Pixelmon Party Squad audit report

Scope of Work

The scope of this audit is based on the material provided by the client. The goal of this audit is to ensure the following:

- Find potential exploits for the contract
- Find issues in the contract that will affect the integrity of the mint and claim processes
- Ensure the contract adheres to the business logics provided by the client

The logic of the contract will be compared to the business logic contained in the materials provided below:

- **3 contracts** were provided in total
  - **Evolution3Serum**
  - **PxPartySquad**
  - **PsWeekManager**
- **Simple contracts logic description**
- **Smart contract list that interacts with the evolution smart contract**

The contracts code was provided on 24th May 2023. Audit report was delivered on 30th May 2023. It was given in the following formats:

- **PDF document**
- **Zip archive with optimization examples**

Disclaimer

This is a limited report based on our analysis of the Smart Contract audit and performed in accordance with best practices as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the smart contract source code analyzed, the details of which are set out in this report, (hereinafter “Source Code”) and the Source Code compiling, deploying and performing the intended functions. In order to get a full view of our findings and the scope of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn’t say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions.

Summary

Contracts are well written and comply to the best practices, no errors were found in all contracts provided. However, some areas were identified where optimization can be implemented to lower the gas consumption.

Recommendations for optimization are detailed in corresponded section of this report.

Findings summary

	Critical	**Medium**	**Low**
Evolution3Serum	0	0	0
PxPartySquad	0	0	0
PsWeekManager	0	0	0
Total	0	0	0

Gas optimizations

Current average claim costs, tested as an average of 10 tx's for one user, numbers in GWEI:

1 claim in total	2 claims in total	10 claims in total
195337	434276	2345788

1. Based on the provided logic, it appears that there are no plans to increase the number of claim spots during the week. The contract has a designated time period for updating winners, followed by a claim start timestamp. As a result, it may be feasible to have individuals claim all tokens in a single contract transaction, which would lower the cost of calling the contract and allow for the reuse of certain variables.
2. By claiming all tokens at once, we can eliminate certain on-chain write operations, such as updating the `treasureTypeClaimed` variable for each user. This is unnecessary since the information doesn't need to be retained between calls, and only one transaction is required. Instead, we can store this information as a memory array and search through it whenever we need to determine if a user has claimed a particular type of treasure. This approach is more cost-effective than using the `treasureTypeClaimed` variable, especially when a user has multiple treasure types to claim. The cost of this approach will increase with the number of treasure types a user can claim, but this is a relatively rare occurrence.
3. To execute all claims in a single transaction, some adjustments need to be made to the randomization logic. This involves introducing a unique variable for each claim iteration to ensure that the outcome is not always the same. To achieve this, the iteration index can be used in the `claim` function, while `weekNumber` can be utilized in the `updateWeeklyWinners` function. These modifications will ensure that the block variables and `msg.sender` are not the only factors determining the outcome of the lottery.

Claim Costs Table with Changes Above Implemented:

1 claim in total	2 claims in total	10 claims in total
198978	295266	1030181

This will allow reducing the gas costs by ~66% for a 10-token claim and ~32% when people claim two tokens at once. However, it will slightly increase the price of a 1-token claim transaction.

4. The `claimTreasure` function also calls the `psChainlinkManagerContract.isSignerVerifiedFromSignature`, which adds additional costs for calling the external contract.

   Moving this function to `PxPartySquad` will follow the good logical division practice and also decrease the one-time call cost to an average of **168,422** GWEI for a 1-token claim (~14% less than the initial one).

   Note: That's possible only in case if you reduce the `PxPartySquad` contract bytecode size, which can be achieved using p.3 from this document.

Final claim costs table with all changes implemented:

1 claim in total	2 claims in total	10 claims in total
168422	284654	1015233

That's all changes that can be performed to significantly decrease the gas costs without reworking the contract's structure, note that the real gas savings may vary from case to case depending on the amount of `Treasure` allocated for the week, average tokens to be claimed per user, etc.

Everything above was tested with 10 types of `Treasure` with 1:1 proportion of ERC1155 and ERC721.

You may also want to revise the data you store on chain and if you really need it.

Currently, the contract uses Chainlink randomization to select winners only, but not for determining the prizes. Instead, prize randomization is done fully on-chain. This approach not only lacks true randomness and has the potential to be predicted, but it also increases the gas costs.

To reduce the gas costs even further, you could consider moving the prize randomization process to your backend.

Notes:
You may find a code sample in attachment, **please note that's just a reference for the changes described above and should not be used in production without proper refining due it's lack of checks.**

Please also note that `specialTreasure` claim was omitted in provided examples, do not forget to add it in case if you will use it as reference for the adjustments.