CERTIK

Security Assessment

MatrixETF

Nov 1st, 2021

Table of Contents

About

Summary

This report has been prepared for Matrix Capital Limited to discover issues and vulnerabilities in the source code of the MatrixETF project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	MatrixETF
Platform	Ethereum
Language	Solidity
Codebase	https://etherscan.io/address/0x1a57367c6194199e5d9aea1ce027431682dfb411 https://github.com/MatrixETF/SingleAsset/tree/main
Commit	eb5e3369a87147c83b9e0aae2093f2f0a0fc3849 2951f3aa94e2cbe507ceff30705d940def277943

Audit Summary

Delivery Date	Nov 01, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	2	0	0	0	0	2
● Major	3	0	0	3	0	0
● Medium	2	0	0	2	0	0
● Minor	6	0	0	3	0	3
● Informational	12	0	0	8	0	4
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
IRC	interfaces/IRecipe.sol	31e85bcac5c0d96c0deba1a6cc5667fa493d8fc56ffc55ed6be9385d2d46a38a
IRK	interfaces/IRegistry.sol	f4329f472db24ac27905b738cce7b2b07c19a104547822cf749ccc115b9dd341
ISP	interfaces/ISmartPool.sol	7c6bdd52b83f76ddb5be8e3464320c85e0071d85046722a689419e8f4c744fa3
ISR	interfaces/ISmartPoolRegistry.sol	9df95877aff3dd9a5ceb902acbfa5db2c966d888d30750bebc3869c1f2fa68d7
IUR	interfaces/IUniRouter.sol	59e5296a05bee7c5e4f2e5f487a7610fe435b290c643c383686f20f585146beb
IWE	interfaces/IWETH.sol	858e885d5947a5c2e6d61daef0d792f50f4f0e291c6e4db932c2a049ae1950ea
MDF	MDF.sol	e98a03341831fafe931958f8cdbcf0dc811d4dac6bc31526d533a4ecbfd0fd86
SPR	SmartPoolRegistry.sol	7d78d0e878a1726e7ab25e65c135fb635f577565a93477f29b587f1d04e3c0f3
URC	UniRecipe.sol	abeff012116d5cb31e34e20d5c1e453067e6618569b746fdc487aefd3b7c2272
VCR	V1CompatibleRecipe.sol	5be6c63d78c53dff7cf26ac1a3bbcc03a74baa534d4e9bed216ee295a6058621

It is noted that there are some external calls below that are unknown.

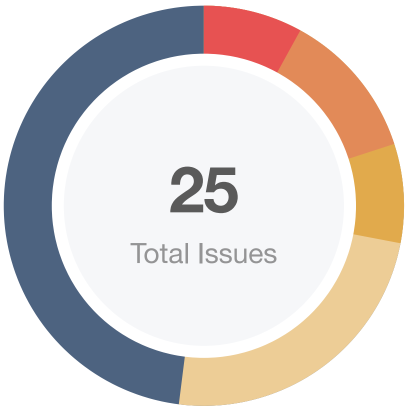
The functions of the contract `ISmartPool` below are unknown:

- `calcTokensForAmount`
- `joinPool`
- `exitPool`

The function of the contract `IUniRouter` below is unknown:

- `swapTokensForExactTokens`

Findings



🟥	Critical	2 (8.00%)
🟧	Major	3 (12.00%)
🟨	Medium	2 (8.00%)
🟨	Minor	6 (24.00%)
🟦	Informational	12 (48.00%)
🟩	Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
MDF-01	Initial Token Distribution	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
MDF-02	Mismatch Between Code Implementation and Introduction Docs	Logical Issue	● Minor	ⓘ Acknowledged
SPR-01	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
SPR-02	Lack of Input Validation	Volatile Code	● Minor	✓ Resolved
URC-01	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
URC-02	Unchecked Value of low-level Call	Volatile Code	● Informational	✓ Resolved
URC-03	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	● Informational	ⓘ Acknowledged
URC-04	Incompatibility With Deflationary Tokens	Volatile Code	● Informational	ⓘ Acknowledged
URC-05	Function Can Revert Instead Of Returning Extreme Value	Volatile Code	● Minor	✓ Resolved
URC-06	Missing Emit Events	Coding Style	● Informational	ⓘ Acknowledged
URC-07	Unused Function Parameter Can Be Removed	Language Specific	● Informational	ⓘ Acknowledged
URC-08	Unused local variable	Gas Optimization	● Informational	ⓘ Acknowledged
URC-09	Unassigned Variable	Logical Issue	● Minor	ⓘ Acknowledged

ID	Title	Category	Severity	Status
URC-10	Incorrect Return Statement	Logical Issue	● Minor	Resolved
URC-11	Lack of Addresses of Token Validation	Volatile Code	● Critical	Resolved
URC-12	The Specified `uniRouter`	Volatile Code	● Minor	Acknowledged
URC-13	Unknown implementations	Volatile Code	● Medium	Acknowledged
URC-14	Risk For Stealing Tokens From `UniRecipe`	Logical Issue	● Critical	Resolved
URC-15	Token Swap Issue	Logical Issue	● Informational	Resolved
VCR-01	Unchecked Value of low-level Call	Volatile Code	● Informational	Resolved
VCR-02	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	● Informational	Acknowledged
VCR-03	Usage of `transfer()` for sending Ether	Volatile Code	● Informational	Acknowledged
VCR-04	Redundant Code	Volatile Code	● Informational	Resolved
VCR-05	Unused local variable	Gas Optimization	● Informational	Acknowledged
VCR-06	Unknown implementations	Volatile Code	● Medium	Acknowledged

MDF-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	MDF.sol: 435	ⓘ Acknowledged

Description

The `initSupply` of tokens are sent to the `account` when deploying the contract. This could be a centralization risk as the `account` can distribute tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process.

Alleviation

`[MatrixETF Team]`: Before Token initial distribution, we had consensus with our investors and more than half of our community about the distribution protocol. Also after launching our Token, we have distributed tokens to the corresponding addresses according to the consensus.

MDF-02 | Mismatch Between Code Implementation and Introduction Docs

Category	Severity	Location	Status
Logical Issue	● Minor	MDF.sol: 434~436	ⓘ Acknowledged

Description

The project whitepaper indicates:

MDF is a token deployed on Ethereum and Solana blockchain network, and its maximum supply limit is 1,000,000,000 tokens.

But the code implementation does not limit the maximum supply of `MDF` tokens.

Recommendation

Consider Keeping the code implementation and the project whitepaper the same.

Alleviation

`[MatrixETF Team]`: We have set up smart contract as only can MINT once when we created the contract, also we have distributed tokens under the consensus to corresponding addresses after launching Token.

SPR-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	SmartPoolRegistry.sol: 11, 17, 28	ⓘ Acknowledged

Description

In the contract `SmartPoolRegistry`, the role `owner` has the authority over the following functions:

- `addSmartPool(address _smartPool)`
- `removeSmartPool(uint256 _index)`
- `removeSmartPoolByAddress(address _address)`

Any compromise to the `owner` account may allow the hacker to take advantage of this and do the following:

- add a smart pool.
- remove a smart pool.
- remove a smart pool by address.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

`[MatrixETF Team]`: We are utilizing the administrator address to control the configuration functions in smart contract, we will manage private key in addresses strictly.

SPR-02 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Minor	SmartPoolRegistry.sol: 17	✓ Resolved

Description

The given input parameter `_index` is missing check.

Recommendation

Consider adding check for the input parameter to prevent unexpected error as below:

```
    require(_index < entries.length, "Wrong index value");
```

Alleviation

The development team heeded our advice and resolved this issue in commit 135e0f17646f566a5e59f0cde9502a7dc45047b0.

URC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	UniRecipe.sol: 166, 170, 159	ⓘ Acknowledged

Description

In the contract `UniRecipe`, the role `owner` has the authority over the following functions:

- `saveToken(address _token, address _to, uint256 _amount)`
- `saveEth(address payable _to, uint256 _amount)`
- `setCustomHop(address _token, address _hop)`

Any compromise to the `owner` account may allow the hacker to take advantage of this and do the following:

- transfer tokens from the current contract to anyone.
- transfer eth from the current contract to anyone.
- set custom hop.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

`[MatrixETF Team]`: We are utilizing the administrator address to control the configuration functions in smart contract, we will manage private key in addresses strictly.

URC-02 | Unchecked Value of low-level Call

Category	Severity	Location	Status
Volatile Code	● Informational	UniRecipe.sol: 171	✓ Resolved

Description

The aforementioned lines perform `_to.call` but the return value is not checked in either case.

```
_to.call{value: _amount}("");
```

Recommendation

It is recommended to make sure that the value returned from low-level calls is checked.

```
(bool success, ) = _to.call{value: _amount}("");
require(success, "unable to send value, recipient may have reverted");
```

Alleviation

The development team heeded our advice and resolved this issue in commit 135e0f17646f566a5e59f0cde9502a7dc45047b0.

URC-03 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	● Informational	UniRecipe.sol: 167, 67	ⓘ Acknowledged

Description

The linked `transfer()` and `transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of proper ERC-20 implementation.

Recommendation

"As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from `transfer()` and `transferFrom()` is checked."

Alleviation

`[MatrixEFT Team]`: The constituted currencies of ETF will only contain the currencies which are in accordance with ERC-20 standard, and they will be utilized in Vault aggregator.

URC-04 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Volatile Code	● Informational	UniRecipe.sol: 14	ⓘ Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, an inconsistency in the amount will occur and the transaction may fail due to the validation checks.

Recommendation

We advise the client to regulate the input/output tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

`[MatrixEFT Team]`: We won't adopt any Deflationary Token as our constituted currency, so we are incompatible with Deflationary Tokens.

URC-05 | Function Can Revert Instead Of Returning Extreme Value

Category	Severity	Location	Status
Volatile Code	● Minor	UniRecipe.sol: 232~237, 245~250	✓ Resolved

Description

When the calling to the `_router.getAmountsIn` or `_router.getAmountsOut` fails, the function can revert it instead of returning `type(uint256).max`.

Recommendation

Consider refactoring the code as below:

```
function getPriceUniLike(address _inputToken, address _outputToken, uint256 _outputAmount,
IUniRouter _router) internal view returns(uint256) {
        if(_inputToken == _outputToken) {
            return(_outputAmount);
        }

        try _router.getAmountsIn(_outputAmount, getRoute(_inputToken, _outputToken))
returns(uint256[] memory amounts) {
            return amounts[0];
        } catch {
            revert();
        }
    }
```

```
function getPriceUniLike2(address _inputToken, address _outputToken, uint256 _inputAmount,
IUniRouter _router) internal view returns(uint256) {
        if(_inputToken == _outputToken) {
            return(_inputAmount);
        }

        try _router.getAmountsOut(_inputAmount, getRoute(_inputToken, _outputToken))
returns(uint256[] memory amounts) {
            return amounts[1];
        } catch {
            revert();
        }
    }
```

Alleviation

The development team heeded our advice and resolved this issue in commit 135e0f17646f566a5e59f0cde9502a7dc45047b0.

URC-06 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	UniRecipe.sol: 159~164, 170	ⓘ Acknowledged

Description

The function that affects the status of sensitive variables should be able to emit events as notifications.

- `setCustomHop()`
- `saveEth()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

`[MatrixEFT Team]`: From the service model, these 2 functions are different from sensitive function. MatrixETF has deal with it correspondingly. SetCustomHop() has not been accomplished yet. SaveETH() is a reserved function for dealing with the outlined situation after Swap (Accuracy Calculation Balance), the latest version of contract has been optimized. If there is a overage balance in Swap, we will return all of it to users. We won't keep any assets in Vault aggregator contract.

URC-07 | Unused Function Parameter Can Be Removed

Category	Severity	Location	Status
Language Specific	● Informational	UniRecipe.sol: 75	ⓘ Acknowledged

Description

In function `_bake`, the `_maxInput` parameter is not used.

Recommendation

Consider removing or commenting out the `_maxInput` parameter.

Alleviation

The development team replied that this parameter will be used in the future.

URC-08 | Unused local variable

Category	Severity	Location	Status
Gas Optimization	● Informational	UniRecipe.sol: 131, 125~126	ⓘ Acknowledged

Description

The following variables are only set values, but never used:

- `inputAmount`
- `dex`
- `customHop`

Recommendation

Consider removing the variables which are never used.

Alleviation

The development team replied that these variables will be used in the future.

URC-09 | Unassigned Variable

Category	Severity	Location	Status
Logical Issue	● Minor	UniRecipe.sol: 50~72	ⓘ Acknowledged

Description

In the function `bake`, there is no assignment for the return value `inputAmountUsed`.

Recommendation

Consider adding calculation logic and assignment for the return value `inputAmountUsed`.

Alleviation

No Alleviation.

URC-10 | Incorrect Return Statement

Category	Severity	Location	Status
Logical Issue	● Minor	UniRecipe.sol: 86~88	✓ Resolved

Description

The aforementioned code infers that if the `_inputToken` equals the `_outputToken`, the `swap` function will return directly without reverting state changes and sending a helpful error message.

Recommendation

Consider using the `require` function instead of `return`.

Alleviation

The development team heeded our advice and resolved this issue in commit 2951f3aa94e2cbe507ceff30705d940def277943.

URC-11 | Lack of Addresses of Token Validation

Category	Severity	Location	Status
Volatile Code	● Critical	UniRecipe.sol: 50~73	✓ Resolved

Description

The `msg.sender` could receive the doubled amount of `_inputToken` tokens if the `_inputToken` is the same as the `_outputToken` since, in that case, the `bake` function will transfer tokens to `msg.sender` twice.

Recommendation

Consider checking whether the `_inputToken` token is same as the `_outputToken` token or not.

Alleviation

The development team heeded our advice and resolved this issue by changing the visibility of the `bake` function to be `internal` in commit 2951f3aa94e2cbe507ceff30705d940def277943.

URC-12 | The Specified `uniRouter`

Category	Severity	Location	Status
Volatile Code	● Minor	UniRecipe.sol: 46	ⓘ Acknowledged

Description

The `uniRouter` is not specified in the contract and has a deep influence on the prices of the tokens when swapping.

Alleviation

`[MatrixETF team]`: the router of Uniswap will be adopted.

URC-13 | Unknown implementations

Category	Severity	Location	Status
Volatile Code	● Medium	UniRecipe.sol: 109, 256, 121, 8, 6, 6, 139, 156	ⓘ Acknowledged

Description

The functions of the contract `ISmartPool` below are unknown:

- `calcTokensForAmount`
- `joinPool`

The function of the contract `IUniRouter` below is unknown:

- `swapTokensForExactTokens`

Alleviation

`[MatrixEFT Team]`: calcTokensForAmount is the function to obtain prices of ETF and constituted currencies joinPool is the function to MINT fund swapTokensForExactTokens is the function to Swap in uniswap-v2 Router contract

URC-14 | Risk For Stealing Tokens From `UniRecipe`

Category	Severity	Location	Status
Logical Issue	● Critical	UniRecipe.sol: 75~81	✓ Resolved

Description

```
function bake(
        address _inputToken,
        address _outputToken,
        uint256 _maxInput,
        bytes memory _data
    ) external override returns(uint256 inputAmountUsed, uint256 outputAmount) {
        IERC20 inputToken = IERC20(_inputToken);
        IERC20 outputToken = IERC20(_outputToken);

        inputToken.safeTransferFrom(_msgSender(), address(this), _maxInput);

        (uint256 mintAmount) = abi.decode(_data, (uint256));

        outputAmount = _bake(_inputToken, _outputToken, _maxInput, mintAmount);

        uint256 remainingInputBalance = inputToken.balanceOf(address(this));
        if(remainingInputBalance > 0) {
            inputToken.transfer(_msgSender(), remainingInputBalance);
        }

        outputToken.safeTransfer(_msgSender(), outputAmount);

        return(inputAmountUsed, outputAmount);
    }

function _bake(address _inputToken, address _outputToken, uint256 _maxInput, uint256 _mintAmount)
internal returns(uint256 outputAmount) {
        swap(_inputToken, _outputToken, _mintAmount);

        outputAmount = IERC20(_outputToken).balanceOf(address(this));

        return(outputAmount);
    }
```

If the `_outputToken` is one of the tokens in the ETF Index, the contract `UniRecipe` will hold amounts of `_outputToken`. When the user calls `bake`, the value of `IERC20(_outputToken).balanceOf(address(this))` will be much greater than the actually swapped amount.

Recommendation

Consider refactoring the calculation of `outputAmount`. For example:

```
function _bake(address _inputToken, address _outputToken, uint256 _maxInput, uint256 _mintAmount)
internal returns(uint256 outputAmount) {
        uint256 outputAmountBefore = IERC20(_outputToken).balanceOf(address(this));
        swap(_inputToken, _outputToken, _mintAmount);
        outputAmount = IERC20(_outputToken).balanceOf(address(this)).sub(outputAmountBefore);
        return(outputAmount);
    }
```

Alleviation

The development team heeded our advice and resolved this issue in commit 135e0f17646f566a5e59f0cde9502a7dc45047b0.

URC-15 | Token Swap Issue

Category	Severity	Location	Status
Logical Issue	● Informational	UniRecipe.sol: 50~55	✓ Resolved

Description

In the aforementioned code, there is no limit with the input parameter `_inputToken`. So let's thinking about this case that the `_inputToken` value is the `ISmartPool` address and the `_outputToken` value is the `WETH` address, the caller can swap `ETF` to `ETH`. This will bypass the function `V1CompatibleRecipe.toETH(address _smartPool, uint256 _inputAmount)`, which will calculate the withdrawal fee.

Is that designed as expected?

Alleviation

The development team heeded our advice and resolved this issue in commit 2951f3aa94e2cbe507ceff30705d940def277943.

VCR-01 | Unchecked Value of low-level Call

Category	Severity	Location	Status
Volatile Code	● Informational	V1CompatibleRecipe.sol: 22	✓ Resolved

Description

The aforementioned lines perform `address(WETH).call` but the return value is not checked in either case.

```
address(WETH).call{value: msg.value}("");
```

Recommendation

It is recommended to make sure that the value returned from low-level calls is checked.

```
(bool success, ) = address(WETH).call{value: msg.value}("");
require(success, "unable to send value, recipient may have reverted");
```

Alleviation

The development team heeded our advice and resolved this issue in commit 135e0f17646f566a5e59f0cde9502a7dc45047b0.

VCR-02 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	● Informational	V1CompatibleRecipe.sol: 41	ⓘ Acknowledged

Description

The linked `transfer()` and `transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of proper ERC-20 implementation.

Recommendation

"As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from `transfer()` and `transferFrom()` is checked."

Alleviation

`[MatrixEFT Team]`: The constituted currencies of ETF will only contain the currencies which are in accordance with ERC-20 standard, and they will be utilized in Vault aggregator.

VCR-03 | Usage of `transfer()` for sending Ether

Category	Severity	Location	Status
Volatile Code	● Informational	V1CompatibleRecipe.sol: 34, 61	ⓘ Acknowledged

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

Alleviation

`[MatrixEFT Team]`: Utilizing Transfer is in order to control the size of contract as well as allow the ETF Transfer object only to be EOAS in Vault Aggregator contract

VCR-04 | Redundant Code

Category	Severity	Location	Status
Volatile Code	● Informational	V1CompatibleRecipe.sol: 19	✓ Resolved

Description

The code in the function `toETF` as below is redundant because the variable `calculatedSpend` is not used.

```
uint256 calculatedSpend = getPrice(address(WETH), _smartPool, _outputAmount);
```

Recommendation

Consider removing the redundant code.

Alleviation

The development team heeded our advice and resolved this issue in commit 135e0f17646f566a5e59f0cde9502a7dc45047b0.

VCR-05 | Unused local variable

Category	Severity	Location	Status
Gas Optimization	● Informational	V1CompatibleRecipe.sol: 43~47	ⓘ Acknowledged

Description

The following variables are only set values, but never used:

- `communitySwapFee`
- `communityJoinFee`
- `communityFeeReceiver`
- `poolAmountInFee`

Recommendation

Consider removing the variables which are never used.

Alleviation

The development team replied that these variables will be used in the future.

VCR-06 | Unknown implementations

Category	Severity	Location	Status
Volatile Code	● Medium	V1CompatibleRecipe.sol: 54	ⓘ Acknowledged

Description

The function of the contract `ISmartPool` below is unknown:

- `exitPool`

Alleviation

`[MatrixEFT Team]`: exitPool is the function to Redeem fund.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

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 you (“Customer” or 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, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

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 CertiK 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. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we’re able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.