# Smart Contract Security Audit Report

# Table Of Contents

# 1 Executive Summary

On 2024.06.24, the SlowMist security team received the Knight Safe Contracts v2 team's security audit application for Knight Safe Contracts v2, 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. |
| 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 |
| 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 |

| Serial Number | Audit Class | Audit Subclass |
| --- | --- | --- |
| 7 | Security Design 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

KnightSafe brings smart contract based custody solution for institutional clients and professional traders, enabling them an efficient and secure access to DeFi protocols. KnightSafe facilitates professional crypto funds and traders with seamless interaction with dApps and DeFi protocols, under role-based access control, segregation of authorization, designated scope of authority and duties.

The KnightSafe protocol mainly includes management, common, control center, event emitter, protocol analyzer, oracle, proxy factory, settings, and transaction modules. Among them, the management module is primarily used for managing policies, admins, and control center changes; the common module is mainly used to implement common module functions, such as allowing contracts to receive native tokens, contract pause functionality, token callbacks, etc.; the control center is used to manage policies, user whitelists, transaction

limits, and more; the event emitter is used for event triggering; the protocol analyzer is used to parse functions of external protocols; the oracle provides necessary token prices for the transaction module to calculate trading volume; the proxy factory is used to create KnightSafe proxy contracts; the settings module is used to call the control center to modify sensitive parameters; and the transaction module is used for specific transaction requests and execution.

## 3.2 Vulnerability Information

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

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| N1 | `_backupOwner` not reset after owner takeover | Design Logic Audit | Suggestion | Fixed |
| N2 | Cautious Operation Policy 0 | Others | Information | Acknowledged |
| N3 | Missing event records | Others | Suggestion | Fixed |
| N4 | Repeat initialization event items | Design Logic Audit | Suggestion | Fixed |
| N5 | Potential risks of signature malleability | Authority Control Vulnerability Audit | Medium | Fixed |
| N6 | Potential risks of not resetting the isKnightSafe status | Design Logic Audit | Low | Fixed |
| N7 | No check on Oracle price validity | Design Logic Audit | Medium | Fixed |
| N8 | Redundant price return value processing | Others | Suggestion | Fixed |
| N9 | Incorrect event logging | Others | Low | Fixed |
| N10 | Any trader can reject a transaction | Authority Control Vulnerability Audit | Information | Acknowledged |
| N11 | Potential risks of misusing | Authority Control Vulnerability | Information | Acknowledged |

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| | useGlobalWhitelist | Audit | | |
| N12 | Transaction processing issue | Authority Control Vulnerability Audit | Information | Acknowledged |
| N13 | Incorrect address parsing in emergencyEtherTransfer | Design Logic Audit | High | Fixed |
| N14 | Wrong parsing of onBehalfOf position in depositETH | Design Logic Audit | High | Fixed |
| N15 | Risk of not being able to perform analysis with GMXAnalyser | Design Logic Audit | Critical | Fixed |
| N16 | Redundant GMX v1 selectors | Others | Suggestion | Fixed |
| N17 | Potential risks of not being able to reset the Gov address | Design Logic Audit | Suggestion | Fixed |
| N18 | Incorrect Gov Address Check | Authority Control Vulnerability Audit | Critical | Fixed |
| N19 | Incorrect PERMIT2_TRANSFER_FROM_BATCH parameter handling in UniswapAnalyser | Design Logic Audit | Critical | Fixed |
| N20 | Potential risks of proxy deployment | Design Logic Audit | Medium | Fixed |
| N21 | Potential flaws in trading volume calculations | Design Logic Audit | Low | Acknowledged |

# 4 Code Overview

## 4.1 Contracts Description

**Audit Version:**

https://github.com/knight-safe/knight-safe-contracts-v2

commit: 4a8b4535e019f365631c8430b5698cbafc6c51e7

**Fixed Version:**

https://github.com/knight-safe/knight-safe-contracts-v2

commit: 4606701a0aeaee830b0d6b0f847101a6ff62ba57

The main network address of the contract is as follows:

| Contract Name | Contract Address | Chain |
| --- | --- | --- |
| ControlCenter | 0x97479A238518498B393c3F5918D590e2bf0E34dA | Ethereum Mainnet |
| TokenCallbackHandler | 0xb275B576a450A6bf7BEF60354260295511Cc8709 | Ethereum Mainnet |
| KnightSafeMaster | 0xE6a21B617B2E1Ab3E5b13395A3691b01fE9C0030 | Ethereum Mainnet |
| KnightSafeProxyFactory | 0xad5F9134b9cC50B74a4F7A370c9F752cC1597a71 | Ethereum Mainnet |
| ERC20Analyser | 0xAfA38DEDEe60E973CFe2E6F101b9bd68Add16FAB | Ethereum Mainnet |
| ControlCenter | 0x97479A238518498B393c3F5918D590e2bf0E34dA | Arbitrum One Mainnet |
| TokenCallbackHandler | 0xb275B576a450A6bf7BEF60354260295511Cc8709 | Arbitrum One Mainnet |
| KnightSafeMaster | 0xE6a21B617B2E1Ab3E5b13395A3691b01fE9C0030 | Arbitrum One Mainnet |
| KnightSafeProxyFactory | 0xad5F9134b9cC50B74a4F7A370c9F752cC1597a71 | Arbitrum One Mainnet |
| ERC20Analyser | 0xAfA38DEDEe60E973CFe2E6F101b9bd68Add16FAB | Arbitrum One Mainnet |
| ControlCenter | 0x97479A238518498B393c3F5918D590e2bf0E34dA | Base Mainnet |
| TokenCallbackHandler | 0xb275B576a450A6bf7BEF60354260295511Cc8709 | Base Mainnet |
| KnightSafeMaster | 0xE6a21B617B2E1Ab3E5b13395A3691b01fE9C0030 | Base Mainnet |
| KnightSafeProxyFactory | 0xad5F9134b9cC50B74a4F7A370c9F752cC1597a71 | Base Mainnet |
| ERC20Analyser | 0xAfA38DEDEe60E973CFe2E6F101b9bd68Add16FAB | Base Mainnet |

## 4.2 Visibility Description

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

| ControlCenterManager | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| _setControlCenter | Internal | Can Modify State | - |

| FallbackManager | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| _internalSetFallbackHandler | Internal | Can Modify State | - |
| <Fallback> | External | Can Modify State | - |

| OwnerManager | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| _checkOwner | Internal | - | - |
| _checkBackupOwner | Private | - | - |
| _initOwnerManager | Internal | Can Modify State | - |
| getOwner | Public | - | - |
| getIsTakeoverInProgress | Public | - | - |
| getTakeoverTimestamp | Public | - | - |
| getTakeoverStatus | Public | - | - |
| setBackupOwner | Public | Can Modify State | onlyOwner |
| requestTakeover | Public | Can Modify State | - |
| confirmTakeover | Public | Can Modify State | - |
| instantTakeover | Public | Can Modify State | - |

| OwnerManager | | | |
|---|---|---|---|
| revokeTakeover | Public | Can Modify State | - |
| isAdmin | Public | - | - |
| getAdmins | External | - | - |
| addAdmin | Public | Can Modify State | onlyOwner |
| removeAdmin | Public | Can Modify State | onlyOwner |
| _takeover | Private | Can Modify State | - |

| PolicyManager | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| _initPolicyManager | Internal | Can Modify State | - |
| getActivePolicyIds | Public | - | - |
| isActivePolicy | External | - | - |
| createPolicy | External | Can Modify State | onlyAdminOrOwner |
| removePolicy | Public | Can Modify State | onlyAdminOrOwner |
| getTraders | External | - | - |
| isTrader | Public | - | - |
| addTrader | Public | Can Modify State | onlyOwner |
| removeTrader | Public | Can Modify State | onlyAdminOrOwner |
| getWhitelistAddresses | External | - | - |
| isPolicyWhitelistAddress | Public | - | - |
| isPolicyOrGlobalWhitelistAddress | Public | - | - |
| updateWhitelist | External | Can Modify State | onlyOwner |
| removeWhitelist | Public | Can Modify State | onlyAdminOrOwner |

| PolicyManager | | | |
|---|---|---|---|
| getKnightSafeAnalyserAddress | Public | - | - |
| getMaxSpendingLimit | Public | - | - |
| getDailyVolumeSpent | Public | - | - |
| setMaxSpendingLimit | Public | Can Modify State | onlyOwner |
| reduceSpendingLimit | Public | Can Modify State | onlyAdminOrOwner |
| resetDailySpent | Public | Can Modify State | onlyOwner |

| ControlCenter | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | EventEmitter |
| _checkAdmin | Private | - | - |
| setAdmin | Public | Can Modify State | onlyOwner |
| isOfficialImplementation | Public | - | - |
| addOfficialImplementation | Public | Can Modify State | onlyOwner |
| removeOfficialImplementation | Public | Can Modify State | onlyOwner |
| isOfficialAnalyser | Public | - | - |
| addOfficialAnalyser | Public | Can Modify State | onlyAdmin |
| removeOfficialAnalyser | Public | Can Modify State | onlyAdmin |
| isSpendingLimitEnabled | Public | - | - |
| setSpendingLimitEnabled | Public | Can Modify State | onlyAdmin |
| getMaxPolicyAllowed | Public | - | - |
| setMaxPolicyAllowed | Public | Can Modify State | onlyAdmin |
| setGlobalMinPolicyAllowed | Public | Can Modify State | onlyOwner |

| ControlCenter | | | |
|---|---|---|---|
| getAdminEventAccess | External | - | - |
| getAdminEventAccessCount | External | - | - |
| getAdminEventAccessById | Public | - | - |
| _isKnightSafeAnalyser | Internal | - | - |
| setPriceFeed | Public | Can Modify State | onlyOwner |
| getPriceFeed | Public | - | - |
| getDailyVolume | Public | - | - |
| setDailyVolume | Public | Can Modify State | onlyAdmin |
| setDailyVolumeExpiryDate | Public | Can Modify State | onlyAdmin |
| setMaxTradingVolume | Public | Can Modify State | onlyAdmin |
| setMaxTradingVolumeExpiryDate | Public | Can Modify State | onlyAdmin |
| getMaxTradingVolume | Public | - | - |
| getMaxVolumeExpiryDate | Public | - | - |

| NativeCurrencyReceiver | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| _emitReceive | Internal | Can Modify State | - |
| <Receive Ether> | External | Payable | - |

| PausableUtils | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| pause | External | Can Modify State | onlyAdminOrOwner |
| unpause | External | Can Modify State | onlyOwner |

| SignatureValidator | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| isValidSignature | External | - | - |
| _recoverSigner | Internal | - | - |

| TokenCallbackHandler | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| onERC1155Received | External | - | - |
| onERC1155BatchReceived | External | - | - |
| onERC721Received | External | - | - |
| tokensReceived | External | - | - |
| supportsInterface | External | - | - |

| AaveAnalyser | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| extractAddressesWithValue | External | - | - |
| _getAddressAndAssetId | Internal | - | - |

| AaveUtilsAnalyser | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| extractAddressesWithValue | External | - | - |
| _mapRewardAmount | Private | - | - |

| BaseKnightSafeAnalyser | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| getSelector | Public | - | - |
| supportsInterface | Public | - | - |
| _getBytes32FromBytes | Internal | - | - |
| _getAddressFromBytes | Internal | - | - |
| _getUintFromBytes | Internal | - | - |
| _getBoolFromBytes | Internal | - | - |
| _getAddressArray | Internal | - | - |

| ERC20Analyser | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| extractAddressesWithValue | External | - | - |

| GMXAnalyser | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| updateFeeReceiver | Public | Can Modify State | onlyOwner |
| extractAddressesWithValue | External | - | - |
| _getAddressArrayFromMulticall | Internal | - | - |
| _dispatch | Internal | - | - |
| _map | Internal | - | - |
| _getClaimableAmount | Private | - | - |

| UniswapAnalyser | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| updateMaxFeeBips | Public | Can Modify State | - |
| extractAddressesWithValue | External | - | - |
| _getAddressListForChecking | Private | - | - |
| _getAddressArrayFromExecute | Private | - | - |
| _dispatch | Private | - | - |
| _toPathTokensV3 | Internal | - | - |
| _toPathTokensV2 | Internal | - | - |
| _toBytes | Internal | - | - |
| _map | Internal | - | - |

| ChainlinkPriceFeed | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| _checkOwner | Internal | - | - |
| isControlCenter | Public | - | - |
| getNativeToken | Public | - | - |
| setPriceFeed | Public | Can Modify State | onlyOwner |
| batchSetPriceFeed | Public | Can Modify State | onlyOwner |
| getPriceFeed | Public | - | - |
| getTransactionVolume | External | - | - |
| getNativeTokenVolume | Public | - | - |

| ChainlinkPriceFeed | | | |
|---|---|---|---|
| getChainlinkDataFeedLatestAnswer | Public | - | - |
| _scalePrice | Internal | - | - |

| KnightSafeProxy | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| _implementation | Internal | - | - |

| KnightSafeProxyFactory | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| computeAddress | External | - | - |
| createProxy | Public | Can Modify State | - |
| _deployProxy | Private | Can Modify State | - |

| SettingRequest | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| getNextSettingRequestId | Public | - | - |
| getSettingRequest | Public | - | - |
| requestSetting | Public | Can Modify State | onlyAdminOrOwner |
| _updateSettingRequestStatus | Private | Can Modify State | - |
| executeSettingByReqId | Public | Can Modify State | onlyOwner |
| cancelSettingByReqId | Public | Can Modify State | onlyAdminOrOwner |
| rejectSettingByReqId | Public | Can Modify State | onlyOwner |

| TransactionRequest | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| getNextTransactionRequestId | Public | - | - |
| getTransactionRequest | Public | - | - |
| getTotalVolumeSpent | Public | - | - |
| validateTradingAccess | Public | - | - |
| validateTradingLimit | Public | Can Modify State | - |
| validatePolicyLimit | Public | Can Modify State | - |
| requestTransaction | Public | Can Modify State | onlyTrader |
| cancelTransactionByReqId | Public | Can Modify State | onlyTrader |
| rejectTransactionByReqId | Public | Can Modify State | onlyTrader |
| executeTransaction | Public | Can Modify State | - |
| executeTransactionByReqId | Public | Can Modify State | - |
| _updateTransactionRequestStatus | Private | Can Modify State | - |
| _executeTransaction | Internal | Can Modify State | onlyTrader whenNotPaused |
| _updateVolumeSpent | Private | Can Modify State | - |

| KnightSafe | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| initialize | External | Can Modify State | - |
| updateFallbackHandler | External | Can Modify State | onlyOwner |

| KnightSafe | | | |
|---|---|---|---|
| updateControlCenter | External | Can Modify State | onlyOwner |

## 4.3 Vulnerability Summary

### [N1] [Suggestion] `_backupOwner` not reset after owner takeover

**Category: Design Logic Audit**

**Content**

In the OwnerManager contract, the owner can modify the `_backupOwner` role through the setBackupOwner function. The `_backupOwner` role can submit an owner takeover request through the requestTakeover function and take over the existing owner through the confirmTakeover function after a waiting delay. After the takeover is completed, the `_isTakeoverInProgress` status will be set to false, but the `_backupOwner` will not be cleared, which makes both the `owner` and `_backupOwner` roles the same address. This may cause confusion when users retrieve `_backupOwner` through getTakeoverStatus.

Solution: It is recommended to set the `_backupOwner` role to the 0 address after completing the owner takeover.

Code location: contracts/base/OwnerManager.sol#L146-L149

```
function _takeover() private {
    _owner = _backupOwner;
    _isTakeoverInProgress = false;
}
```

**Solution**

**Status**

Fixed

### [N2] [Information] Cautious Operation Policy 0

**Category: Others**

**Content**

In the protocol, only allowed traders can request transactions and access control is mainly achieved through the onlyTrader modifier in the PolicyManager contract. The onlyTrader modifier checks that the caller must be included in an active policy using `!isTrader(0, _msgSender()) && !isTrader(policyId, _msgSender())`, where the policy ID is passed in by the caller from the outside. It is important to note that as long as the caller is included in policy 0, they can initiate transaction requests regardless of whether they are included in the specified active policy id. This means that addresses in policy 0 have greater permissions, and users should be cautious when operating with policy 0.

Code location: contracts/base/PolicyManager.sol#L31

```
    modifier onlyTrader(uint256 policyId) {
        if (!_activePolicyIds.contains(policyId)) revert
Errors.PolicyNotExist(policyId);
        if (!isTrader(0, _msgSender()) && !isTrader(policyId, _msgSender())) {
            revert Errors.Unauthorized(_msgSender(), "TRADER");
        }
        _;
    }
```

**Solution**

N/A

**Status**

Acknowledged; After communicating with the project team, the project team said this policy pertains to a special group within the wallet, created during initialization, and cannot be removed. Traders in this group function as "super traders", allowing them to trade under any policy group. Whitelisted addresses in this group act as a "global whitelist", enabling their use across all policy groups.

### [N3] [Suggestion] Missing event records

**Category: Others**

**Content**

In the PolicyManager contract, the owner or admin can modify maxSpendingLimit and dailyVolumeSpent through the setMaxSpendingLimit, reduceSpendingLimit, and resetDailySpent functions, but no event logging is performed.

Code location: contracts/base/PolicyManager.sol#L175-L196

```
    function setMaxSpendingLimit(uint256 policyId, uint256 maxSpendingLimit) public
onlyOwner {
        ...
        _policyMap[policyId].maxSpendingLimit = maxSpendingLimit;
    }

    function reduceSpendingLimit(uint256 policyId, uint256 maxSpendingLimit) public
onlyAdminOrOwner {
        ...
        _policyMap[policyId].maxSpendingLimit = maxSpendingLimit;
    }

    function resetDailySpent(uint256 policyId) public onlyOwner {
        _policyMap[policyId].dailyVolumeSpent = 0;
    }
```

**Solution**

It is recommended to log events when modifying sensitive contract parameters to facilitate subsequent self-inspection or community auditing.

**Status**

Fixed

### [N4] [Suggestion] Repeat initialization event items

**Category: Design Logic Audit**

**Content**

In the PolicyManagerEventUtils library, the emitRemovedTrader function is used to log the event of removing a trader. When logging the policyId, the initItems operation is performed twice on uintItems, which is redundant.

Code location: contracts/base/PolicyManagerEventUtils.sol#L43

```
    function emitRemovedTrader(EventEmitter eventEmitter, address profile, uint256
_policyId, address _trader)
        external
    {
        EventUtils.EventLogData memory eventData;
        eventData.uintItems.initItems(1);
        eventData.uintItems.initItems(1);
```

```
        ...
    }
```

**Solution**

It is recommended to perform the initItems operation only once.

**Status**

Fixed

### [N5] [Medium] Potential risks of signature malleability

**Category: Authority Control Vulnerability Audit**

**Content**

In the SignatureValidator contract, the `_recoverSigner` function is used to recover the signer from the signature data. It only performs the recovery through ecrecover and does not check the s value of the signature. It is important to note that EIP2 still allows for signature malleability, so not checking the s value may lead to malleability risks.

Code location: contracts/common/SignatureValidator.sol#L46

```
    function _recoverSigner(bytes32 _hash, bytes memory _signature) internal pure
returns (address signer) {
        if (_signature.length != 65) revert Errors.InvalidLength();

        // Variables are not scoped in Solidity.
        bytes32 r;
        bytes32 s;
        uint8 v;
        /* solhint-disable no-inline-assembly */
        assembly {
            r := mload(add(_signature, 0x20))
            s := mload(add(_signature, 0x40))
            v := and(mload(add(_signature, 0x41)), 0xff)
        }

        // Recover ECDSA signer
        signer = ecrecover(_hash, v, r, s);

        // // Prevent signer from being 0x0
        // require(
        //     signer != address(0x0),
        //     "INVALID_SIGNER"
```

```
        // );

        return signer;
    }
```

**Solution**

It is recommended to check the validity of the s value before performing the ecrecover operation or to use the OpenZeppelin ECDSA library for signature data recovery.

**Status**

Fixed

### [N6] [Low] Potential risks of not resetting the isKnightSafe status

**Category: Design Logic Audit**

**Content**

In the ControlCenter contract, the owner can update the `_priceFeed` address through the setPriceFeed function and set the isKnightSafe of the new priceFeed to true, so that the priceFeed contract can trigger events through EventEmitter. However, when the `_priceFeed` address changes, the isKnightSafe status of the old priceFeed is not reset. If there is a risk of unauthorized access to the priceFeed, it may lead to malicious triggering of events. Although the owner can reset the isKnightSafe status through the disableActiveAccount function, it may be too late.

Code location: contracts/controlCenter/ControlCenter.sol#L188

```
    function setPriceFeed(address priceFeed) public onlyOwner {
        _priceFeed = priceFeed;
        isKnightSafe[priceFeed] = true;

        ControlCenterEventUtils.emitSetPriceFeed(this, priceFeed);
    }
```

**Solution**

It is recommended to set the isKnightSafe status of the old priceFeed to false when changing the `_priceFeed` address through the setPriceFeed function.

**Status**

Fixed

### [N7] [Medium] No check on Oracle price validity

**Category: Design Logic Audit**

**Content**

In the ChainlinkPriceFeed contract, users can obtain prices from the Chainlink oracle through the getChainlinkDataFeedLatestAnswer function, but there is no further check on the price returned by the oracle. It is important to note that answer is of type int256, and its value may be less than 0. Failing to check this may result in obtaining unexpected prices. updatedAt is the update time of the oracle price. Once the price deviation exceeds the threshold or the heartbeat time is reached, the price will be updated. Failing to check this will make it impossible to ensure that the obtained price is valid in real-time. Moreover, if the protocol is deployed on L2, the availability of the sequencer will affect the validity of the price. Therefore, when obtaining prices from Chainlink on L2, it is necessary to check the availability of the L2 sequencer. Fortunately, Chainlink provides an interface for this check.

Code location: contracts/priceFeed/ChainlinkPriceFeed.sol#L124

```
    function getChainlinkDataFeedLatestAnswer(address token) public view returns
(uint256 amt, uint8 decimals) {
        address priceFeed = _priceFeedMap[token];
        if (priceFeed == address(0)) {
            // revert Errors.IsNullValue();
            return (0, 0);
        }

        (, int256 answer,,,) = AggregatorV2V3Interface(priceFeed).latestRoundData();
        ...
    }
```

**Solution**

It is recommended to check that answer must be greater than 0 when obtaining prices, check that the difference between updatedAt and the current time must be less than its heartbeat interval, and when obtaining prices on L2, the availability of the sequencer must be checked through the Chainlink sequencerUptimeFeed.

**Status**

Fixed

### [N8] [Suggestion] Redundant price return value processing

**Category: Others**

**Content**

In the ChainlinkPriceFeed contract, the getChainlinkDataFeedLatestAnswer function is used to obtain token prices from the oracle. The price return value answer is added to the value of the amt variable as the final return value of the function. However, there is no logic in this function to obtain multiple prices for addition, so the addition operation of the amt parameter is redundant.

Code location: contracts/priceFeed/ChainlinkPriceFeed.sol#L127

```
    function getChainlinkDataFeedLatestAnswer(address token) public view returns
(uint256 amt, uint8 decimals) {
        ...
        amt += uint256(answer);
    }
```

**Solution**

It is recommended to directly use the answer value as the return value to save gas.

**Status**

Fixed

### [N9] [Low] Incorrect event logging

**Category: Others**

**Content**

In the PriceFeedEventUtils library, the emitSetPriceFeed function is used to log the set price feed event. When the owner performs batch setting, the event should record the number of token oracles that have been changed, but it incorrectly records the `length` as 1.

Code location: contracts/priceFeed/PriceFeedEventUtils.sol#L28-L29

```
    function emitSetPriceFeed(address eventEmitter, address[] memory token, address[]
memory priceFeed) external {
        EventUtils.EventLogData memory eventData;
        eventData.uintItems.initItems(token.length);
        eventData.uintItems.setItem(0, "length", 1);
        ...
    }
```

**Solution**

It is recommended that the length of initItems be 1, and when setItem is called, the `length` should be recorded as `token.length`.

**Status**

Fixed

### [N10] [Information] Any trader can reject a transaction

**Category: Authority Control Vulnerability Audit**

**Content**

In the TransactionRequest contract, a trader can reject a transaction with a specified `reqId` using the rejectTransactionByReqId function. It is important to note that the contract checks whether the caller is on the allowlist of the specified `onBehalfOfPolicyId` and verifies that the transaction data is allowed under this policy. However, it does not check whether the `policyId` of the transaction to be rejected is also `onBehalfOfPolicyId`. This means that if the data allowlist of policy A includes the data allowlist of another policy B, then the trader of policy A can arbitrarily reject all transactions proposed by traders in policy B. This does not conform to the rules of permission separation and poses the risk of unauthorized operation on transactions belonging to other policies.

Code location: contracts/transaction/TransactionRequest.sol#L159-L169

```
    function rejectTransactionByReqId(uint256 onBehalfOfPolicyId, bool
useGlobalWhitelist, uint256 reqId)
        public
        onlyTrader(onBehalfOfPolicyId)
    {
        Transaction.Request memory request = getTransactionRequest(reqId);

        _updateTransactionRequestStatus(reqId, Transaction.Status.Rejected);
```

```
        validateTradingAccess(onBehalfOfPolicyId, useGlobalWhitelist,
request.params.to, request.params.data);

        TransactionEventUtils.emitRejectedTransactionRequest(_controlCenter,
address(this), reqId);
    }
```

**Solution**

It is recommended to check whether the `onBehalfOfPolicyId` passed in by the user is the same as the `policyId` of the transaction during the rejectTransactionByReqId operation. Alternatively, only allow the owner role to perform rejection operations.

**Status**

Acknowledged; After communicating with the project team, the project team said that this is the expected design, and if the transaction can be executed by global policy or policy group. All traders under policy group can reject the transaction.

### [N11] [Information] Potential risks of misusing useGlobalWhitelist

**Category: Authority Control Vulnerability Audit**

**Content**

In the TransactionRequest contract, when a trader performs rejectTransactionByReqId, executeTransaction, or executeTransactionByReqId operations, it involves the use of the useGlobalWhitelist variable. When the useGlobalWhitelist passed in by the user is true, the validateTradingAccess, validatePolicyLimit, and _updateVolumeSpent functions will use policy 0 for necessary checks and spent amount updates. It is important to note that during these operations, `msg.sender` may be a trader who does not belong to policy 0. In the onlyTrader modifier, as long as `msg.sender` belongs to the trader of the specified `onBehalfOfPolicyId`, it can pass the check and perform data checks and spend updates as the GlobalWhitelist. This leads to the risk of traders not in policy 0 having unauthorized access to policy 0.

Code location: contracts/transaction/TransactionRequest.sol#L172C1-L196C6

```
    function rejectTransactionByReqId(uint256 onBehalfOfPolicyId, bool
useGlobalWhitelist, uint256 reqId)
        public
        onlyTrader(onBehalfOfPolicyId)
```

```
    {
        ...
    }

    function executeTransaction(
        uint256 onBehalfOfPolicyId,
        bool useGlobalWhitelist,
        address to,
        uint256 value,
        bytes memory data
    ) public {
        ...
    }

    function executeTransactionByReqId(uint256 onBehalfOfPolicyId, bool
useGlobalWhitelist, uint256 reqId) public {
        ...
    }
```

**Solution**

If this is not the intended design, it is recommended to only allow `msg.sender` to operate on data within their own policy.

**Status**

Acknowledged; This is the intended design. All policy groups can inherit and execute transactions from the global policy group.

### [N12] [Information] Transaction processing issue

**Category: Authority Control Vulnerability Audit**

**Content**

In the TransactionRequest contract, whitelisted traders can submit transactions through the requestTransaction function and then execute them using the executeTransactionByReqId function. Traders can also cancel or reject transactions using the cancelTransactionByReqId and rejectTransactionByReqId functions. It is important to note that traders can also execute transactions directly through the executeTransaction function without first performing a requestTransaction operation. This means that the protocol allows traders to execute transactions directly while also allowing them to submit transactions first through the requestTransaction function and then execute them, without any time lock restrictions between transaction submission and execution. This makes the

transaction processing flow quite chaotic and lacks hierarchical design and necessary vertical permission control.

Code location: contracts/transaction/TransactionRequest.sol#L172

```
function executeTransaction(
    uint256 onBehalfOfPolicyId,
    bool useGlobalWhitelist,
    address to,
    uint256 value,
    bytes memory data
) public {
    ...
}
```

**Solution**

It is recommended to clarify the expected design. If it is not the intended design, consider only allowing the owner to execute transactions directly, while the trader role submits transactions through the requestTransaction function first and then executes them after a time lock.

**Status**

Acknowledged; This is the intended design. Traders can directly execute transactions that belong to their own policy group, and transactions submitted by other traders can only be executed by traders of the policy group to which the transaction belongs.

### [N13] [High] Incorrect address parsing in emergencyEtherTransfer

**Category: Design Logic Audit**

**Content**

In the AaveUtilsAnalyser contract, the extractAddressesWithValue function is used to parse some functions of the AAVE protocol. For the emergencyEtherTransfer function, it sets the to address in `addrList[0]` and the transferred ETH token amount in `valueList[0]`. This is inappropriate because the to address is not a token address but a recipient address, which makes it impossible for the protocol to calculate the value of the transferred tokens using the getTransactionVolume function.

Code location: contracts/knightSafeAnalyser/AaveUtilsAnalyser.sol#L31

```
    function extractAddressesWithValue(address to, bytes calldata data)
        external
        view
        override
        returns (address[] memory addrList, uint256[] memory valueList)
    {
        bytes4 selector = getSelector(data);

        if (
            selector == 0xeed88b8d //
emergencyEtherTransfer(to:address,amount:uint256)
        ) {
            addrList = new address[](1);
            valueList = new uint256[](addrList.length);
            addrList[0] = _getAddressFromBytes(data, 0); // to
            valueList[0] = _getUintFromBytes(data, 1); // amount
            return (addrList, valueList);
        }
        ...
    }
```

**Solution**

It is recommended to set nativeToken as `addrList[0]` instead of the to address when handling the emergencyEtherTransfer function.

**Status**

Fixed

### [N14] [High] Wrong parsing of onBehalfOf position in depositETH

**Category: Design Logic Audit**

**Content**

In the AaveUtilsAnalyser contract, when processing the depositETH function of the AAVE protocol, `addrList[1]` should theoretically be assigned the onBehalfOf address. It obtains the value at position 0 as the onBehalfOf address through `_getAddressFromBytes(data, 0)`, but in reality, the onBehalfOf address is at position 1. This causes `addrList[1]` to store an unexpected address.

Code location: contracts/knightSafeAnalyser/AaveUtilsAnalyser.sol#L61

```
    function extractAddressesWithValue(address to, bytes calldata data)
        external
        view
        override
        returns (address[] memory addrList, uint256[] memory valueList)
    {
        ...
        } else if (
            selector == 0x474cf53d //
depositETH(:address,onBehalfOf:address,referralCode:uint16)
        ) {
            addrList = new address[](2);
            valueList = new uint256[](addrList.length);
            addrList[0] = nativeToken; // ETH
            addrList[1] = _getAddressFromBytes(data, 0); // onBehalfOf
            ...
        }
        ...
    }
```

**Solution**

It is recommended to retrieve onBehalfOf from position 1 and store it in `addrList[1]`.

**Status**

Fixed

### [N15] [Critical] Risk of not being able to perform analysis with GMXAnalyser

**Category: Design Logic Audit**

**Content**

The GMXAnalyser contract is used to parse functions in the GMX protocol. External contracts call the extractAddressesWithValue function and pass in the necessary parameters for specific parsing. It matches function signatures using if statements for parameter handling. Unfortunately, after each if statement is processed, addrList and valueList are not returned. This causes the extractAddressesWithValue function to inevitably execute the final revert statement, preventing users from successfully obtaining the GMX protocol parsing through the extractAddressesWithValue function.

Code location: contracts/knightSafeAnalyser/GmxAnalyser.sol#L79

```
    function extractAddressesWithValue(address to, bytes calldata data)
        external
        view
        override
        returns (address[] memory addrList, uint256[] memory valueList)
    {
        ...
        revert UnsupportedCommand();
    }
```

**Solution**

It is recommended to return addrList and valueList after each if statement is processed.

**Status**

Fixed

### [N16] [Suggestion] Redundant GMX v1 selectors

**Category: Others**

**Content**

In the extractAddressesWithValue function of the GMXAnalyser contract, only the functions of GMX v2 are parsed, while the selectors for GMX v1 defined in the contract are not handled, which is redundant.

Code location: contracts/knightSafeAnalyser/GmxAnalyser.sol#L252

```
    /// GMX V1
    //
decreasePositionAndSwap(_path:address[],_indexToken:address,_collateralDelta:uint256,_
sizeDelta:uint256,_isLong:bool,
    // _receiver:address,_price:uint256,_minOut:uint256)
    bytes4 internal constant DECREASE_POSITION_AND_SWAP = 0x5fc8500e;
    //
increasePosition(_path:address[],_indexToken:address,_amountIn:uint256,_minOut:uint256
,_sizeDelta:uint256,_isLong:bool,_price:uint256)
    bytes4 internal constant INCREASE_POSITION = 0xb7ddc992;
    ...
```

**Solution**

If this is not the intended design, it is recommended to remove the redundant function selectors.

**Status**

Fixed

### [N17] [Suggestion] Potential risks of not being able to reset the Gov address

**Category: Design Logic Audit**

**Content**

In the constructor of the UniswapAnalyser contract, the deployer is set as the gov role. It is important to note that there are no other functions in the contract that can modify the gov parameter value, which means that the gov address will default to the deployer address and cannot be changed. If the protocol launches a governance module in the future, the UniswapAnalyser will not be able to modify the gov address to the actual governance address.

Code location: contracts/knightSafeAnalyser/UniswapAnalyser.sol#L17

```
    constructor(address nativeToken_) {
        gov = msg.sender;
        ...
    }
```

**Solution**

It is recommended to add a new function to modify the gov address.

**Status**

Fixed

### [N18] [Critical] Incorrect Gov Address Check

**Category: Authority Control Vulnerability Audit**

**Content**

In the UniswapAnalyser contract, the updateMaxFeeBips function is used to modify the maxFeeBips parameter. Theoretically, the updateMaxFeeBips function should only allow the gov role to call it, but the function performs an incorrect check. If the caller is the gov role, the call will revert.

Code location: contracts/knightSafeAnalyser/UniswapAnalyser.sol#L23

```
function updateMaxFeeBips(uint256 newMaxFeeBips) public {
    if (gov == msg.sender) {
        revert Unauthorized("GOV");
    }
    ...
}
```

**Solution**

It is recommended to revert when the caller is not the gov role to avoid allowing other users to arbitrarily change the maxFeeBips parameter.

**Status**

Fixed

### [N19] [Critical] Incorrect PERMIT2_TRANSFER_FROM_BATCH parameter handling in UniswapAnalyser

**Category: Design Logic Audit**

**Content**

The UniswapAnalyser contract is used to parse functions of the Uniswap protocol. When the command to be parsed is PERMIT2_TRANSFER_FROM_BATCH, the contract processes the to, token, and amount parameters in batchDetails through a for loop. Theoretically, the value corresponding to the to address should be 0, and the value corresponding to the token should be the amount. Unfortunately, the contract incorrectly scrambles the positions of the data during processing.

It first places the `to[0]` address in `addrList[0]` using `x++`. After processing `to[0]`, x will be 1, so `valueList[1]` will be set to 0. In reality, `valueList[1]` should be set to `amount[0]` to correspond with `token[0]`.

Code location: contracts/knightSafeAnalyser/UniswapAnalyser.sol#L431-L436

```
function _dispatch(bytes1 commandType, bytes calldata inputs)
    private
    view
    returns (address[] memory addrList, uint256[] memory valueList, uint256 bips)
{
    ...
    } else if (command == Commands.PERMIT2_TRANSFER_FROM_BATCH) {
```

```
            ...
            uint256 x = 0;
            for (uint256 i = 0; i < batchDetails.length; i++) {
                addrList[x++] = batchDetails[i].to;
                valueList[x] = 0;
                addrList[x++] = batchDetails[i].token;
                valueList[x] = batchDetails[i].amount;
            }
            return (addrList, valueList, bips);
        }

        ...
    }
```

**Solution**

It is recommended to modify the address sorting method. Alternatively, consider processing the to addresses and tokens separately and then merging them into a single list at the end.

**Status**

Fixed

### [N20] [Medium] Potential risks of proxy deployment

**Category: Design Logic Audit**

**Content**

In the KnightSafeProxyFactory contract, the createProxy function is used to deploy new KnightSafeProxy contracts. Users can participate in the contract creation by passing in a saltNonce value. It is important to note that the contract creation only depends on the user-provided saltNonce and the creationCode. If the saltNonce value provided by the user has already been used, the creation of KnightSafeProxy will fail, which may cause some inconvenience for users. If KnightSafeProxy is deployed on multiple chains in the future, other users may be able to occupy these addresses for fraudulent activities.

Code location: contracts/proxies/KnightSafeProxyFactory.sol#L62

```
function createProxy(address implementation, bytes memory data, uint256 saltNonce)
    public
    returns (KnightSafeProxy proxy)
{
    if (implementation == address(0)) revert Errors.IsNullValue();
    if (!CONTROL_CENTER.isOfficialImplementation(implementation)) {
```

```
        revert Errors.AddressIsNotKnightSafeImplementation(implementation);
    }
    bytes32 salt = keccak256(abi.encodePacked(saltNonce));
    proxy = _deployProxy(implementation, data, salt);

    ProxyFactoryEventUtils.emitProxyCreation(CONTROL_CENTER, implementation,
address(proxy), msg.sender, saltNonce);
}
```

**Solution**

It is recommended to include `msg.sender` as part of the salt when creating contracts to mitigate the above risks.

**Status**

Fixed

### [N21] [Low] Potential flaws in trading volume calculations

**Category: Design Logic Audit**

**Content**

In the protocol, the TransactionRequest contract is used to manage transactions performed by whitelisted traders and to check whether the trading volume of these transactions exceeds the limit. The calculation of trading volume mainly relies on oracle prices. If the transaction involves tokens not supported by the oracle, the calculated trading volume will be smaller than the actual amount. Traders may be able to exploit this flaw by using unknown tokens to bypass the trading volume check.

Code location:

contracts/priceFeed/ChainlinkPriceFeed.sol#L99

contracts/priceFeed/ChainlinkPriceFeed.sol#L121

```
    function getTransactionVolume(address[] memory contractAddresses, uint256[]
memory amounts)
        external
        view
        returns (uint256)
    {
        ...
            if (success && returnData.length == 32) {
                tokenDecimals = abi.decode(returnData, (uint8));
```

```
            } else {
                continue;
            }

            ...
    }


    function getChainlinkDataFeedLatestAnswer(address token) public view returns
(uint256 amt, uint8 decimals) {
        address priceFeed = _priceFeedMap[token];
        if (priceFeed == address(0)) {
            // revert Errors.IsNullValue();
            return (0, 0);
        }
        ...
    }
```

**Solution**

It is recommended to prohibit transactions involving tokens not supported by the oracle, but this would greatly limit the flexibility of the protocol and user experience. Alternatively, consider having the contract owner manually manage the trading volume of tokens not supported by the oracle.

**Status**

Acknowledged; After communicating with the project team, the project team said that in the long term, we plan to develop our own volume checker. For the current version utilizing Chainlink, we will disclose the tokens supported for volume calculation. Users may interact with tokens not supported by the oracle, but the corresponding volume will not be included in the calculations.

# 5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
| --- | --- | --- | --- |
| 0X002407030002 | SlowMist Security Team | 2024.06.24 - 2024.07.03 | 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 3 critical risks, 2 high risks, 3 medium risks, 3 low risks, 6 suggestions,

and 4 information. All the findings were fixed or acknowledged.

# 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