F Y E O

Security Code Review of DriftToken

Drift Token

March 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

Drift Token engaged FYEO Inc. to perform a Security Code Review of DriftToken.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on March 15 - March 26, 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-DRIF-01 – The selling fee is unbounded in size
- FYEO-DRIF-02 – Using dangerous slippage when swapping on Uniswap
- FYEO-DRIF-03 – A state variable is never initialized but is read
- FYEO-DRIF-04 – Insufficient test coverage
- FYEO-DRIF-05 – OpenZeppelin contracts are not used as an external dependency
- FYEO-DRIF-06 – A code that should be executed once is run multiple times
- FYEO-DRIF-07 – Floating pragma
- FYEO-DRIF-08 – Redundant nonReentrant modifiers in ERC-20 operations
- FYEO-DRIF-09 – Unused state variables
- FYEO-DRIF-10 – Emit on-chain events when changing crucial contract parameters

- FYEO-DRIF-11 – Group functions by their visibility
- FYEO-DRIF-12 – Improve code readability
- FYEO-DRIF-13 – Improve function and variable naming

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 of DriftToken. 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 private repository at https://github.com/moazzamgodil/drift_token_contract with the commit hash 6958c3f8f8a9493696e790383285c5b38bee32c.

Files included in the code review
drift_token/ └── DriftToken.sol

Table 1: Scope

TECHNICAL ANALYSES AND FINDINGS

During the Security Code Review of DriftToken, we discovered:

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

The following chart displays the findings by severity.

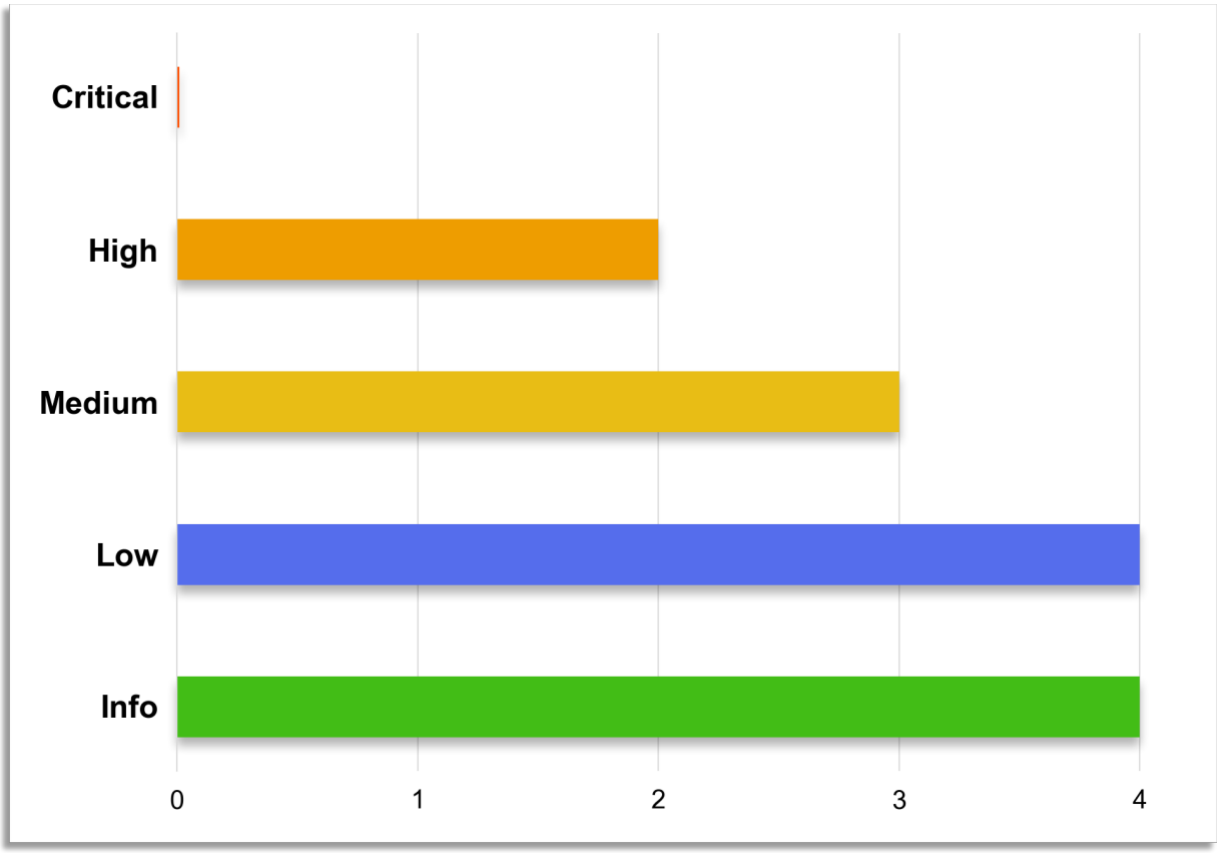


Figure 1: Findings by Severity

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-DRIF-01	**High**	The selling fee is unbounded in size
FYEO-DRIF-02	**High**	Using dangerous slippage when swapping on Uniswap
FYEO-DRIF-03	**Medium**	A state variable is never initialized but is read
FYEO-DRIF-04	**Medium**	Insufficient test coverage
FYEO-DRIF-05	**Medium**	OpenZeppelin contracts are not used as an external dependency
FYEO-DRIF-06	**Low**	A code that should be executed once is run multiple times
FYEO-DRIF-07	**Low**	Floating pragma
FYEO-DRIF-08	**Low**	Redundant nonReentrant modifiers in ERC-20 operations
FYEO-DRIF-09	**Low**	Unused state variables
FYEO-DRIF-10	**Informational**	Emit on-chain events when changing crucial contract parameters
FYEO-DRIF-11	**Informational**	Group functions by their visibility
FYEO-DRIF-12	**Informational**	Improve code readability
FYEO-DRIF-13	**Informational**	Improve function and variable naming

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

We conducted a thorough audit of the Drift token in order to assess its security posture.

During the audit, our security assessment team concluded that the code is well-structured and is extensively documented, which made it easier to read and understand the smart contract.

Moreover, a whitepaper was provided to us that described the audited system in general. Over the course of this audit, our team found that the implementation is consistent with the behavior described in the whitepaper.

However, we would like to recommend covering all code with end-to-end and unit tests as it may prevent most of the issues from happening.

To summarize, all the issues found **don't** endanger users and their funds but may severely impact the user experience.

Overall, the code is nicely written and documented, but there is room for improvement both security- and quality-wise.

During the course of the audit, the team worked diligently with the FYEO team to mitigate all the errors and issues identified in the code.

THE SELLING FEE IS UNBOUNDED IN SIZE

Finding ID: FYEO-DRIF-01
Severity: **High**
Status: **Remediated**

Description

The contract has a concept of dynamic sell fee that decreases gradually from 25% to 5%. The problem happens because the `changeDays` function can be used to set `_tradeFeeReduceDays` to any number of days, but the `getSellFee` function expects the `_tradeFeeReduceDays` to be: 7 >= `_tradeFeeReduceDays` >= 0.

Proof of Issue

File name: DriftToken.sol

Line number: 3625-3629, 3772-3774

```
...
    function getSellFee() private view returns (uint256) {
        uint256 diff = (_initSellFee - sellFee);
        uint256 fees = diff.div(7).mul(_getDays(block.timestamp,
_tradeFeeReduceDays)).add(sellFee);
        return fees;
    }
...
    function changeDays(uint256 _seconds) external onlyOwner {
        _tradeFeeReduceDays = block.timestamp + _seconds;
    }
...
```

This function doesn't restrict the `_seconds` parameter and thus can be used to add any number of days to the `_tradeFeeReduceDays` state variable. This will cause the `_getDays` function to return a bigger number of days than expected by the `getSellFee` function.

A PoC case (using the default parameters: `_initSellFee` is 25% and `sellFee` is 5%):

If we add 30 days to `_tradeFeeReduceDays`, the `_getDays` function will return 30 >= `days` >= 0, and the `getSellFee` function will return
90% >= `fee`>= 5%.

Severity and Impact Summary

Setting `_tradeFeeReduceDays` to values greater than `7` will cause users to pay an extreme fee, because the `getSellFee` function expects `_tradeFeeReduceDays` to be in range: `block.timestamp + 7 days` >= `_tradeFeeReduceDays` >= `block.timestamp`

Recommendation

It is recommended to restrict the number of days that could be added to the `_tradeFeeReduceDays` state variable in the `changeDays` function OR change the logic inside the `getSellFee` to allow having larger fee adjusting periods.

USING DANGEROUS SLIPPAGE WHEN SWAPPING ON UNISWAP

Finding ID: FYEO-DRIF-02
Severity: **High**
Status: **Remediated**

Description

The `amountOutMin` parameter is set to `0` when calling Uniswap swap functions.

Proof of Issue

File name: DriftToken.sol

Line number: 3569-3585, 3594-3602

```
...
    function swapTokensForToken(uint256 _tokenAmount, address _dexRouter, address[]
memory _path) private  {
        IDexRouter(_dexRouter).swapExactTokensForTokensSupportingFeeOnTransferTokens(
            _tokenAmount,
            0,
            _path,
            feeWallet,
            block.timestamp
        );
    }
...
    function swapTokensForEth(uint256 _tokenAmount, address _dexRouter, address[]
memory _path) private  {
        IDexRouter(_dexRouter).swapExactTokensForETHSupportingFeeOnTransferTokens(
            _tokenAmount,
            0,
            _path,
            feeWallet,
            block.timestamp
        );
    }
...
```

The second argument to the functions `swapExactTokensForETHSupportingFeeOnTransferTokens`and `swapExactTokensForTokensSupportingFeeOnTransferTokens` corresponds to:

The minimum amount of output tokens that must be received for the transaction not to revert.

So, by setting this argument to `0` we agree to the 100% slippage. This can be exploited by malicious miners and users to perform a sandwich attack.

Severity and Impact Summary

Setting the `amountOutMin` to `0` could lead to fund losses and must be avoided.

Recommendation

It is recommended to set a reasonable maximum slippage (the recommended one is 2-15% based on the swap size) to protect against frontrunning and sandwich attacks.

A STATE VARIABLE IS NEVER INITIALIZED BUT IS READ

Finding ID: FYEO-DRIF-03
Severity: **Medium**
Status: **Remediated**

Description

A state variable is never initialized but is read from in a function.

Proof of Issue

File name: DriftToken.sol

Line number: 3234

```
...
    mapping(IDexRouter => bool) private dexRouterExist; // Router address check
```

This variable is not initialized in the code, but is read in the `_createPair` function:

```
...
    function _createPair(IDexRouter _dexRouter, address _otherTokenForPair) internal {
        ...
        if(!dexRouterExist[_dexRouter]) {
            dexRouters.push(_dexRouter);
        }
        ...
    }
...
```

This `if` statement will cause `dexRouters` array to hold duplicate `dexRouter` entries, because `dexRouterExist` is *never updated*.

Severity and Impact Summary

Reading an uninitialized variable could lead to various bugs and hard-to-debug issues.

Recommendation

It is recommended to update the `dexRouterExist` state variable after inserting a new `dexRouter` into the `dexRouters` array.

INSUFFICIENT TEST COVERAGE

Finding ID: FYEO-DRIF-04
Severity: **Medium**
Status: **Open**

Description

No unit and E2E tests present

Severity and Impact Summary

Testing is important when dealing with applications that are responsible for storing user funds like dApps and tokens. It's important to cover all code with unit tests to verify that its behavior is consistent with the whitepaper and avoid introducing new issues in the future.

Recommendation

It is recommended to increase test coverage by adding unit and E2E tests

OPENZEPPELIN CONTRACTS ARE NOT USED AS AN EXTERNAL DEPENDENCY

Finding ID: FYEO-DRIF-05
Severity: **Medium**
Status: **Remediated**

Description

The contract relies on the contracts provided by OpenZeppelin, but these contracts aren't used as an external dependency and instead are just copied from the OpenZeppelin's repository.

Proof of Issue

File name: DriftToken.sol

Line number: 22-2235, 2467-3130

```
...
// OpenZeppelin Contracts (last updated v5.0.0) (utils/structs/EnumerableSet.sol)
...
// OpenZeppelin Contracts (last updated v5.0.0) (utils/Pausable.sol)
...
```

Severity and Impact Summary

OpenZeppelin contracts are well-known and are best used as an external dependency. This way, we don't have to manage the contract versions ourselves and automatically receive important security updates and new features.

Recommendation

It is recommended to remove the copied contracts and replace them with external dependencies using one of the available frameworks (for example, Hardhat).

A CODE THAT SHOULD BE EXECUTED ONCE IS RUN MULTIPLE TIMES

Finding ID: FYEO-DRIF-06
Severity: **Low**
Status: **Remediated**

Description

This code should only run once but instead runs every time the `launchPair` function is invoked.

Proof of Issue

File name: DriftToken.sol

Line number: 3696-3697

```
...
        openTrading();
        feeEnable = true;
...
```

Severity and Impact Summary

Calling the `openTrading` function and setting the `feeEnable` variable each time is redundant, as it doesn't introduce any new changes and only burns Gas.

Recommendation

It it recommended to refactor this logic so that this code is executed only when the `launchPair` function is called the first time.

FLOATING PRAGMA

Finding ID: FYEO-DRIF-07
Severity: **Low**
Status: **Remediated**

Description

The Solidity version is not pinned.

Proof of Issue

File name: DriftToken.sol

Line number: 19

```
pragma solidity ^0.8.9;
```

Severity and Impact Summary

Contracts should be tested and deployed using the same Solidity version. Floating pragma may introduce uncertainty when it comes to the used Solidity version.

Recommendation

It is recommended to use a specific Solidity version. It's also recommended to take a look at the List of Known Bugs when choosing a version.

REDUNDANT NONREENTRANT MODIFIERS IN ERC-20 OPERATIONS

Finding ID: FYEO-DRIF-08
Severity: **Low**
Status: **Remediated**

Description

The `nonReentrant` modifier is used when doing operation on ERC-20 tokens.

Proof of Issue

File name: DriftToken.sol

Line number: 3345, 3356, 3366, 3374, 3382

```
...
function mint(address to, uint256 amount) public onlyMinter nonReentrant
isSupplyExceed(amount) {
...
```

The `nonReentrant` modifier can be safely removed from these functions as we don't give the control over the transaction flow to the user when dealing with ERC-20 tokens.

Severity and Impact Summary

Redundant modifiers can lead to higher gas usage and should be avoided.

Recommendation

It is recommended to remove the `nonReentrant` modifiers from functions that deal with `ERC-20` tokens as it doesn't introduce any security benefits.

UNUSED STATE VARIABLES

Finding ID: FYEO-DRIF-09
Severity: **Low**
Status: **Remediated**

Description

Some state variables are unused: they are never read, can't be publicly accessed, and are only written to.

Proof of Issue

File name: DriftToken.sol

Line number: 3232-3233

```
...
    mapping(address => IDexRouter) private dexRouter; // Mapping Pair address => Router address
    mapping(IDexRouter => address[]) private dexPoolPair; // Mapping Router address => array of pair addresses
...
```

Both these variables are updated in the `_createPair` function, but they are never read from. Moreover, they are private and don't have a public "getter", so we can't access them conveniently.

Severity and Impact Summary

Having unused (or *dead*) code is considered to be error prone and should be avoided when possible.

Recommendation

It is recommended to remove the unused state variables to avoid unnecessary gas usage.

EMIT ON-CHAIN EVENTS WHEN CHANGING CRUCIAL CONTRACT PARAMETERS

Finding ID: FYEO-DRIF-10
Severity: **Informational**
Status: **Remediated**

Description

The are missing events after changing crucial state variables in the code.

Proof of Issue

File name: DriftToken.sol

Line number: 3435-3437, 3693, 3772-3774

```
...
    function setEnableFees(bool _enable) external onlyOwner {
        feeEnable = _enable;
    }
...
    function launchPair(address _dexRouter, address _otherTokenForPair) external
onlyOwner {
        ...
            _tradeFeeReduceDays = block.timestamp + 1 weeks;
        ...
    }
...
    function changeDays(uint256 _seconds) external onlyOwner {
        _tradeFeeReduceDays = block.timestamp + _seconds;
    }
...
```

Consider emitting an event after changing these state variables in the code as they are crucial to the smart contract in question.

Severity and Impact Summary

Emitting on-chain events is important for easy debugging and tracing changes in a contract.

Recommendation

It is recommended to emit an on-chain event when changing an important to the contract state variable.

GROUP FUNCTIONS BY THEIR VISIBILITY

Finding ID: FYEO-DRIF-11
Severity: **Informational**
Status: **Remediated**

Description

Functions in a contract should be grouped based on their visibility modifier.

Proof of Issue

File name: DriftToken.sol

Line number: 3200-3849

Functions in the `DriftToken`contract are not grouped by their visibility modifier.

Severity and Impact Summary

Grouping functions by their visibility makes contracts more readable and easier to understand, as we can clearly separate the public interface from the internal one.

Recommendation

It is recommended to group functions based on their visibility. Consider taking a look at the Official Solidity Style Guide.

IMPROVE CODE READABILITY

Finding ID: FYEO-DRIF-12
Severity: **Informational**
Status: **Remediated**

Description

The code readability can be improved by making a few small changes.

Proof of Issue

File name: DriftToken.sol

Line number: 2407-2410

```
...
    function grantMintAndBurnRoles(address burnAndMinter) external {
        grantMintRole(burnAndMinter);
        grantBurnRole(burnAndMinter);
    }
...
```

This function can only be called by the owner, but its signature doesn't say anything about that. A developer needs to take a look at the comment or functions that are called inside the `grantMintAndBurnRoles` function in order to figure out who can call it. Even though omitting the modifier here doesn't introduce any security issues, it's still recommended to add it, because the function's signature should **fully** express its intent.

Suggested fix:

```
...
    function grantMintAndBurnRoles(address burnAndMinter) external onlyOwner {
...
```

File name: DriftToken.sol

Line number: 3214

```
    uint256 public maxSupply = 10000000000 * 10**18; // 10 Billion
```

Long constants are hard to read and understand. Consider adding visual delimiters to improve that:

```
    uint256 public maxSupply = 10_000_000_000 * 10**18; // 10 Billion
```

File name: DriftToken.sol

Line number: 3478

```
    _newUserBuy[to] = block.timestamp + (60 * 15); // 15 Mins
```

The `minutes` time unit can be used to avoid calculating this manually:

```
    _newUserBuy[to] = block.timestamp + 15 minutes;
```

File name: DriftToken.sol

Line number: 3526-3529, 3538-3541

```
        if(!presale_dynamic_to_stake.isStaker(_address) &&
presale_ico.amountOfAddressPerType(_address, 0) > 0) {
            return true;
        }
        return false;
```

In cases like this, we can just return the condition inside `if` directly (as it's already a boolean):

```
        return !presale_dynamic_to_stake.isStaker(_address) &&
presale_ico.amountOfAddressPerType(_address, 0) > 0;
```

File name: DriftToken.sol

Line number: 3399-3400, 3415-3416

```
        super.transfer(to, _value);
        return true;
```

In cases like this, we can just return the call result directly (as it has the same return type as our function):

```
        return super.transfer(to, _value);
```

Severity and Impact Summary

Improving code readability is an important step towards maintainable and secure code. Addressing the described problems can improve the code and make it easier for fellow developers to read and understand it.

Recommendation

It is recommended to improve the code readability by addressing the described issues.

IMPROVE FUNCTION AND VARIABLE NAMING

Finding ID: FYEO-DRIF-13
Severity: **Informational**
Status: **Remediated**

Description

Function and variable naming can be improved by making a few small changes to the existing names.

Proof of Issue

File name: DriftToken.sol

Line number: 3225

```
...
    uint256 private _tradeFeeReduceDays = 0;
...
```

Consider renaming this variable to `_tradeFeeReducedUntil`, as it stores a timestamp and not the number of days.

File name: DriftToken.sol

Line number: 3425

```
...
    function excludeFromFees(address _address, bool _isExclude) public onlyOwner {
...
```

Consider renaming this function to `setExcludeFromFees`, as it allows both disabling and enabling fees for a user.

Severity and Impact Summary

Picking a right name for a function or variable can be a daunting task. Nevertheless, it's important to choose the most reasonable names that best shows this variable/function's intent. Improving component naming increases code readability and reduces chances to introduce an error while using them.

Recommendation

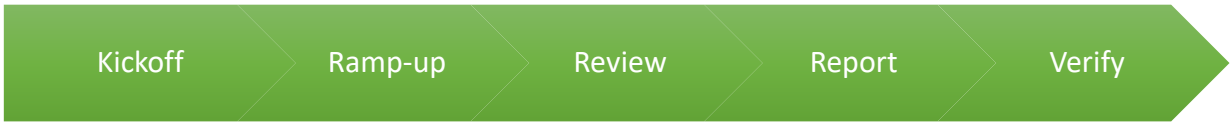
It is recommended to improve the code readability by addressing the mentioned issues.

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