modify State | - |

| CoordinapeEpoch | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | ERC20 |
| addParticipant | Public | Can Modify State | onlyOwner |
| removeParticipant | Public | Can Modify State | onlyOwner |
| editParticipant | Public | Can Modify State | onlyOwner |
| addNote | Public | Can Modify State | onlyParticipant beforeEnd |
| stopReceiving | Public | Can Modify State | onlyParticipant |
| leave | Public | Can Modify State | onlyParticipant |
| participants | Public | - | - |
| receivedOf | Public | - | - |
| permissionsOf | Public | - | - |
| isParticipant | Public | - | - |
| startBlock | Public | - | - |
| endBlock | Public | - | - |
| ended | Public | - | - |
| decimals | Public | - | - |
| _beforeTokenTransfer | Internal | Can Modify State | - |

| Vesting | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| min | Private | - | - |
| max | Private | - | - |
| createVehicule | External | Can Modify State | onlyOwner |
| killVehicule | External | Can Modify State | onlyOwner |
| endVehicule | External | Can Modify State | onlyOwner |
| fetchTokens | External | Can Modify State | onlyOwner |
| claim | External | Can Modify State | - |
| _claimUpfront | Private | Can Modify State | - |
| balanceOf | External | - | - |
| pendingReward | Public | - | - |
| claimed | External | - | - |

## 4.3 Vulnerability Summary

**[N1] [Suggestion] Risk of replay attack**

**Category: Replay Vulnerability**

**Content**

In the ApeToken contract, DOMAIN_SEPARATOR is defined when the contract is initialized, but it is not reimplemented when DOMAIN_SEPARATOR is used in the permit function. So the DOMAIN_SEPARATOR contains the chainId and is defined at contract deployment instead of reconstructed for every signature, there is a risk of possible replay attacks between chains in the event of a future chain split.

- coordinape-protocol/contracts/ApeProtocol/token/ApeToken.sol#L18

```
        constructor() {
                uint chainId = block.chainid;
        DOMAIN_SEPARATOR = keccak256(
            abi.encode(
                keccak256('EIP712Domain(string name,string version,uint256 
chainId,address verifyingContract)'),
                keccak256(bytes("coordinape.com")),
                keccak256(bytes('1')),
                chainId,
                address(this)
            )
        );
        }
```

- coordinape-protocol/contracts/ApeProtocol/token/ApeToken.sol#L42

```
        function permit(address owner, address spender, uint256 value, uint256 
deadline, uint8 v, bytes32 r, bytes32 s) public {
        require(block.timestamp <= deadline, "COToken: expired deadline");
        require(owner != address(0), "COToken: owner can't be ZERO address ");

        bytes32 digest = keccak256(
            abi.encode(
                                '\x19\x01',
                DOMAIN_SEPARATOR,
                keccak256(abi.encode(_PERMIT_TYPEHASH, owner, spender, value, 
nonces[owner]++, deadline))
            )
        );

        address signer = ECDSA.recover(digest, v, r, s);
        require(signer == owner, "COToken: invalid signature");
        _approve(owner, spender, value);
    }
```

**Solution**

It is recommended to redefine when using DOMAIN_SEPARATOR.

Reference: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2612.md

**Status**

Fixed

### [N2] [Suggestion] Missing event records

**Category: Others**

**Content**

Missing event records are not conducive to the review of community users.

- coordinape-protocol/contracts/ApeProtocol/token/TokenAccessControl.sol#L32-47

```
        function disableAllowlist() external onlyOwner {
                require(!allowlistDisabled, "AccessControl: Allowlist already 
disabled");
                allowlistDisabled = true;
        }

        function changePauseStatus(bool _status) external onlyOwner {
                require(!foreverUnpaused, "AccessControl: Contract is unpaused 
forever");
                paused = _status;
        }


        function disablePausingForever() external onlyOwner {
                require(!foreverUnpaused, "AccessControl: Contract is unpaused 
forever");
                foreverUnpaused = true;
                paused = false;
        }
```

- coordinape-protocol/contracts/ApeProtocol/token/TokenAccessControl.sol#L81-84

```
        function disableMintingForever() external onlyOwner {
                require(!mintingDisabled, "AccessControl: Contract cannot mint
anymore");
                mintingDisabled = true;
        }
```

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/ApeBeacon.sol#L23-28

```
        function transferProxyOwnership(address _newOwner) external {
                require(msg.sender == proxyOwner());
                assembly {
            sstore(_OWNER_SLOT, _newOwner)
        }
        }
```

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/ApeRegistryBeacon.sol#37-L40

```
        function pushNewImplementation(address _newImplementation) public itself {
                require(Address.isContract(_newImplementation), "ApeRegistryBeacon:
implementaion is not a contract");
                deployments[++deploymentCount] = _newImplementation;
        }
```

- coordinape-protocol/contracts/ApeProtocol/ApeRegistry.sol#L17-L35

```
function setFeeRegistry(address _registry) external itself {
                feeRegistry = _registry;
        }

        function setRouter(address _router) external itself {
                router = _router;
        }

        function setDistributor(address _distributor) external itself {
                distributor = _distributor;
        }
```

```
	function setFactory(address _factory) external itself {
		factory = _factory;
	}

	function setTreasury(address _treasury) external itself {
		treasury = _treasury;
	}
```

coordinape-protocol/contracts/ApeProtocol/ApeRouter.sol#L87-L89

```
	function setRegistry(address _registry) external itself {
		yearnRegistry = _registry;
	}
```

**Solution**

It is recommended to add event records to facilitate review by community users.

**Status**

Fixed

**[N3] [Suggestion] Coding optimization**

**Category: Authority Control Vulnerability**

**Content**

getVariableFee function does not modify contract data, but does not use view.

- coordinape-protocol/contracts/ApeProtocol/FeeRegistry.sol#L24

```
function getVariableFee(uint256 _yield, uint256 _tapTotal) external returns(uint256
variableFee) {
		if (!on)
			return 0;
		uint256 yieldRatio = _yield * 1000 / _tapTotal;
		uint256 baseFee = 100;
		if (yieldRatio >= 900)
			variableFee = baseFee;		// 1%	  @ 90% yield ratio
		else if (yieldRatio >= 800)
			variableFee = baseFee + 25;	// 1.25%  @ 80% yield ratio
```

```
            else if (yieldRatio >= 700)
                    variableFee = baseFee + 50;   // 1.50%  @ 70% yield ratio
            else if (yieldRatio >= 600)
                    variableFee = baseFee + 75;   // 1.75%  @ 60% yield ratio
            else if (yieldRatio >= 500)
                    variableFee = baseFee + 100;  // 2.00%  @ 80% yield ratio
            else if (yieldRatio >= 400)
                    variableFee = baseFee + 125;  // 2.25%  @ 80% yield ratio
            else if (yieldRatio >= 300)
                    variableFee = baseFee + 150;  // 2.50%  @ 80% yield ratio
            else if (yieldRatio >= 200)
                    variableFee = baseFee + 175;  // 2.75%  @ 80% yield ratio
            else if (yieldRatio >= 100)
                    variableFee = baseFee + 200;  // 3.00%  @ 80% yield ratio
            else
                    variableFee = baseFee + 250;  // 3.50%  @  0% yield ratio
    }
```

**Solution**

It is recommended to add view visibility.

**Status**

Fixed

### [N4] [High] Business logic is not clear

**Category: Others**

**Content**

The _issueInvite function will execute the mint logic, but the burn is annotated in _revokeInvite, the business logic is not clear.

- coordinape-protocol/contracts/circles_obsolete/CoordinapeCircle.sol#L158-L175

```
    function _issueInvite(address recipient, uint8 role) internal {
        Counters.increment(_inviteIds);
        uint256 tokenId = Counters.current(_inviteIds);
        _mint(recipient, tokenId);
        _roles[tokenId] = role;
        _invites[recipient] = tokenId;
        _vouches[recipient] = 0;
        emit InviteIssued(recipient, role);
    }

    function _revokeInvite(address recipient) internal {
        uint256 tokenId = _invites[recipient];
        _inactiveMembers.increment();
        //_burn(tokenId);
        _roles[tokenId] = 0;
        _invites[recipient] = 0;
        emit InviteRevoked(recipient, 0);
    }
```

_epochEnds is never initialized, and used in `_epochInProgress` function.

- coordinape-protocol/contracts/circles_obsolete/CoordinapeCircle.sol#L155

```
    function _epochInProgress() internal view returns (bool) {
        uint256 epochId = Counters.current(_epochIds);
        // return epochId > 0 && !CoordinapeEpoch(_epochs[epochId]).ended();
        return epochId > 0 && block.number < _epochEnds[epochId];
    }
```

_epochState is never initialized. and it is used in `state` function.

- coordinape-protocol/contracts/circles_obsolete/CoordinapeCircle.sol#L105

```
    function state(uint256 _epoch) external view returns (uint8) {
        return _epochState[_epoch];
    }
```

The address passed in by the _migrate function is address(this), which means migrating to the address(this) contract, the logic here is not clear.

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/ApeVault.sol#L166-L169

```
        function apeMigrate() external onlyOwner returns(uint256 migrated){
                migrated = _migrate(address(this));
                vault = VaultAPI(registry.latestVault(address(token)));
        }
```

`migrated = _deposit(address(this), account, withdrawn, false);` account is address(this), the logic here is wrong

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/BaseWrapperImplementation.sol#L387-L427

```
function _migrate(address account) internal returns (uint256) {
        return _migrate(account, MIGRATE_EVERYTHING);
    }

    function _migrate(address account, uint256 amount) internal returns (uint256) {
        // NOTE: In practice, it was discovered that <50 was the maximum we've see
for this variance
        return _migrate(account, amount, 0);
    }

    function _migrate(
        address account,
        uint256 amount,
        uint256 maxMigrationLoss
    ) internal returns (uint256 migrated) {
        VaultAPI _bestVault = bestVault();

        // NOTE: Only override if we aren't migrating everything
        uint256 _depositLimit = _bestVault.depositLimit();
        uint256 _totalAssets = _bestVault.totalAssets();
        if (_depositLimit <= _totalAssets) return 0; // Nothing to migrate (not a
failure)

        uint256 _amount = amount;
        if (_depositLimit < UNCAPPED_DEPOSITS && _amount < WITHDRAW_EVERYTHING) {
            // Can only deposit up to this amount
            uint256 _depositLeft = _depositLimit.sub(_totalAssets);
            if (_amount > _depositLeft) _amount = _depositLeft;
        }
        if (_amount > 0) {
            // NOTE: `false` = don't withdraw from `_bestVault`
            uint256 withdrawn = _withdraw(account, address(this), _amount, false);
            if (withdrawn == 0) return 0; // Nothing to migrate (not a failure)

            // NOTE: `false` = don't do `transferFrom` because it's already local
            migrated = _deposit(address(this), account, withdrawn, false);
            // NOTE: Due to the precision loss of certain calculations, there is a
small inefficency
            //       on how migrations are calculated, and this could lead to a DoS
issue. Hence, this
            //       value is made to be configurable to allow the user to specify
how much is acceptable
            require(withdrawn.sub(migrated) <= maxMigrationLoss);
        } // else: nothing to migrate! (not a failure)
    }
```

The return value of decimals is 0, and developers need to confirm the business logic here.

- coordinape-protocol/contracts/circles_obsolete/CoordinapeEpoch.sol#L143-L145

```
    function decimals() public pure override returns (uint8) {
        return 0;
    }
```

**Solution**

It is recommended that developers check the business logic of the code and supplement the implementation of the code.

**Status**

Fixed; The issue has been fixed in commit: 59df785cf0d8fba3e27e038cc09a514e4956523e.

The project team response: The circle contracts are not going to be deployed. We have a PR on this branch to completely remove them from the directory.

**[N5] [Suggestion] Coding standards issues**

**Category: Others**

**Content**

Executed first `_call(id, _target, _data);` and then executed `timestamps[id] = _DONE_TIMESTAMP;`, which does not meet the specification(Checks-Effects-Interactions).

- coordinape-protocol/contracts/ApeProtocol/TimeLock.sol#L72-L73

```
        function execute(address _target, bytes calldata _data, bytes32 _predecessor,
bytes32 _salt, uint256 _delay) external onlyOwner {
                bytes32 id = hashOperation(_target, _data, _predecessor, _salt);
                require(isReadyCall(id), "TimeLock: Not ready for execution or
executed");
                require(_predecessor == bytes32(0) || isDoneCall(_predecessor),
"TimeLock: Predecessor call not executed");
                _call(id, _target, _data);
                timestamps[id] = _DONE_TIMESTAMP;
        }

        function _call(
        bytes32 id,
        address target,
        bytes calldata data
    ) internal {
        (bool success, ) = target.call(data);
        require(success, "Timelock: underlying transaction reverted");

        emit CallExecuted(id, target, data);
    }
```

**Solution**

It is recommended to follow the coding convention of Checks-Effects-Interactions.

**Status**

Confirmed

### [N6] [Medium] The external call does not judge the return value

**Category: Unsafe External Call Audit**

**Content**

The external call in the _withdraw function does not evaluate the return value,

E.g：`vaults[id].transferFrom`, `vault.transfer`, `IERC20(_token).transfer`

If the transferFrom function and transfer of the externally called token contract return false, the code logic will be wrong.

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/BaseWrapperImplementation.sol#L353-L362

```
    function _withdraw(
        address sender,
        address receiver,
        uint256 amount, // if `MAX_UINT256`, just withdraw everything
        bool withdrawFromBest // If true, also withdraw from `_bestVault`
    ) internal returns (uint256 withdrawn) {
        VaultAPI _bestVault = bestVault();

        VaultAPI[] memory vaults = allVaults();
        _updateVaultCache(vaults);

        // NOTE: This loop will attempt to withdraw from each Vault in `allVaults`
that `sender`
        //       is deposited in, up to `amount` tokens. The withdraw action can be
expensive,
        //       so it if there is a denial of service issue in withdrawing, the
downstream usage
        //       of this wrapper contract must give an alternative method of
withdrawing using
        //       this function so that `amount` is less than the full amount
requested to withdraw
        //       (e.g. "piece-wise withdrawals"), leading to less loop iterations
such that the
        //       DoS issue is mitigated (at a tradeoff of requiring more txns from
the end user).
        for (uint256 id = 0; id < vaults.length; id++) {
            if (!withdrawFromBest && vaults[id] == _bestVault) {
                continue; // Don't withdraw from the best
            }
            // Start with the total shares that `sender` has
            uint256 availableShares = vaults[id].balanceOf(sender);

            // Restrict by the allowance that `sender` has to this contract
            // NOTE: No need for allowance check if `sender` is this contract
            if (sender != address(this)) {
                availableShares = Math.min(availableShares,
vaults[id].allowance(sender, address(this)));
            }

            // Limit by maximum withdrawal size from each vault
            availableShares = Math.min(availableShares,
vaults[id].maxAvailableShares());

            if (availableShares > 0) {
                // Intermediate step to move shares to this contract before
withdrawing
                // NOTE: No need for share transfer if this contract is `sender`
                // if (sender != address(this)) vaults[id].transferFrom(sender,
address(this), availableShares);

                if (amount != WITHDRAW_EVERYTHING) {
                    // Compute amount to withdraw fully to satisfy the request
                    uint256 estimatedShares =
                        amount
                            .sub(withdrawn) // NOTE: Changes every iteration
                            .mul(10**uint256(vaults[id].decimals()))
                            .div(vaults[id].pricePerShare()); // NOTE: Every Vault is
different

                    // Limit amount to withdraw to the maximum made available to this
contract
                    // NOTE: Avoid corner case where `estimatedShares` isn't precise
enough
                    // NOTE: If `0 < estimatedShares < 1` but `availableShares > 1`,
this will withdraw more than necessary
                    if (estimatedShares > 0 && estimatedShares < availableShares) {
                        if (sender != address(this)) vaults[id].transferFrom(sender,
address(this), estimatedShares);
                        withdrawn =
withdrawn.add(vaults[id].withdraw(estimatedShares));
                    } else {
                        if (sender != address(this)) vaults[id].transferFrom(sender,
address(this), availableShares);
```

```
                        withdrawn =
withdrawn.add(vaults[id].withdraw(availableShares));
                    }
                } else {
                    if (sender != address(this)) vaults[id].transferFrom(sender,
address(this), availableShares);
                    withdrawn = withdrawn.add(vaults[id].withdraw());
                }

                // Check if we have fully satisfied the request
                // NOTE: use `amount = WITHDRAW_EVERYTHING` for withdrawing
everything
                if (amount <= withdrawn) break; // withdrawn as much as we needed
            }
        }

        // If we have extra, deposit back into `_bestVault` for `sender`
        // NOTE: Invariant is `withdrawn <= amount`
        if (withdrawn > amount && withdrawn.sub(amount) >
_bestVault.pricePerShare().div(10**_bestVault.decimals())) {
            // Don't forget to approve the deposit
            if (token.allowance(address(this), address(_bestVault)) <
withdrawn.sub(amount)) {
                token.safeApprove(address(_bestVault), UNLIMITED_APPROVAL); // Vaults
are trusted
            }

            _bestVault.deposit(withdrawn.sub(amount), sender);
            withdrawn = amount;
        }

        // `receiver` now has `withdrawn` tokens as balance
        if (receiver != address(this)) token.safeTransfer(receiver, withdrawn);
    }
```

- coordinape-protocol/contracts/ApeProtocol/ApeDistributor.sol#L147

```
function tapEpochAndDistribute(
                address _vault,
                bytes32 _circle,
                address _token,
                address[] calldata _users,
                uint256[] calldata _amounts,
                uint256 _amount,
                uint8 _tapType)
                external {
                require(_users.length == _amounts.length, "ApeDistributor: Array
lengths do not match");
                require(sum(_amounts) == _amount, "ApeDistributor: Amount does not
match sum of values");

                _tap(_vault, _circle, _token, _amount, _tapType,
bytes32(type(uint256).max));

                for (uint256 i = 0; i < _users.length; i++)
                        IERC20(_token).transfer(_users[i], _amounts[i]);
        }
```

- coordinape-vesting-contracts/contracts/Vesting.sol#L77-L99

```
        function fetchTokens(uint256 _amount) external onlyOwner {
                IERC20(co).transfer(msg.sender, _amount);
        }

        function claim(uint256 _index) external override {
                uint256 _now = block.timestamp;

                Vehicule storage vehicule = vehicules[msg.sender][_index];

                uint256 upfront = _claimUpfront(vehicule);
                uint256 start = vehicule.start;
                if (start == 0)
                        revert("Vesting: vehicule does not exist");
                require(_now > start, "Vesting: cliff !started");
                uint256 end = vehicule.end;
                uint256 elapsed = min(end, _now) - start;
                uint256 maxDelta = end - start;
                // yield = amount * delta / vest_duration - claimed_amount
                uint256 yield = (vehicule.amount * elapsed / maxDelta) -
vehicule.claimed;
                vehicule.claimed += yield;
                IERC20(co).transfer(msg.sender, yield + upfront);
                emit YieldClaimed(msg.sender, yield);
        }
```

**Solution**

It is recommended that all external calls involving transfers should judge the return value or use safeTransfer or safeTransferFrom.

**Status**

Fixed

**[N7] [High] Excessive authority issue**

**Category: Authority Control Vulnerability**

**Content**

Owner can transfer assets in the contract.

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/ApeVault.sol#L129

```
        function apeWithdrawSimpleToken(uint256 _amount) public onlyOwner {
                simpleToken.safeTransfer(msg.sender, _amount);
        }
```

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/ApeVault.sol#L139

```
        function apeWithdraw(uint256 _shareAmount, bool _underlying) external
onlyOwner {
                uint256 underlyingAmount = shareValue(_shareAmount);
                require(underlyingAmount <= underlyingValue, "underlying amount
higher than vault value");

                address router = ApeRegistry(apeRegistry).router();
                underlyingValue -= underlyingAmount;
                vault.safeTransfer(router, _shareAmount);
                ApeRouter(router).delegateWithdrawal(owner(), address(this),
vault.token(), _shareAmount, _underlying);
        }
```

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/ApeVault.sol#L154

```
	function exitVaultToken(bool _underlying) external onlyOwner {
		underlyingValue = 0;
		uint256 totalShares = vault.balanceOf(address(this));
		address router = ApeRegistry(apeRegistry).router();
		vault.safeTransfer(router, totalShares);
		ApeRouter(router).delegateWithdrawal(owner(), address(this),
vault.token(), totalShares, _underlying);
	}
```

Owner can transfer the tokens in the contract.

- coordinape-vesting-contracts/contracts/Vesting.sol#L77

```
	function fetchTokens(uint256 _amount) external onlyOwner {
		IERC20(co).transfer(msg.sender, _amount);
	}
```

**Solution**

It is recommended to transfer the Owner's authority to the timelock contract or governance contract, at least a multi-sign contract.

The project team response: The money in the vaults belongs to the "Owner" and they need to have free ability to move the funds as they see fit. This one is not an issue but is the intended design.

The SlowMist security team response: `Change from high risk to the enhanced recommendation`. It is recommended to use a multi-signature contract to manage the Owner's wallet.

**Status**

Confirmed

### [N8] [Low] Lack of permission checks

**Category: Authority Control Vulnerability**

**Content**

The createApeVault function does not perform permission checks. Anyone can create ApeVault. If the incoming parameters are malicious (malicious Token or incompatible Token), it will affect the funds in the project.

- coordinape-protocol/contracts/ApeProtocol/wrapper/beacon/ApeVaultFactory.sol#L22-L27

```
        function createApeVault(address _token, address _simpleToken) external {
                bytes memory data =
abi.encodeWithSignature("init(address,address,address,address,address)", apeRegistry,
_token, yearnRegistry, _simpleToken, msg.sender);
                ApeBeacon proxy = new ApeBeacon(beacon, msg.sender, data);
                vaultRegistry[address(proxy)] = true;
                emit VaultCreated(address(proxy));
        }
```

**Solution**

It is recommended to add permission checks and perform security audits on contracts when creating ApeVault to ensure that contracts and projects are compatible.

The project team response: Due to the fact that all calls don't interact with more than one token at a time, or are gated by modifiers, and conform to the “Check, Effect, Interact” pattern, even if a malicious contract was injected, it would not be able to interact with any other tokens, steal funds, or deteriorate the experience of users (unless users are being paid in tokens of the malicious contract, which is also limited to the circle the users are interacting with).

**Status**

Confirmed

# 5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
|---|---|---|---|
| 0X002204270002 | SlowMist Security Team | 2022.04.18 - 2022.05.01 | Low Risk |

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project. During the audit work, we found 2 high risk, 1 medium risk, 1 low risk, and 4 suggestion vulnerabilities. All the items were thoroughly addressed and effectively mitigated. Note that high risk issue N4 was found in code that was removed from the project entirely. N7 was downgraded to "Enhanced Recommendation" after feedback about the product design received.

# 6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

**Official Website**

www.slowmist.com

**E-mail**

team@slowmist.com

**Twitter**

@SlowMist_Team

**Github**

https://github.com/slowmist