Smart Contract Audit Report

Filswan Smart Contract

9 Dec 2022

Numen Cyber Labs - Security Services

Table of Content

Numen Cyber

1 Executive Summary

Numen Cyber Technology was engaged by Filswan to review smart contract implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

Five high severities findings are related to DAO_Role authority, Business Issues and Oracle Issues. In addition, there is also 1 Informational finding.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

Numen Cyber

Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood and impact are categorized into three ratings: High, Medium and Low. Severity is determined by likelihood and impact and can be classified into four categories accordingly, Critical, High, Medium, Low shown in table 1.1.

Table 1.1: Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.

- Code and business security testing: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

- Additional Recommendations: We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

Category	Assessment Item

Numen Cyber

Basic Coding Assessment	Apply Verification Control
	Authorization Access Control
	Forged Transfer Vulnerability
	Forged Transfer Notification
	Numeric Overflow
	Transaction Rollback Attack
	Transaction Block Stuffing Attack
	Soft fail Attack
	Hard fail Attack
	Abnormal Memo
	Abnormal Resource Consumption
	Secure Random Number
Advanced Source Code Scrutiny	Asset Security
	Cryptography Security
	Business Logic Review
	Source Code Functional Verification
	Account Authorization Control
	Sensitive Information Disclosure
	Circuit Breaker

Numen Cyber

	Blacklist Control
	System API Call Analysis
	Contract Deployment Consistency Check
Additional Recommendations	Semantic Consistency Checks
	Following Other Best Practices

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

Numen Cyber

2 Findings Overview

2.1 Project Info and Contract Address

Project Name: Filswan

Project URL: https://mcs.filswan.com/

Audit Time: 2022/10.31 - 2022/12.9

Language: solidity

Source Code Link	Commit Hash
https://github.com/filswan/multi-chain-storage/tree/main/on-chain	32c445386a1e154dc0b99d130a922a742b78c74c

2.2 Summary

Severity	Found	
Critical	0	
High	5	
Medium	0	
Low	0	
Informational	1	

2.3 Key Findings

Five high severities findings are related to DAO_Role authority, Business Issues and Oracle Issues. In addition, there is also 1 Informational finding.

ID	Severity	Findings Title	Status	Confirm
NVE-001	High	DAO_Role vote verification	Fixed	Confirmed
NVE-002	High	Function parameter pass-in security	Ignored	Confirmed
NVE-003	High	LockFee Fee Calculation	Fixed	Confirmed
NVE-004	High	Vulnerability of refund function	Fixed	Confirmed
NVE-005	High	Data source information acquisition	Fixed	Confirmed
NVE-006	Informational	Out of gas	Ignored	Confirmed

Table 2.1: Key Audit Findings

Numen Cyber

3 Detailed Description of Findings

3.1 DAO_Role vote verification

ID: NVE-001

Location: FilswanOracleV2.sol

Severity: High

Category: Authority Issues

Likelihood: High

Impact: High

Description:

As shown in Figures 1 and 2 below, the users with DAO_Role permissions can call the signCarTransaction function to vote. According to the design of the project party, at least 3 of 4 DAO_Role users required vote to pass. However, one signle user with DAO_Role permission can repeatedly vote to reach the threshold by calling the signCarTransaction function and the signHash function, which cause a serious permission security issue.

Numen Cyber

```
function signCarTransaction(
    string[] memory cidList,
    string memory dealId,
    string memory network,
    address recipient
) public onlyRole(DAO_ROLE) {
    string memory key = concatenate(dealId, network);

    require(
        txInfoMap[key][msg.sender].flag == false,
        "You already sign this transaction"
    );

    txInfoMap[key][msg.sender].recipient = recipient;
    txInfoMap[key][msg.sender].flag = true;
    txInfoMap[key][msg.sender].cidList = cidList;
    txInfoMap[key][msg.sender].signer = msg.sender;
    txInfoMap[key][msg.sender].timestamp = block.timestamp;
    txInfoMap[key][msg.sender].blockNumber = block.number;

    bytes32 voteKey = keccak256(
        abi.encodeWithSignature(
            "f(string,string,address,string[])",
            dealId,
            network,
            recipient,
            cidList
        )
    );
```

Figure 1 signCarTransaction function

```
function signHash(string memory dealId, string memory network, address recipient, bytes32 voteKey) public onlyRole(DAO_ROLE) {
    string memory key = concatenate(dealId, network);

    txVoteMap[voteKey] = txVoteMap[voteKey] + 1;
    if(txInfoMap[key][msg.sender].signStatus == 0){
        if (txVoteMap[voteKey] >= _threshold
        && _filinkAddress != address(0)
        ) {
            cidListMap[key] = txInfoMap[key][msg.sender].cidList;
            FilinkConsumer(_filinkAddress).requestDealInfo(dealId, network);
        }
    }
    // todo: add check total count of cid list and do chianlink requestDealInfo
    emit SignHash(dealId, network, recipient, voteKey);
}

function f(string memory s1,string memory s2,address a1,string[] calldata sa) public{
```

Figure 2 signHash function

Recommendations:

Numen Cyber Lab recommends to delete the signHash function or modify the function logic.

Result: Pass

Fix Result:

Fixed

The fixed code is as follows:

- Deleted the signCarTransaction function.
- Modified the signhash function

```
function signHash(string memory dealId, string memory network, address recipient, bytes32 voteKey) public onlyRole(DAO_ROLE) {
    string memory key = concatenate(dealId, network);

    // a user CANNOT vote again
    require(!userVotedMap[voteKey][msg.sender], 'you already signed this hash');
    userVotedMap[voteKey][msg.sender] = true;
    txVoteMap[voteKey] = txVoteMap[voteKey] + 1;

    // if all batches are signed
    if(txInfoMap[key][msg.sender].signStatus == 0){
        if (txVoteMap[voteKey] >= _threshold
        && _filinkAddress != address(0) &&
        voteKeyCidListMap[voteKey].length > 0
        ) {
            cidListMap[key] = voteKeyCidListMap[voteKey];
            FilinkConsumer(_filinkAddress).requestDealInfo(dealId, network);
        }
    }
    // todo: add check total count of cid list and do chianlink requestDealInfo
```

Numen Cyber

3.2 FUNCTION PARAMETER PASS-IN SECURITY

ID: NVE-002 Location:SwanPayment.sol

Severity: High Category: Business Issues

Likelihood: High

Impact: High

Description:

As shown in Figure 3 below, when an user calls the lockTokenPayment function, he can structure the parameters to bypass the “require” judgement in the contract and execute the function. This will cause exceptions when voting for transaction.

```
function lockTokenPayment(lockPaymentParam calldata param)
    public
    override
    returns (bool)
{
    require(
        !txMap[param.id]._isExisted && !txCarMap[param.id]._isExisted,
        "Payment of transaction is already locked"
    );

    require(
        param.minPayment > 0 && param.amount > param.minPayment,
        "payment should greater than min payment"
    );

    require(
        IERC20(_ERC20_TOKEN).allowance(msg.sender, address(this)) >=
            param.amount,
        "please approve spending token"
    );
    IERC20(_ERC20_TOKEN).transferFrom(
        msg.sender,
        address(this),
        param.amount
    );
```

Figure 3 lockTokenPayment function

Recommendations:

Numen Cyber Lab recommends to modify the code logic.

Result: Pass

Fix Result:

Ignore(After communicating with the project party, it will be validated in the backend and will not vote on invalid transactions).

3.3 LockFee Fee Calculation

ID: NVE-003 Location:SwanPayment.sol

Severity: High Category: Business Issues

Likelihood: High

Impact: High

Description:

As shown in Figure 4 below, the project party will fail to withdraw the storage fee for the specified dealId while the user does not transfer enough amount or the FIL price has significant floating in a short period of time.

Numen Cyber

```
if (serviceCost > 0) {
    tokenAmount = IPriceFeed(_priceFeed).consult(
        _ERC20_TOKEN,
        serviceCost
    );
    uint256 size = 0;
    for (uint8 i = 0; i < cidList.length; i++) {
        TxInfo storage t = txCarMap[cidList[i]];
        if (!t._isExisted) {
            continue;
        } else {
            size += t.size;
        }
    }

    require(size > 0, "file size should be greater than 0");

    uint256 unitPrice = tokenAmount / size;
    for (uint8 i = 0; i < cidList.length; i++) {
        TxInfo storage t = txCarMap[cidList[i]];
        if(t.copyLimit == 0) continue;
        uint256 cost = unitPrice * t.size;

        t.lockedFee = t.lockedFee - cost;
        t.copyLimit = t.copyLimit - 1;
        if (t.lockedFee < 0) {
            t.lockedFee = 0;
        }
        t._isExisted = (t.lockedFee > 0);
    }
```

Figure 4 Part of code of unlockCarPayment function

Recommendations:

Numen Cyber Lab recommends to modify the code logic.

Result: Pass

Fix Result:

Fixed(After communicating with the project party, under certain circumstances, when the user Lockfee is insufficient, the excess fee will be borne by the project party).

The fixed code is as follows:

```
uint256 unitPrice = tokenAmount / size;
for (uint8 i = 0; i < cidList.length; i++) {
    TxInfo storage t = txCarMap[cidList[i]];
    if(t.copyLimit == 0) continue;
    uint256 cost = unitPrice * t.size;

    if (t.lockedFee < cost) {
        t.lockedFee = 0;
    } else {
        t.lockedFee = t.lockedFee - cost;
    }
    t.copyLimit = t.copyLimit - 1;

    t._isExisted = (t.lockedFee > 0);
}
```

3.4 VULNERABILITY OF REFUND FUNCTION

ID: NVE-004

Location: SwanPayment.sol

Severity: High

Category: Business Issues

Likelihood: High

Impact: High

Description:

As shown in Figure 5 below, the project party will fail to withdraw the storage fee while the user withdraws the storage fee advance, in the case that user has submitted the storage request and the Dao_Role has finished vote processing.

```
function refund(string[] memory cidList) public {
    for (uint8 i = 0; i < cidList.length; i++) {
        TxInfo storage t = txCarMap[cidList[i]];
        if (t._isExisted) {
            t._isExisted = false;
            if (t.lockedFee > 0) {
                IERC20(_ERC20_TOKEN).transfer(t.owner, t.lockedFee);
                emit Refund(cidList[i], t.owner, t.lockedFee);
            }
```

Figure 5 refund function

Cyber

Result: Pass

Fix Result:

Fixed

The fixed code is as follows:

```
function refund(string[] memory cidList) public {
    for (uint8 i = 0; i < cidList.length; i++) {
        TxInfo storage t = txCarMap[cidList[i]];
        if (t._isExisted && block.timestamp > t.deadline) {
            t._isExisted = false;
            if (t.lockedFee > 0) {
                IERC20(_ERC20_TOKEN).transfer(t.owner, t.lockedFee);
                emit Refund(cidList[i], t.owner, t.lockedFee);
            }
        }
    }
}
```

3.5 Data source information acquisition

ID: NVE-005

Location: FilinkConsumer.sol

Severity: High

Category: Oracle Issues

Likelihood: High

Impact: High

Description:

As shown in Figure 6 below, the storage price during contract proceeding is related to external data source, which is using HTTP protocol. The project party might encounter data source security issues in data pragmaticality, data security and data accuracy.

```
function requestDealInfo(string calldata deal, string calldata network) public returns (bytes32 requestId) {
    require(mapDealPrice[deal] == 0, "deal price is already on-chain, call getPrice(deal)");

    Chainlink.Request memory request = buildChainlinkRequest(jobId, address(this), this.fulfill.selector);

    // <deal>?network=<network>
    string memory tmp = concatenate(deal, "?network=");
    string memory params = concatenate(tmp, network);

    string memory key = concatenate(deal, network);

    /**
     * GET http://35.168.51.2:7886/deal/<deal>?network=<network>
     * ex. GET http://35.168.51.2:7886/deal/123456?network=filecoin_calibration, data.deal.storage_price = 8294400600825600
     */
    request.add("get", concatenate("http://35.168.51.2:7886/deal/", params));
    request.add("path", "data,deal,storage_price");
    request.addInt('times', 1);

    bytes32 id = sendChainlinkRequestTo(oracle, request, fee);
    mapRequestDeal[id] = key;

    return id;
```

Figure 6 requestDealInfo function

Result: Pass

Fix Result:

fixed(After communicating with the project party, they will ensure the safety of information from external data sources).

The fixed code is as follows:

```
function requestDealInfo(string calldata deal, string calldata network) public returns (bytes32 requestId) {
    require(mapDealPrice[deal] == 0, "deal price is already on-chain, call getPrice(deal)");

    Chainlink.Request memory request = buildChainlinkRequest(jobId, address(this), this.fulfill.selector);

    // <deal>?network=<network>
    string memory tmp = concatenate(deal, "?network=");
    string memory params = concatenate(tmp, network);

    string memory key = concatenate(deal, network);

    /**
     * GET https://flink-adapter.filswan.com/deal/<deal>?network=<network>
     * ex. GET https://flink-adapter.filswan.com/deal/123456?network=filecoin_mainnet, data.deal.storage_price = 42855481110
     */
    request.add("get", concatenate("https://flink-adapter.filswan.com/deal/", params));
    request.add("path", "data,deal,storage_price");
    request.addInt('times', 1);

    bytes32 id = sendChainlinkRequestTo(oracle, request, fee);
    mapRequestDeal[id] = key;

    return id;
}
```

3.6 Out of gas

ID: NVE-001

Location: FilswanOracleV2.sol

Severity: Informational

Category: Business Issues

Likelihood: Informational

Impact: Informational

Description:

As shown in Figure 7 below, when DAO_Role calls sign to vote, if the incoming cidList is too long, it will cause insufficient gas.

Numen Cy

```
function sign(string memory dealId, string memory network, string[] memory cidList, uint8 batchNo) public onlyRole(DAO_ROLE) {

    string memory key = concatenate(dealId, network);

    require(txInfoMap[key][msg.sender].flag, "no presign");
    require(txInfoMap[key][msg.sender].batch > batchNo, "wrong batch No");

    uint256 bitStatus = 1<<batchNo;//2

    require((bitStatus & txInfoMap[key][msg.sender].signStatus) == bitStatus, "already signed the batch");//2&7

    txInfoMap[key][msg.sender].signStatus = txInfoMap[key][msg.sender].signStatus ^ bitStatus;

    txInfoMap[key][msg.sender].batchCidList[batchNo] = cidList;

    if(txInfoMap[key][msg.sender].signStatus == 0){ // all signs are done.

        for(uint i = 0; i < txInfoMap[key][msg.sender].batch; i++){
            for(uint j = 0; j < txInfoMap[key][msg.sender].batchCidList[i].length; j++){
                // todo: add existed check?
                if(!cidMap[key][msg.sender][txInfoMap[key][msg.sender].batchCidList[i][j]]){
                    cidMap[key][msg.sender][txInfoMap[key][msg.sender].batchCidList[i][j]] = true;
                    txInfoMap[key][msg.sender].cidList.push(txInfoMap[key][msg.sender].batchCidList[i][j]);
                }
                // txInfoMap[key][msg.sender].cidList.push(txInfoMap[key][msg.sender].batchCidList[i][j]);
```

Figure 7 sign function

Result: **Pass**

Fix Result:

Ignore(After communicating with the project party, the function is called for DAO_Role and will not pass in too long cidList).

Numen Cy

4 CONCLUSION

In this audit, we thoroughly analyzed **Filswan**’s smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been brought up to the project party, ignored issues are in line with the project design, and permissions are only used for the project to properly function. We therefore deem the audit result to be a **PASS.** To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

Numen Cyber

5 Appendix

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: Critical

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: Critical

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.6 soft fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.7 hard fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

5.1.8 Abnormal Memo Assessment

Numen Cyber

- Description: Assess whether there is abnormal memo vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: Critical

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: Medium

5.2.3 Malicious Code Behaviour

- Description: Examine whether sensitive behaviour present in the code
- Results: Not found
- Severity: Medium

5.2.4 Sensitive Information Disclosure

Numen Cyber

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

Numen Cyber

6 Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without Numen's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Numen to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Numen's position is that each company and individual are responsible for their own due diligence and continuous security. Numen's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Numen Cyber

References

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound). https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error. https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption. https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation. https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure. https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features. https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems. https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors. https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors. https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology. https://www.owasp.org/index.php/OWASP_Risk_ Rating_Methodology

Numen Cyber

Numen Cyber

Numen Cyber Technology Pte. Ltd.

11 North Buona Vista Drive, #04-09,

The Metropolis, Singapore 138589

Tel: 65-63555555

Fax: 65-63666666

Email: sales@numencyber.com

Web: https://numencyber.com