FYEO

Security Code Review Gateway Protocol EVM

Identity

July 2024
Version 1.0

Presented by:
FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

Table of Contents

Executive Summary

Overview

Identity engaged FYEO Inc. to perform a Security Code Review Gateway Protocol EVM.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on July 08 - July 15, 2024, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following issues have been identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- FYEO-IDENTITY-01 – Initializers are not disabled
- FYEO-IDENTITY-02 – Recreated network could be taken over
- FYEO-IDENTITY-03 – Flags may be improperly unset
- FYEO-IDENTITY-04 – Missing zero address checks
- FYEO-IDENTITY-05 – Network fees not checked during initialization
- FYEO-IDENTITY-06 – Admin modification is done in a single step
- FYEO-IDENTITY-07 – Code clarity
- FYEO-IDENTITY-08 – Fees can be changed to frontrun a select user
- FYEO-IDENTITY-09 – Tokens can be expired on mint

Based on our review process, we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

The FYEO Review Team performed a Security Code Review Gateway Protocol EVM. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a public repository at https://github.com/identity-com/gateway-protocol-evm with the commit hash 08c2b17870e49b994cded74dafbe79bce97565d5.

Commit hash of remediated code: 6d83b2e1a119c39cc95f79a98f8670edf69edbe1

Files included in the code review
```
gateway-protocol-evm/
└── smart-contract/
    └── contracts/
        ├── library/
        │   ├── BitMask.sol
        │   ├── Charge.sol
        │   ├── CommonErrors.sol
        │   └── InternalTokenApproval.sol
        ├── ChargeHandler.sol
        ├── FlagsStorage.sol
        ├── FlexibleNonceForwarder.sol
        ├── Gated.sol
        ├── GatedERC2771.sol
        ├── GatedERC2771Upgradeable.sol
        ├── Gatekeeper.sol
        ├── GatewayNetwork.sol
        ├── GatewayStaking.sol
        ├── GatewayToken.sol
        ├── MultiERC2771Context.sol
        ├── MultiERC2771ContextUpgradeable.sol
        ├── ParameterizedAccessControl.sol
        └── TokenBitMask.sol
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review Gateway Protocol EVM, we discovered:

- 2 findings with MEDIUM severity rating.
- 3 findings with LOW severity rating.
- 4 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

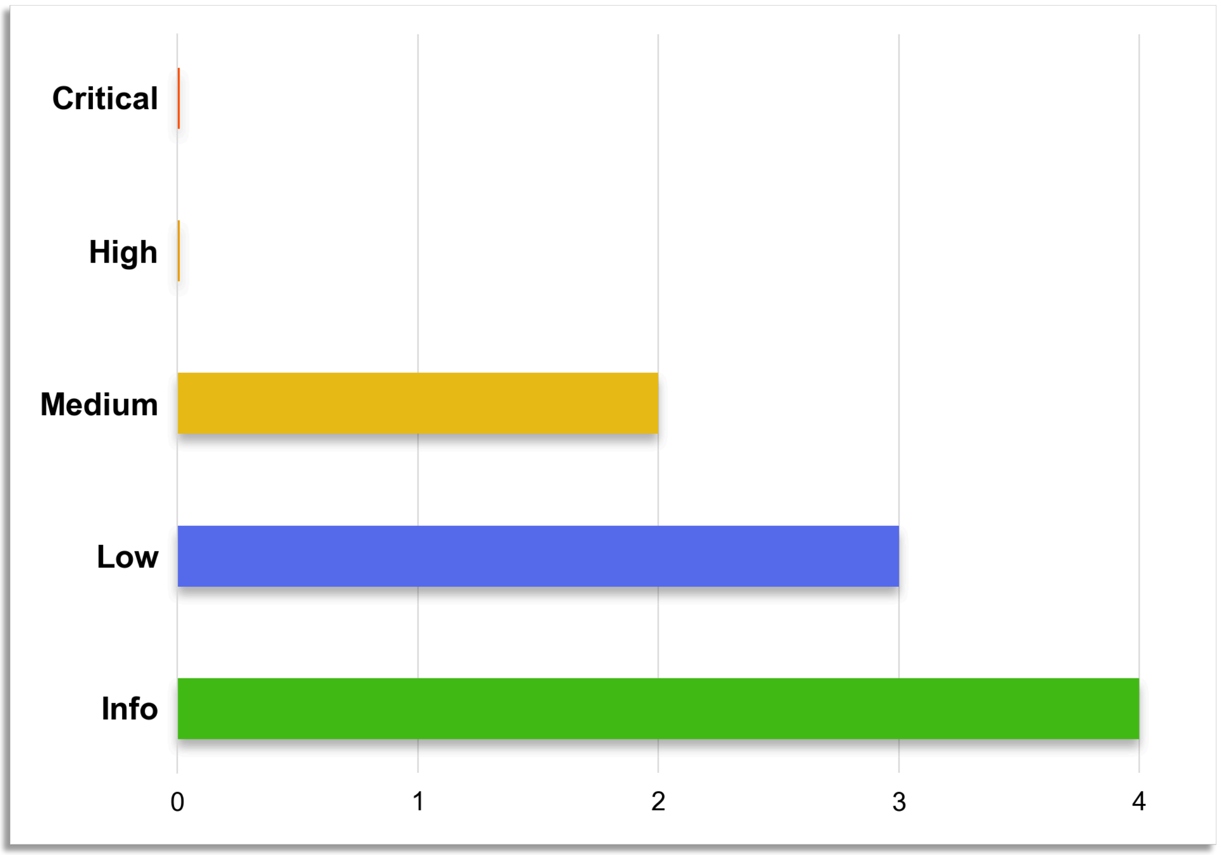

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description
FYEO-IDENTITY-01	**Medium**	Initializers are not disabled
FYEO-IDENTITY-02	**Medium**	Recreated network could be taken over
FYEO-IDENTITY-03	**Low**	Flags may be improperly unset
FYEO-IDENTITY-04	**Low**	Missing zero address checks
FYEO-IDENTITY-05	**Low**	Network fees not checked during initialization
FYEO-IDENTITY-06	**Informational**	Admin modification is done in a single step
FYEO-IDENTITY-07	**Informational**	Code clarity
FYEO-IDENTITY-08	**Informational**	Fees can be changed to frontrun a select user
FYEO-IDENTITY-09	**Informational**	Tokens can be expired on mint

Table 2: Findings Overview

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

Conclusion

Based on our review process, we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

The EVM Gateway Protocol is a standard that enables smart contracts to enforce access control by requiring users to possess a valid Gateway Token. Gateway Tokens are non-transferable, semi-fungible tokens issued by “Gatekeepers”, who verify user eligibility. These gatekeepers operate within a “Gatekeeper Network”, a collaborative group managing the issuance and governance of Gateway Tokens for specific purposes, such as verifying age or identity. The protocol integrates various Ethereum standards to support features like meta-transactions, upgradeability, and staking, ensuring flexible and secure token management. The Gateway Tokens can also be flagged for specific attributes, enhancing their functionality and interoperability across smart contracts.

During the audit, the team was highly responsive and provided prompt assistance with any questions that arose. Overall, the codebase is well-structured and complemented by good documentation, ensuring clarity and ease of understanding.

Initializers are not disabled

Finding ID: FYEO-IDENTITY-01
Severity: **Medium**
Status: **Remediated**

Description

The absence of the `_disableInitializers()` call in the constructor of the upgradeable GatewayNetwork and Gatekeeper contracts leaves them vulnerable. An uninitialized contract can be taken over by an attacker. This applies to both a proxy and its implementation contract, which may impact the proxy.

Proof of Issue

File name: smart-contract/contracts/GatewayNetwork.sol
File name: smart-contract/contracts/Gatekeeper.sol

```
contract GatewayNetwork is ParameterizedAccessControl, IGatewayNetwork, UUPSUpgradeable
and
contract Gatekeeper is ParameterizedAccessControl, IGatewayGatekeeper, UUPSUpgradeable
```

Severity and Impact Summary

If exploited, this vulnerability could result in an attacker taking over ownership of the contract. Such an exploit could severely compromise the security and functionality of the contract.

Recommendation

To prevent the implementation contract from being used, the `_disableInitializers` function should be called in the constructor to automatically lock it when it is deployed.

Recreated network could be taken over

Finding ID: FYEO-IDENTITY-02
Severity: **Medium**
Status: **Remediated**

Description

Networks can be closed and recreated. Most of the relevant data is properly checked to be empty or reset. The `_nextPrimaryAuthoritys[networkName]` is not cleared out however.

Proof of Issue

File name: smart-contract/contracts/GatewayNetwork.sol
Line number: 108

```
function closeNetwork(bytes32 networkName) external override
onlyPrimaryNetworkAuthority(networkName) {
    require(_networks[networkName].primaryAuthority != address(0), "Network does not
exist");
    require(_networks[networkName].gatekeepers.length == 0, "Network can only be removed
if no gatekeepers are in it");
    require(networkFeeBalances[networkName] == 0, "Network has fees that need to be
withdrawn");

    delete _networks[networkName];

    emit GatekeeperNetworkDeleted(networkName);
}
```

Severity and Impact Summary

This means if a new admin is proposed for network “ABCD” but the new admin does not accept their role, the network is closed in the meantime. And someone else recreates it with the same name “ABCD” the originally proposed admin can now accept ownership and become the owner of a network they should not have access to.

Recommendation

Make sure `_nextPrimaryAuthoritys[networkName]` is zero or unset it in the call.

Flags may be improperly unset

Finding ID: FYEO-IDENTITY-03
Severity: **Low**
Status: **Acknowledged**

Description

The return values of both the add and remove flag operations are ignored.

Proof of Issue

File name: smart-contract/contracts/FlagsStorage.sol
Line number: 141

```
function _addFlag(bytes32 flag, uint8 index) internal {
    if (supportedFlagsMask.checkBit(index)) revert
FlagsStorage__IndexAlreadyUsed(index);
    if (_supportedFlags.contains(flag)) revert FlagsStorage__FlagAlreadyExists(flag);

    flagIndexes[flag] = index;
    _supportedFlags.add(flag);
    supportedFlagsMask = supportedFlagsMask.setBit(index);

    emit FlagAdded(flag, index);
}

function _removeFlag(bytes32 flag) internal {
    _supportedFlags.remove(flag);
    uint8 _index = flagIndexes[flag];

    supportedFlagsMask = supportedFlagsMask.clearBit(_index);
    delete flagIndexes[flag];

    emit FlagRemoved(flag);
}
```

Severity and Impact Summary

The remove flag operation does not check if the flag is being used. The returned `_index` can be the default value 0. This will then proceed to clear the wrong bit.

Recommendation

Make sure to check the returned value and proceed accordingly.

Missing zero address checks

Finding ID: FYEO-IDENTITY-04
Severity: **Low**
Status: **Remediated**

Description

Some parameters are not checked against the zero address.

Proof of Issue

File name: smart-contract/contracts/ChargeHandler.sol
Line number: 69

```
function setNetworkContractAddress(address gatewayNetworkContract) external
onlyRole(DEFAULT_ADMIN_ROLE) {
    _gatewayNetworkContract = gatewayNetworkContract;
}
```

File name: smart-contract/contracts/Gatekeeper.sol
Line number: 19

```
function initialize(address owner) initializer public {
  // Contract deployer is the initial super admin
  _superAdmins[owner] = true;
}
function setNetworkContractAddress(address gatewayNetworkContract) external
onlySuperAdmin override {
    _gatewayNetworkContract = gatewayNetworkContract;
}
```

File name: smart-contract/contracts/GatewayNetwork.sol
Line number: 32

```
function initialize(address owner, address gatewayGatekeeperContractAddress, address
gatewayStakingContractAddress) initializer public {
    _superAdmins[owner] = true;

    // Allow contract deployer to set NETWORK_FEE_PAYER_ROLE role
    _grantRole(DEFAULT_ADMIN_ROLE, 0, owner);
    _setRoleAdmin(NETWORK_FEE_PAYER_ROLE, 0, DEFAULT_ADMIN_ROLE);

    _gatewayGatekeeperContractAddress = gatewayGatekeeperContractAddress;
    _gatewayGatekeeperStakingContractAddress = gatewayStakingContractAddress;
}
```

File name: smart-contract/contracts/GatewayToken.sol
Line number: 450

```
function _setChargeHandler(address chargeHandler) internal {
    _chargeHandler = IChargeHandler(chargeHandler);

    emit ChargeHandlerUpdated(chargeHandler);
}
```

File name: smart-contract/contracts/TokenBitMask.sol
Line number: 36

```
function _setFlagsStorage(address _flagsStorage) internal {
    flagsStorage = IFlagsStorage(_flagsStorage);

    emit FlagsStorageUpdated(_flagsStorage);
}
```

Severity and Impact Summary

Missing zero checks could lead to misconfigurations that break the system.

Recommendation

Make sure to check the addresses for zero inputs.

Network fees not checked during initialization

Finding ID: FYEO-IDENTITY-05
Severity: **Low**
Status: **Remediated**

Description

Creating a network does not properly check the fees being set.

Proof of Issue

File name: smart-contract/contracts/GatewayNetwork.sol
Line number: 43

```
function createNetwork(GatekeeperNetworkData calldata network) external override {
    bytes32 networkName = network.name;

    require(networkName != bytes32(0), "Network name cannot be an empty string");
    require(network.primaryAuthority != address(0), "Network primary authority cannot be
zero address");
    // Check if network name already exist. If it does throw and error
    if(_networks[networkName].primaryAuthority != address(0)) {
        revert GatewayNetworkAlreadyExists(string(abi.encodePacked(networkName)));
    }

    _networks[networkName] = network;

function updateFees(NetworkFeesBps calldata fees, bytes32 networkName) external override
onlyPrimaryNetworkAuthority(networkName) {
    // checks
    require(fees.issueFee <= MAX_FEE_BPS, "Issue fee must be below 100%");
    require(fees.refreshFee <= MAX_FEE_BPS, "Refresh fee must be below 100%");
    require(fees.expireFee <= MAX_FEE_BPS, "Expiration fee must be below 100%");
    require(fees.freezeFee <= MAX_FEE_BPS, "Freeze fee must be below 100%");
```

The `updateFees` function does however check bounds.

Severity and Impact Summary

Fees can be configured to exceed 100% during the network’s creation.

Recommendation

Make sure to verify the fees given in the `createNetwork` function.

Admin modification is done in a single step

Finding ID: FYEO-IDENTITY-06
Severity: **Informational**
Status: **Acknowledged**

Description

The update of the admin can be done in one step - which implies a small risk of losing access if the wrong address is specified.

Proof of Issue

File name: smart-contract/contracts/FlagsStorage.sol
Line number: 71

```
function updateSuperAdmin(address newSuperAdmin) external onlySuperAdmin {
    if (newSuperAdmin == address(0)) revert Common__MissingAccount();

    emit SuperAdminUpdated(superAdmin, newSuperAdmin);
    superAdmin = newSuperAdmin;
}
```

Severity and Impact Summary

If the wrong address is somehow entered, admin access to the contract is lost.

Recommendation

A better approach would be to do it in two steps. First propose a new admin and then have the new admin accept the transfer.

Code clarity

Finding ID: FYEO-IDENTITY-07
Severity: **Informational**
Status: **Acknowledged**

Description

There are some locations in which the code clarity can be improved.

Proof of Issue

File name: smart-contract/contracts/library/InternalTokenApproval.sol
Line number: 28

```
/**
 * @dev Set a token approval.
 * @param tokenAddress The address of the token to approve.
 * @param tokens The number of tokens to approve.
 * @param network The specific network for which the tokens are approved.
 */
function _setApproval(address gatewayTokenAddress, address tokenAddress, uint256 tokens,
uint256 network) internal {
```

Missing doc for param `gatewayTokenAddress`.

```
/**
 * @dev Consume a certain amount of the token approval.
 * @param user The user for which the approval details are to be consumed.
 * @param gatewayTokenAddress The address of the gateway token contract that the tokens
are approved to be drwan by.
 * @param tokenAddress The address of the token being spent.
 * @param tokens The number of tokens to consume.
 * @param network The specific network from which the tokens are consumed.
 * @return A boolean indicating if the operation was successful.
 */
function _consumeApproval(
    address user,
    address gatewayTokenAddress,
    address tokenAddress,
    uint256 tokens,
    uint256 network
) internal returns (bool, uint256) {
```

Missing doc for returned tuple.

File name: smart-contract/contracts/library/BitMask.sol
Line number: 27

```
uint internal constant _ONE = uint256(1);

...

function checkBit(uint256 self, uint8 index) internal pure returns (bool) {
    return (self & (uint256(1) << index)) > 0;
}
```

Does not make use of the previously declared constant `_ONE`.

File name: smart-contract/contracts/ParameterizedAccessControl.sol
Line number: 147

```
function renounceRole(bytes32 role, uint256 domain, address account) public virtual
override {
    if (account != _msgSender()) revert
ParameterizedAccessControl__RenounceRoleNotForSelf(role, account);

    _revokeRole(role, domain, account);
}
```

The `_msgSender` could also be used directly in this case.

File name: smart-contract/contracts/GatewayToken.sol
Line number: 337

```
function getToken(uint tokenId)
    external
    view
    virtual
    returns (address owner, uint8 state, string memory identity, uint expiration, uint
bitmask)
{
    owner = ownerOf(tokenId);
    state = uint8(_tokenStates[tokenId]);
    expiration = _expirations[tokenId];
    bitmask = _getBitMask(tokenId);

    return (owner, state, identity, expiration, bitmask);
}

/**
 * @dev Triggers to get specified `tokenId` expiration timestamp
 * @param tokenId Gateway token id
 */
function getExpiration(uint tokenId) external view virtual returns (uint) {
    _checkTokenExists(tokenId);

    return _expirations[tokenId];
}

/**
 * @dev Triggers to get gateway token bitmask
 */
function getTokenBitmask(uint tokenId) external view virtual returns (uint) {
    return _getBitMask(tokenId);
}
```

The `_checkTokenExists` function is inconsistently used.

File name: smart-contract/contracts/GatewayNetwork.sol
Line number: 62

```
function transferNetworkFees(uint256 feeAmount, bytes32 networkName, address
tokenSender) external payable override onlyRole(NETWORK_FEE_PAYER_ROLE, 0) {
    // Check
    require(_networks[networkName].primaryAuthority != address(0), "Network does not
exist");
```

The `transferNetworkFees` function accepts `0` as `feeAmount`.

File name: smart-contract/contracts/FlexibleNonceForwarder.sol
Line number: 58

```
if (success == false) {
```

Use `!success` instead.

File name: smart-contract/contracts/GatewayNetwork.sol
Line number: 157

```
for(uint i = 0; i < currentGatekeepers.length; i++) {
    if(currentGatekeepers[i] == gatekeeper) {
        // Swap gatekeeper to be removed with last element in the array
        _networks[networkName].gatekeepers[i] =
_networks[networkName].gatekeepers[currentGatekeepers.length - 1];
        // Remove last element in array
        _networks[networkName].gatekeepers.pop();
    }
}
```

Insert a `break`, there is no need to iterate further once the item was found.

File name: smart-contract/contracts/Gatekeeper.sol
Line number: 36

```
function initializeGatekeeperNetworkData(bytes32 networkName, address gatekeeper,
GatekeeperStatus initialStatus) external onlyNetworkContract override {
    _gatekeeperStates[gatekeeper][networkName].initialized = true;
    _gatekeeperStates[gatekeeper][networkName].lastFeeUpdateTimestamp = 0;
    updateGatekeeperStatus(networkName, gatekeeper, initialStatus);
}
```

This function may overwrite / re-initialize data. Make sure this is implemented as designed and document accordingly.

File name: smart-contract/contracts/GatewayToken.sol
Line number: 591

```
function _checkSenderRole(bytes32 role, uint network) internal view {
    _checkRole(role, network, _msgSender());
}
```

This function is unused.

File name: smart-contract/contracts/GatewayStaking.sol
Line number: 37

```
function setMinimumGatekeeperStake (uint256 minStakeAmount) public override
onlySuperAdmin {
   GLOBAL_MIN_GATEKEEPER_STAKE = minStakeAmount;
}
```

No bounds are checked for the `minStakeAmount`, which can therefore be 0. Make sure this is acceptable.

Severity and Impact Summary

Not a security concern. Improving the code clarity and documentation does however improve the maintainability of the project.

Recommendation

Follow best coding practices to create a maintainable codebase.

Fees can be changed to frontrun a select user

Finding ID: FYEO-IDENTITY-08
Severity: **Informational**
Status: **Acknowledged**

Description

While there is a limit on how often fees can be changed, they can be changed on demand. A user can thus be frontrun and end up paying more tokens than expected.

Proof of Issue

The issue is due to a lack of the expected fees sent in the user request.

Severity and Impact Summary

Some users may be frontrun and pay more fees than expected (or less).

Recommendation

Consider sending the expected fee and failing the transaction if the fee is unexpectedly higher.

Tokens can be expired on mint

Finding ID: FYEO-IDENTITY-09
Severity: **Informational**
Status: **Remediated**

Description

There is no check to see if the given expiration time is in the past.

Proof of Issue

File name: smart-contract/contracts/GatewayToken.sol
Line number: 200

```
function mint(
    address to,
    uint network,
    uint expiration,
    uint mask,
    ChargeParties memory partiesInCharge
) external payable virtual {
    // CHECKS
    _checkGatekeeper(network);
    address gatekeeper = _msgSender();

    // EFFECTS
    uint tokenId = ERC3525Upgradeable._mint(to, network, 1);
    uint networkExpiration = block.timestamp +
IGatewayNetwork(_gatewayNetworkContract).getNetwork(network).passExpireDurationInSeconds
;

    if(networkExpiration > block.timestamp) {
        _expirations[tokenId] = networkExpiration;
    } else if (expiration > 0) {
        _expirations[tokenId] = expiration;
    }
```

Severity and Impact Summary

Tokens could already be expired when minted.

Recommendation

Make sure to verify the expiry time.

Our Process

Methodology

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

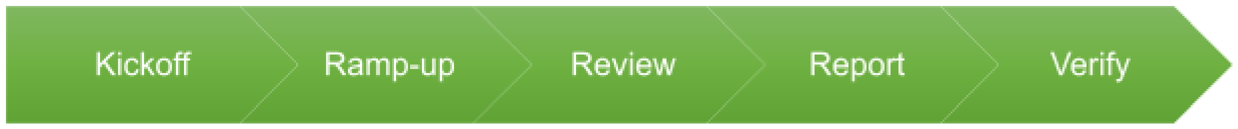

Figure 2: Methodology Flow

Kickoff

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

The Classification of vulnerabilities

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low
- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions
- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations