PIEDAO EXPERIPIE SMART CONTRACT AUDIT

December 11, 2020

PieDAO

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of PieDAO. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

ExperiPie (TPIE++) is a new pool design with unlimited possibilities. The ExperiPie is based on the Diamond standard, this standard ensures contracts can grow beyond their restricted size. In the current DeFi space there are lots of opportunities to participate in governance of various protocols. The ExperiPie uses a very flexible CallFacet which makes it possible to execute any call on behalf of the pool. The ExperiPie will be used to participate in governance protocols to benefit PieDao participants. Yield farm opportunities can show up any time, not everyone has the liquidity or attention to fulfill every opportunity. Using the ExperiPie everyone can pool their tokens. Through the flexible nature of the pool it is possible to use active governance to take any yield farm opportunity.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The audited contract is a new pool design with unlimited possibilities. ExperiPie is based on the Diamond standard, this standard ensures that contracts can exceed their limited size. Due to the flexible nature of the pool, active management can be used. But these opportunities must be used with great care.

1.5 PROJECT DASHBOARD

Client	PieDAO
Audit name	ExperiPIE
Initial version	facf3c246d9c43f5b1e0bad7dc2b0a9a2a2393c5
Final version	abcc4aa5fdeac8a1a854d0b7a3aa88ff8f2bfb08
SLOC	1227
Date	2020-11-27 - 2020-12-11
Auditors engaged	2 auditors

FILES LISTING

LendingLogicAave.sol	LendingLogicAave.sol
LendingLogicCompound.sol	LendingLogicCompound.sol
LendingManager.sol	LendingManager.sol
LendingRegistry.sol	LendingRegistry.sol
RSIManager.sol	RSIManager.sol
BasketFacet.sol	BasketFacet.sol
LibBasketStorage.sol	LibBasketStorage.sol
CallFacet.sol	CallFacet.sol
LibCallStorage.sol	LibCallStorage.sol
ERC20Facet.sol	ERC20Facet.sol
LibERC20.sol	LibERC20.sol
LibERC20Storage.sol	LibERC20Storage.sol
CallProtection.sol	CallProtection.sol
LibReentryProtectionStorage.sol	LibReentryProtectionStorage.sol
ReentryProtection.sol	ReentryProtection.sol
PieFactoryContract.sol	PieFactoryContract.sol
IAaveLendingPool.sol	IAaveLendingPool.sol
IAToken.sol	IAToken.sol
IBasketFacet.sol	IBasketFacet.sol
ICallFacet.sol	ICallFacet.sol
ICToken.sol	ICToken.sol
IERC20Facet.sol	IERC20Facet.sol
IExperiPie.sol	IExperiPie.sol

ILendingLogic.sol	ILendingLogic.sol
IPriceReferenceFeed.sol	IPriceReferenceFeed.sol
ISynthetix.sol	ISynthetix.sol
Imports.sol	Imports.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	5
Warning	12
Comment	7

CONCLUSION

Several problems have been identified and properly addressed. The client responded well to our comments. According to our analysis, fixed contracts are free of vulnerabilities.

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Invalid setter function
File	PieFactoryContract.sol BasketFacet.sol
Severity	Major
Status	**Fixed** at **063c9237**

DESCRIPTION

- In the `setDefaultController()` function defined at line PieFactoryContract.sol#L34, the new value is assigned to the `defaultController` variable. But it is not taken into account that earlier using the `bakePie()` function, the pool owner could be changed to address of `defaultController`. Thus, as a result of the work of the `setDefaultController()` function, the pool owner will remain the same, while it is expected that the pool will be managed from a different address.
- In the `setFeeBeneficiary()` function defined at line BasketFacet.sol#L87 , the `LibBasketStorage.basketStorage().FeeBeneficiary` variable is assigned a new value. But it is not taken into account that earlier tokens could be issued to this address in the following places: BasketFacet.sol#L141, BasketFacet.sol#L175, BasketFacet.sol#L216.

RECOMMENDATION

It is necessary to carefully check the logic of these functions and make corrections if necessary.

CLIENT'S COMMENTARY

- Setting the default controller will set the owner of newly deployed pools to that address. Older pools will keep their current controller.
- I'll charge the fee upon changing the fee beneficiary so any fees up to that point will go to the current `feeBeneficiary`.

MJR-2	Incorrect logic when burning and minting tokens
File	ERC20Facet.sol BasketFacet.sol
Severity	Major
Status	**No issue**

DESCRIPTION

- The `mint()` function defined at the line ERC20Facet.sol#L44 is for minting new tokens.
  The amount of tokens is increased on the wallet with the address `_receiver`.
  But the ERC-20 specification also uses the value of the `totalSupply` variable.
  This variable is not incremented here.

- At line: ERC20Facet.sol#L48. The `burn()` function is for burning tokens. The amount of tokens is reduced on the wallet with the address `_from`.
  But the ERC-20 specification also uses the value of the `totalSupply` variable.
  The value of this variable is not decremented here.
  At the same time, the value of the variable `totalSupply` is used 7 times for calculations in this smart contract:
  BasketFacet.sol.

RECOMMENDATION

This problem needs to be corrected so that the calculations would be correct.

CLIENT'S COMMENTARY

- The total supply is incremented when minting at LibERC20.sol#L18
- The total supply is decreased on burning at LibERC20.sol#L26

MJR-3	Approve / TransferFrom multiple withdrawal attack
File	ERC20Facet.sol
Severity	Major
Status	**Fixed** at **d05aa266**

DESCRIPTION

This known issue is documented here: https://github.com/ethereum/EIPs/issues/738.

Summary / example of the attack:

1. Alice allows Bob to transfer N of Alice's tokens (N>0) by calling approve method on Token smart contract passing Bob's address and N as method arguments
2. After some time, Alice decides to change from N to M (M>0) the number of Alice's tokens Bob is allowed to transfer, so she calls approve method again, this time passing Bob's address and M as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls transferFrom method to transfer N Alice's tokens somewhere
4. If Bob's transaction is be executed before Alice's transaction, then Bob will successfully transfer N Alice's tokens and will gain an ability to transfer another M tokens
5. Before Alice noticed that something went wrong, Bob calls transferFrom method again, this time to transfer M Alice's tokens.

So, Alice's attempt to change Bob's allowance from N to M (N>0 and M>0) made it possible for Bob to transfer N+M of Alice's tokens, while Alice never wanted to allow so many of her tokens to be transferred by Bob.

RECOMMENDATION

We recommend adding additional sanity checks to the `approve()` function and using the `increaseAllowance()` / `decreaseAllowance ()` workaround functions:
`require((_amount == 0) || (allowed[msg.sender][_spender] == 0));`

CLIENT'S COMMENTARY

Doing these checks in the normal approve methods causes behaviour which is unexpected for some DeFi protocols and not strictly part of the ERC20 standard. I'll add increaseAllowance and decreaseAllowance functions.

MJR-4	Gas overflow during iteration (DoS)
File	PieFactoryContract.sol CallFacet.sol BasketFacet.sol
Severity	Major
Status	**Fixed** at **d05aa266**, **09690d34**

DESCRIPTION

Each iteration of the cycle requires a gas flow.
A moment may come when more gas is required than it is allocated to record one block. In this case, all iterations of the loop will fail.
Affected lines:

- PieFactoryContract.sol#L72
- CallFacet.sol#L43
- CallFacet.sol#L67
- CallFacet.sol#L81
- BasketFacet.sol on lines 45, 126, 158, 274, 296

RECOMMENDATION

We recommend adding a check for the maximum possible number of elements of the arrays.

CLIENT'S COMMENTARY

We will add a max length for the targets. Since every call can have a widely varying gas cost, you cannot really make sensible assumptions on what the max number of calls should be.

MJR-5	Global Protection of malicious backward calls in raw methods execution
File	LendingManager.sol RSIManager.sol#94 RSIManager.sol#131
Severity	Major
Status	Fixed at 7b548e30

DESCRIPTION

At the lines: LendingManager.sol#L44, LendingManager.sol#L70, LendingManager.sol#L102, LendingManager.sol#L114, RSIManager.sol#94, RSIManager.sol#131 anything can happen in target methods execution.

We expect one-way class methods execution-graph like:

```
A.method1 -> B.method2 -> C.method3
```

So it is better to ensure it.

RECOMMENDATION

With growing complexity and cross-dependency of SmartContracts such mistaken or malicious case can occure and it will not be easy to track. So it is better to place protection against unexpected calls, e.g. place a `totalCallsCount` class-level variable. Increase it every time, any class-method was called.
And do something like:

```
    totalCallsCountBefore = totalCallsCount;
    call(target, bytes)
    require(totalCallsCount == totalCallsCountBefore)
```

CLIENT'S COMMENTARY

It's a good catch. We'll fix it.

2.3 WARNING

WRN-1	No validation of the address parameter value in contract constructor
File	LendingLogicAave.sol LendingLogicCompound.sol LendingManager.sol RSIManager.sol
Severity	Warning
Status	Fixed at bd652b27, 37e9bed5, 4fdbfd78, f5bddc2e

DESCRIPTION

The variable is assigned the value of the constructor input parameter. But this parameter is not checked before this. If the value turns out to be zero, then it will be necessary redeploy the contract, since there is no other functionality to set this variable.

At the line LendingLogicAave.sol#L16 the `lendingPool` variable is set to the value of the `_lendingPool` input parameter.

At the line LendingLogicCompound.sol#L16 the `lendingRegistry` variable is set to the value of the `_lendingRegistry` input parameter.

At the line LendingManager.sol#L24 the `lendingRegistry` variable is set to the value of the `_lendingRegistry` input parameter.

At the line LendingManager.sol#L25 the variable `basket` is assigned the value of the input parameter `_basket`.

At the line RSIManager.sol#L31 values of the following variables are not checked: `_assetShort`, `_assetLong`, `_priceFeed`, `_basket`, `_synthetix`.

RECOMMENDATION

In all the cases, it is necessary to add a check of the input parameter to zero before initializing the variables.

CLIENT'S COMMENTARY

We'll add check if the a parameters are not zero.

WRN-2	No validation of the address parameter value in function before using this parameter in access modifiers
File	LendingRegistry.sol PieFactoryContract.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

It is possible that parameter value will be equal to zero. Then the work of the smart contract will be blocked.

At the line LendingRegistry.sol#L33 for the `_wrapped` variable in the `setWrappedToProtocol()` function.

At the line LendingRegistry.sol#L43 for the variable `_underlying` in the `setWrappedToUnderlying ()` function.

At the linehttps://github.com/pie-dao/ExperiPie/blob/facf3c246d9c43f5b1e0bad7dc2b0a9a2a2393c5/contracts/callManagers/LendingManager/LendingRegistry.sol#L53 for the `_logic` variable in the `setProtocolToLogic()` function.

At the line LendingRegistry.sol#L64 for the variable `_underlying` and `_wrapped` in the `setUnderlyingToProtocolWrapped()` function.

At the line PieFactoryContract.sol#L35 for the `_controller` variable in the `setDefaultController ()` function.

RECOMMENDATION

It is necessary to add a check of the parameter value to zero before initializing the variables.

CLIENT'S COMMENTARY

This is a feature. It effectively allows us to disable a specific token in the lending adapter. It also allows us to disable a token in the lending registry and to create pools which are finalized and cannot be changed after it.

WRN-3	Possible loss of tokens
File	LibERC20.sol
Severity	Warning
Status	Fixed at **e80418a8**

DESCRIPTION

LibERC20.sol#L17
If the value of the `_to` parameter is equal to zero, then the tokens will be lost, because they cannot be disposed of.
However, the value of the `es.totalSupply` variable will be incremented.

RECOMMENDATION

In the `mint` function, we recommend adding a check for the `_to` parameter for zero.

```
require(_to != address(0), "mint to the zero address
```

CLIENT'S COMMENTARY

We'll add this check.

WRN-4	Invalid deletion of an array element
File	PieFactoryContract.sol
Severity	Warning
Status	Fixed at 32df7957

DESCRIPTION

- First, for the `removeFacet()` function defined at PieFactoryContract.sol#L39, the value of the `_index` parameter is not checked. If its value turns out to be greater than the number of array elements, then an interrupt will occur due to going out of bounds of the array.

- Second, on line 41, the array element with the number `_index` is assigned to the value of the last array element. Then the last element is removed. Thus, the integrity of the array is violated, because the elements are shuffled and not in the order they were added to the array.

RECOMMENDATION

We recommend that you redo the logic of this function.
An example of the correct implementation can be found here:
https://ethereum.stackexchange.com/questions/1527/how-to-delete-an-element-at-a-certain-index-in-an-array
But keep in mind that when implementing a cycle, you must set a limit on the maximum number of its iterations.

CLIENT'S COMMENTARY

- We'll check if the index is within the length of the array -1
- The order of the array is not important in this case. Removing the element like this is the most gas efficient way with the least amount of code complexity.

WRN-5	The function should return a variable, but this is not done
File	ERC20Facet.sol
Severity	Warning
Status	Fixed at 5a5c9de9

DESCRIPTION

ERC20Facet.sol#L75
The `transferFrom()` function should return a boolean variable, but this is not in the source code.

RECOMMENDATION

We recommend adding the return value of this function.

CLIENT'S COMMENTARY

It's a good catch, we'll return true.

WRN-6	It is necessary to check the correctness of the variable value
File	LibERC20.sol ERC20Facet.sol
Severity	Warning
Status	Fixed at 65868cce, 32035cd1

DESCRIPTION

- At the line LibERC20.sol#L25 the `burn()` function decreases the value of the amount of `es.balances` tokens for the `_from` wallet. We recommend adding a check of the current value of the `_from` variable for zero: `require(_from != address(0), "burn from the zero address");`
- At the line ERC20Facet.sol#L57 the `approve()` function gives a permission to dispose of own tokens for the `_spender` wallet. We recommend adding a check of the current value of the `_spender` variable to zero: `require(_spender != address(0), "approve to the zero address");`
- At the line ERC20Facet.sol#L79 in the `transferFrom()` function, there is a decrease in the value of the amount tokens allowed to the sender `msg.sender` for the `_from` wallet. We recommend adding a check of the current value of the `_from` variable for zero: `require(_from != address(0), "transfer from the zero address");`
- At the line ERC20Facet.sol#L115 in the `_transfer ()` function, the value of the amount of tokens for the `_from` wallet is decreasing. We recommend adding a check of the current value of the `_from` variable for zero: `require(_from != address(0), "transfer from the zero address");`

RECOMMENDATION

We recommend adding a the particular checks.

CLIENT'S COMMENTARY

- This is a feature as it allows us to recover tokens by burning them from the zero address and then minting them to another address.
- Fixed
- Fixed
- Checked in above function.

WRN-7	Possible incorrect operation of the function
File	BasketFacet.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the lines BasketFacet.sol#L200 and BasketFacet.sol#L209. The result of the function depends on the value of `block.timestamp`. But the value `block.timestamp` is set by the miner. This may result in an incorrect result. According to the specification Block-Protocol-2.0.md, the miner can shift `block.timestamp` up to 900 seconds. If the range is greater, then you are safe.

RECOMMENDATION

We recommend that you do not use `block.timestamp` or calculate the difference between the two values so that it is greater than 900.

CLIENT'S COMMENTARY

Blocks in ethereum always have an increasing block.timestamp and the possible shift of 900 seconds does not cause any unexpected behaviour.

WRN-8	Too open rebalance method
File	RSIManager.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

This is potentially dangerous because of potential manipulations on the market: RSIManager.sol#L50

RECOMMENDATION

We suggest adding some restrictions for the caller account.

CLIENT'S COMMENTARY

This specific manager uses synthetix which has its own frontrunning protection by settling trades after 3 minutes.

WRN-9	`maxCap` can go under `totalSupply`
File	BasketFacet.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

If the supply is bigger than new `maxCap`: BasketFacet.sol#L242

RECOMMENDATION

We recommend to add `require(totalSupply <= _maxCap, '...')`.

CLIENT'S COMMENTARY

Setting the `maxCap` below the total supply allows us to only allow withdraws. This can be used when sunsetting an ExperiPie.

WRN-10	ERC20Face initialize can be called several times
File	ERC20Facet.sol
Severity	Warning
Status	Fixed at edeee96a

DESCRIPTION

It is not logical and potentially can cause misunderstanding and mistakes:
ERC20Facet.sol#L16

RECOMMENDATION

We suggest adding the following check:

```
require(!finalized, '...')
```

CLIENT'S COMMENTARY

Although it can only be called by the contract owner which is assumed to be non-malicious, it's a good recommendation to add this check.

WRN-11	Require success in search over tokens using for-loop
File	BasketFacet.sol
Severity	Warning
Status	Fixed at d7eb866e

DESCRIPTION

If a token was not found, it means that something unexpected happen, moreover in that case we also emit event that the token was removed, but it's not true: BasketFacet.sol#L52

RECOMMENDATION

We recommend to fail the call or not to emit a particular event.

CLIENT'S COMMENTARY

Good catch. Will revert on faillure.

WRN-12	Exit pool potentially is impossible
File	BasketFacet.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the lines BasketFacet.sol#L163 and BasketFacet.sol#L179
Due to `require(tokenBalance.sub(tokenAmount) >= MIN_AMOUNT, "TOKEN_BALANCE_TOO_LOW");` it is impossible to leave the pool completely, so some funds will be locked in the pool forever.

It is not clear why there should be such unsolvable problems if you want just to "take all your tokens and leave".

RECOMMENDATION

Probably `MIN_AMOUNT` is not 0 because of the division by `totalSupply` in some places. At least this moment should be covered in source code somewhere.

CLIENT'S COMMENTARY

Token amounts going below the minimum amount might cause issues with the calculations during join and exit that's why a minimum is enforced. In most cases it should not be an issue and if it is, the tokens can be salvaged through the `callFacet`.

2.4 COMMENTS

CMT-1	Duplicate code
File	CallFacet.sol
Severity	Comment
Status	Fixed at a0403f51

DESCRIPTION

Duplicate code, i.e. using the same code structures in several places. Combining these structures will improve your code. The use of duplicate code structures impairs the perception of the program logic and can easily lead to errors in subsequent code edits. Duplicate code violates SOLID (single responsibility, open-closed, Liskov substitution, interface segregation и dependency inversion) software development principles.

RECOMMENDATION

At the lines 24, 36 in file CallFacet.sol instead of this code:

```
require(msg.sender == LibDiamond.diamondStorage().contractOwner, "NOT_ALLOWED");
```

we recommend making the access modifier

```
modifier onlyContractOwner() { require(msg.sender ==
LibDiamond.diamondStorage().contractOwner, "NOT_ALLOWED"); _; }
```

CLIENT'S COMMENTARY

We'll convert the duplicate code to a modifier.

CMT-2	There is no check of the amount of ether before calling the paid function
File	CallFacet.sol
Severity	Comment
Status	**Fixed** at **fbe21739**

DESCRIPTION

At the line CallFacet.sol#L99 in the `_call` function, paid functions are called in other contracts. But on the balance of the contract there may not be enough ether to perform these functions and the `_call` function will not be executed.

RECOMMENDATION

We recommend adding such a check.

CLIENT'S COMMENTARY

We'll add the check.

CMT-3	No event registration when changing the parameters of the contract
File	LendingLogicAave.sol LendingLogicCompound.sol LendingManager.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

LendingLogicAave.sol#L20
We recommend adding the `Lend` event logging for the `lend()` function.

LendingLogicAave.sol#L42
We recommend adding logging of the `UnLend` event for the `unlend()` function.

LendingLogicCompound.sol#L19
We recommend adding the `Lend` event logging for the `lend()` function.
LendingLogicCompound.sol#L43
We recommend adding logging of the `UnLend` event for the `unlend()` function.

LendingManager.sol#L93
We recommend adding logging of the `RemoveToken` event to the `removeToken()` function.

LendingManager.sol#L105
We recommend adding logging of the `AddToken` event for the `addToken()` function.

RECOMMENDATION

We recommend adding logging.

CLIENT'S COMMENTARY

We'll add the events. These events are already emitted by either the lending manager or the experiPie itself.

CMT-4	Incorrect function logic
File	ERC20Facet.sol
Severity	Comment
Status	Fixed at 36e136c7

DESCRIPTION

At the line ERC20Facet.sol#L105 the `_transfer ()` function is designed to transfer tokens from one wallet to another wallet.
But here, for the situation, if the value of the sender's wallet turns out to be equal to zero, tokens are burned from `msg.sender` account instead of from `_from`.

RECOMMENDATION

We recommend that you redo the logic of this function so that only logic remains here,
responsible for the transfer of tokens, but not burning.

CLIENT'S COMMENTARY

It's a good catch, we'll remove the burning logic.

CMT-5	We recommend adding an additional parameter when registering an event
File	CallFacet.sol
Severity	Comment
Status	**Fixed** at **8a6dfb4f**

DESCRIPTION

At the line CallFacet.sol#L101 calling the `_call()` function performs any action with any contracts.
However, the `Call` event does not store who called the function.

RECOMMENDATION

We recommend adding one more parameter to save the value of `msg.sender`.

CLIENT'S COMMENTARY

We'll add the event parameter.

CMT-6	Unclear constants
File	RSIManager.sol BasketFacet.sol
Severity	Comment
Status	**Fixed** at **05090894**

DESCRIPTION

For example, it is absolutely unclear what `30 * 10**18` means. At the following lines: RSIManager.sol#L54, RSIManager.sol#L58, RSIManager.sol#L78, BasketFacet.sol#L97, BasketFacet.sol#L107, BasketFacet.sol#L124, BasketFacet.sol#L139

RECOMMENDATION

Constants should be described on the top of the class once and have descriptive names and comments if need.

CLIENT'S COMMENTARY

It's a good catch. We'll convert to clearly defined constants or variables assigned in the constructor.

CMT-7	Use `totalSupply <= maxCap` instead of `<`
File	BasketFacet.sol
Severity	Comment
Status	**Fixed** at **6468c4ed**

DESCRIPTION

It is more rational to use `<=` instead of `<` at the line BasketFacet.sol#L122.

RECOMMENDATION

We suggest to replace operator to `<=` instead of `<`.

CLIENT'S COMMENTARY

It's a good catch. We'll incorporate the suggested check.

3.ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

Ethereum

Cosmos

EOS

Substrate

TECH STACK

Python

Solidity

Rust

C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes