OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 20	2021-11-20	Zapmore, Plemonade	Audit Final

Audit Notes

Audit Date	2021-11-07 - 2021-11-20
Auditor/Auditors	Plemonade, Mechwar
Auditor/Auditors Contact Information	contact@obeliskauditing.com
Notes	Specified code and contracts are audited for security flaws. UI/UX (website), logic, team, and tokenomics are not audited.
Audit Report Number	OB574774459

Disclaimer

This audit is not financial, investment, or any other kind of advice and is for informational purposes only. This report is not a substitute for doing your own research and due diligence. Obelisk is not responsible or liable for any loss, damage, or otherwise caused by reliance on this report for any purpose. Obelisk has based this audit report solely on the information provided by the audited party and on facts that existed before or during the audit being conducted. Obelisk is not responsible for any outcome, including changes done to the contract/contracts after the audit was published. This audit is fully objective and only discerns what the contract is saying without adding any opinion to it. The audit is paid by the project but neither the auditors nor Obelisk has any other connection to the project and has no obligations other than to publish an objective report. Obelisk will always publish its findings regardless of the outcome of the findings. The audit only covers the subject areas detailed in this report and unless specifically stated, nothing else has been audited. Obelisk assumes that the provided information and material were not altered, suppressed, or misleading. This report is published by Obelisk, and Obelisk has sole ownership of this report. Use of this report for any reason other than for informational purposes on the subjects reviewed in this report including the use of any part of this report is prohibited without the express written consent of Obelisk.

Obelisk Auditing

Defi is a relatively new concept but has seen exponential growth to a point where there is a multitude of new projects created every day. In a fast-paced world like this, there will also be an enormous amount of scams. The scams have become so elaborate that it's hard for the common investor to trust a project, even though it could be legit. We saw a need for creating high-quality audits at a fast phase to keep up with the constantly expanding market. With the Obelisk stamp of approval, a legitimate project can easily grow its user base exponentially in a world where trust means everything. Obelisk Auditing consists of a group of security experts that specialize in security and structural operations, with previous work experience from among other things, PricewaterhouseCoopers. All our audits will always be conducted by at least two independent auditors for maximum security and professionalism.

As a comprehensive security firm, Obelisk provides all kinds of audits and project assistance.

Audit Information

The auditors always conducted a manual visual inspection of the code to find security flaws that automatic tests would not find. Comprehensive tests are also conducted in a specific test environment that utilizes exact copies of the published contract.

While conducting the audit, the Obelisk security team uses best practices to ensure that the reviewed contracts are thoroughly examined against all angles of attack. This is done by evaluating the codebase and whether it gives rise to significant risks. During the audit, Obelisk assesses the risks and assigns a risk level to each section together with an explanatory comment. Take note that the comments from the project team are their opinion and not the opinion of Obelisk.

Table of Contents

Project Information

Name	PearZap
Description	High yield farms & pools on the Polygon, Binance Smart Chain & Fantom Chain. - Crosschain platform
Website	https://pearzap.com/
Contact	@AndrewPearZap
Contact information	@AndrewPearZap on TG
Token Name(s)	N/A
Token Short	N/A
Contract(s)	See Appendix A
Code Language	Solidity
Chain	Multichain

The PearZap team worked swiftly to resolve all outstanding issues of relevance.

Obelisk was commissioned by PearZap on the 3d of November 2021 to conduct a comprehensive audit of PearZaps' Dex contracts. The following audit was conducted between the 7th of November 2021 and the 20th of November 2021. Two of Obelisk's security experts went through the related contracts manually using industry standards to find if any vulnerabilities could be exploited either by the project team or users.

The contracts were not deployed during the audit which means that the project team could implement fixes to issues found without problem. During the audit of the supplied contracts, we found two issues of relevance. The first one involved a calculation error that was deemed a critical issue. The project team implemented a fix.

The second finding was a low-risk finding regarding the fee assumption, which was corrected by the project team.

After the contracts were audited, the project team deployed the contracts and Obelisk did an on-chain analysis in order to establish that the audited contracts are the same ones deployed and that the deployment is correct. It was concluded that this is the case. One note is that the swap fee can be changed by an EOA wallet between 0%-0.3%, and as long as the stated fee correlates to the actual fee, this is not a problem, but adding this function behind a timelock would benefit the users to react on a fee change.

The informational findings are good to know while interacting with the project but don't directly damage the project in its current state, hence it's up to the project team if they deem that it's worth solving these issues.

The team has not reviewed the UI/UX, logic, team, or tokenomics of the PearZap project**.**

Please read the full document for a complete understanding of the audit.

Summary Table

Finding	ID	Severity	Status
Incorrect Calculation Of Reserve Multiples	#0001	High Risk	Closed
Incorrect Fee Assumption	#0002	Low Risk	Closed
Unnecessary Call	#0003	Informational	Closed
Different Versions Of Solidity	#0004	Informational	Open
Fee Setter/Receiver Is EOA	#0005	Informational	Open
Init Code Hash Changed	#0006	Informational	Closed

Findings

Manual Analysis

Incorrect Calculation Of Reserve Multiples

FINDING ID	#0001
SEVERITY	High Risk
STATUS	Closed
LOCATION	PearzapPair.sol -> 199-204

```
1     { // scope for reserve{0,1}Adjusted, avoids stack too
  deep errors
2     uint _swapFee = swapFee;
3     uint balance0Adjusted =
  (balance0.mul(10000).sub(amount0In.mul(_swapFee)));
4     uint balance1Adjusted =
  (balance1.mul(10000).sub(amount1In.mul(_swapFee)));
5     require(balance0Adjusted.mul(balance1Adjusted) >=
  uint(_reserve0).mul(_reserve1).mul(1000**2), 'PearzapSwap:
  K');
6     }
```

DESCRIPTION	The swap function optimistically transfers tokens to a user. This is used for lending tokens in a flashloan. However, the adjusted balance incorrectly calculates the balances to be one hundred times larger than the balance of the contract. This leads to an attacker only needing to return 1% of the flashloan and being able to steal 99% of tokens on a given swap.
RECOMMENDATION	Make sure the final check assumes the same accuracy so that the k formula is correctly calculated. The denominator should match instead of one being 10^3 and the other 10^4.
RESOLUTION	The project team has implemented the recommended fix.

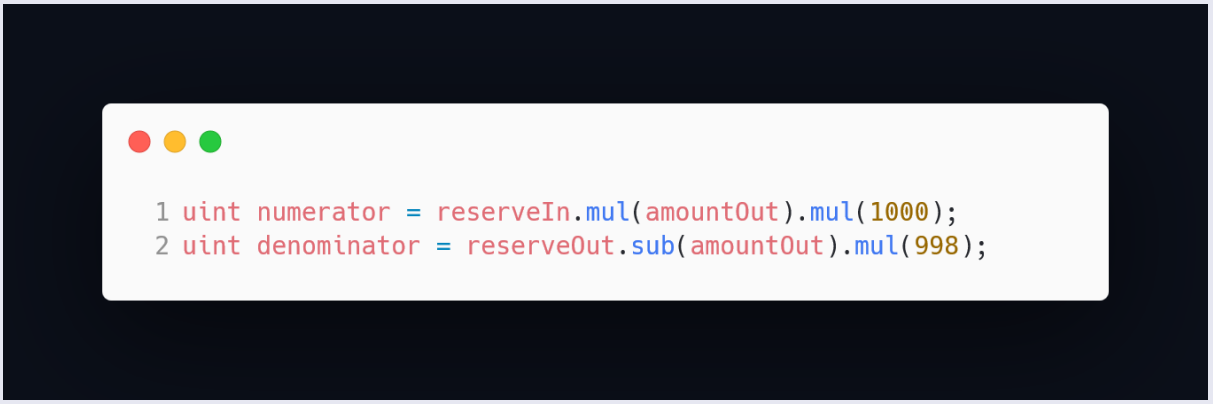
	Reviewed in commit 8a00c6dce62df20760527b587f01aea5ebda5f97

Incorrect Fee Assumption

FINDING ID	#0002
SEVERITY	Low Risk
STATUS	Closed
LOCATION	PearzapLibrary.sol -> 55-57

```
1 uint amountInWithFee = amountIn.mul(998);
2 uint numerator = amountInWithFee.mul(reserveOut);
3 uint denominator =
  reserveIn.mul(1000).add(amountInWithFee);
```

LOCATION	PearzapLibrary.sol -> 65-66



```
1 uint numerator = reserveIn.mul(amountOut).mul(1000);
2 uint denominator = reserveOut.sub(amountOut).mul(998);
```

DESCRIPTION	The *getAmountOut()* and *getAmountIn()* function calls will return the incorrect amount by assuming the swap fee is 0.2%. However, in the *PearzapPair.sol* contract the swap fee could be set to a maximum of 0.3% or a minimum of 0.01%.
RECOMMENDATION	The swap fee is changeable and should be retrieved from the *PearzapPair.sol* contract for a correct calculation. The nominator and denominator should also use 10^4 instead of 10^3 to be able to handle fees as low as 0.01%.
RESOLUTION	The project team has implemented the recommended fix.

	Reviewed in commit 8a00c6dce62df20760527b587f01aea5ebda5f97 and a6bc8ed33ff001d443428bf22908cf2d4c4991a8

Unnecessary Call

FINDING ID	#0003
SEVERITY	Informational
STATUS	Closed
LOCATION	PearzapLibrary.sol -> 39

```
1 pairFor(factory, tokenA, tokenB);
```

DESCRIPTION	The function call to *pairFor()* is unnecessary as the return value is not used.
RECOMMENDATION	Remove the function call to *pairFor()*.
RESOLUTION	The project team has implemented the recommended fix. Reviewed in commit: 8a00c6dce62df20760527b587f01aea5ebda5f97 and a6bc8ed33ff001d443428bf22908cf2d4c4991a8

Static Analysis

Different Versions Of Solidity

FINDING ID	#0004
SEVERITY	Informational
STATUS	Open
LOCATION	IERC20.sol (>=0.5.0) IPearzapCallee.sol (>=0.5.0) IPearzapERC20.sol (>=0.5.0) IPearzapFactory.sol (>=0.5.0) IPearzapPair.sol (>=0.5.0) IPearzapRouter01.sol (>=0.6.2) IPearzapRouter02.sol (>=0.6.2) IWETH.sol (>=0.5.0) Math.sol (=0.5.16) PearzapERC20.sol (=0.5.16) PearzapFactory.sol (=0.5.16) PearzapLibrary.sol (>=0.5.0) PearzapPair.sol (=0.5.16) PearzapRouter.sol (=0.6.6) SafeMath.sol (=0.5.16) TransferHelper.sol (>=0.6.0) UQ112x112.sol (=0.5.16)

DESCRIPTION	Different versions of Solidity are used.
RECOMMENDATION	Use a single version of Solidity.
RESOLUTION	N/A

On-Chain Analysis

Fee Setter/Receiver Is EOA

FINDING ID	#0005
SEVERITY	Informational
STATUS	Open
LOCATION	PearzapFactory 0x8a0239C06104dc4A7A8869cf68C6d98B96EF95DD in PearzapFactory.sol

DESCRIPTION	The *feeToSetter* can arbitrarily change the fee receiver (*feeTo*) and the swap fee (between 0%-0.3%). It is currently set to an EOA (Externally Owned Account). feeToSetter 0xF301911451E7Fdb9900a65cb7acd6DED9e915A1D feeTo 0xb4C9DAfb76FeCb2Ac072bb77c3B276d182E080a2
RECOMMENDATION	No change is required, though a timelock to allow users to react to changes could be beneficial.
RESOLUTION	N/A

Init Code Hash Changed

FINDING ID	#0006
SEVERITY	Informational
STATUS	Closed
LOCATION	PearzapRouter 0xB245F2e2A694cDf08e69671125a2605803cAb006 in PearzapLibrary.sol

DESCRIPTION	The init code hash differs from audited contracts but checks out on-chain and does not pose any issues.
RECOMMENDATION	No change is required.
RESOLUTION	N/A

Appendix A - Reviewed Documents

Document	Address
IERC20.sol	N/A
IPearzapCallee.sol	N/A
IPearzapERC20.sol	N/A
IPearzapFactory.sol	N/A
IPearzapPair.sol	N/A
IPearzapRouter01.sol	N/A
IPearzapRouter02.sol	N/A
IWETH.sol	N/A
Math.sol	N/A
PearzapERC20.sol	N/A
PearzapFactory.sol	0x8a0239C06104dc4A7A8869cf68C6d98B96EF95DD
PearzapLibrary.sol	N/A
PearzapPair.sol	N/A
PearzapRouter.sol	0xB245F2e2A694cDf08e69671125a2605803cAb006
SafeMath.sol	N/A
TransferHelper.sol	N/A
UQ112x112.sol	N/A

Revisions

Revision 1: cbdddde623a2477f77d4a3c972441d603eba9d60
Revision 2: 8a00c6dce62df20760527b587f01aea5ebda5f97
Revision 3: a6bc8ed33ff001d443428bf22908cf2d4c4991a8

Imported Contracts

Sushiswap: Canary Branch

Externally Owned Accounts

feeToSetter: 0xF301911451E7Fdb9900a65cb7acd6DED9e915A1D
feeTo: 0xb4C9DAfb76FeCb2Ac072bb77c3B276d182E080a2

Appendix B - Risk Ratings

Risk	Description
High Risk	A fatal vulnerability that can cause the loss of all Tokens / Funds.
Medium Risk	A vulnerability that can cause the loss of some Tokens / Funds.
Low Risk	A vulnerability that can cause the loss of protocol functionality.
Informational	Non-security issues such as functionality, style, and convention.

Appendix C - Finding Statuses

Closed	Contracts were modified to permanently resolve the finding.
Mitigated	The finding was resolved by other methods such as revoking contract ownership. The issue may require monitoring, for example in the case of a time lock.
Partially Closed	Contracts were updated to fix the issue in some parts of the code.
Partially Mitigated	Fixed by project specific methods which cannot be verified on chain. Examples include compounding at a given frequency.
Open	The finding was not addressed.

Appendix D - Audit Procedure

A typical Obelisk audit uses a combination of the three following methods:

Manual analysis consists of a direct inspection of the contracts to identify any security issues. Obelisk auditors use their experience in software development to spot vulnerabilities. Their familiarity with common contracts allows them to identify a wide range of issues in both forked contracts as well as original code.

Static analysis is software analysis of the contracts. Such analysis is called “static” as it examines the code outside of a runtime environment. Static analysis is a powerful tool used by auditors to identify subtle issues and to verify the results of manual analysis.

On-chain analysis is the audit of the contracts as they are deployed on the blockchain. This procedure verifies that:

- deployed contracts match those which were audited in manual/static analysis;
- contract values are set to reasonable values;
- contracts are connected so that interdependent contract function correctly;
- and the ability to modify contract values is restricted via a timelock or DAO mechanism. (We recommend a timelock value of at least 72 hours)

Each obelisk audit is performed by at least two independent auditors who perform their analysis separately.

After the analysis is complete, the auditors will make recommendations for each issue based on best practices and industry standards. The project team can then resolve the issues, and the auditors will verify that the issues have been resolved with no new issues introduced.

Our auditing method lays a particular focus on the following important concepts:

- Quality code and the use of best practices, industry standards, and thoroughly tested libraries.
- Testing the contract from different angles to ensure that it works under a multitude of circumstances.
- Referencing the contracts through databases of common security flaws.

Follow Obelisk Auditing for the Latest Information

 ObeliskOrg

 ObeliskOrg

OBELISK

Part of Tibereum Group