shieldify

Honeypot Finance

SECURITY REVIEW

Date: 27 November 2024

CONTENTS

1. **About Shieldify Security** 3
2. **Disclaimer** 3
3. **About Honeypot Finance** 3
4. **Risk classification** 3
   - 4.1 Impact 3
   - 4.2 Likelihood 4
5. **Security Review Summary** 4
   - 5.1 Protocol Summary 4
   - 5.2 Scope 4
6. **Findings Summary** 5
7. **Findings** 5

1. About Shieldify

Positioned as the first hybrid Web3 Security company, Shieldify shakes things up with a unique subscription-based auditing model that entitles the customer to unlimited audits within its duration, as well as top-notch service quality thanks to a disruptive 6-layered security approach. The company works with very well-established researchers in the space and has secured multiple millions in TVL across protocols, also can audit codebases written in Solidity, Rust, Go, Vyper, Move and Cairo.

Learn more about us at shieldify.org.

2. Disclaimer

This security review does not guarantee bulletproof protection against a hack or exploit. Smart contracts are a novel technological feat with many known and unknown risks. The protocol, which this report is intended for, indemnifies Shieldify Security against any responsibility for any misbehavior, bugs, or exploits affecting the audited code during any part of the project's life cycle. It is also pivotal to acknowledge that modifications made to the audited code, including fixes for the issues described in this report, may introduce new problems and necessitate additional auditing.

3. About Honeypot Finance

Honeypot Finance acts as a Proof-of-Liquidity (PoL) Accelerator that unites a fair launchpad, Dreampad, and a secure DEX, Henlo DEX:

- Dreampad supports our innovative Fair Token Offering (FTO) model and both Fjord Foundry's LBP and Fixed Price Sales to ensure successful and sustainable token launches for projects.
- Henlo DEX is powered by the A2MM protocol, enables automatic liquidity deployment and maximizes liquidity utilization for traders and investors.
- Pot2Pump combines all the advantages of the FTO model, with specific adjustments for meme tokens and protection against bots. Their PoL Accelerator aims to embody the aspirations of the Berachain community by providing a comprehensive suite of DeFi tools. These tools are crafted to empower individuals with financial autonomy. Their unique flywheel operates on a community-driven paradigm, fostering an ecosystem of protocols and validators where increased engagement leads to enhanced liquidity.

Learn more about Honeypot(Pot2Pump)'s concept and the technicalities behind it here.

4. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** – results in a significant risk for the protocol's overall well-being. Affects all or most users

- **Medium** – results in a non-critical risk for the protocol affects all or only a subset of users, but is still unacceptable
- **Low** – losses will be limited but bearable – and covers vectors similar to griefing attacks that can be easily repaired

4.2 Likelihood

- **High** – almost certain to happen and highly lucrative for execution by malicious actors
- **Medium** – still relatively likely, although only conditionally possible
- **Low** – requires a unique set of circumstances and poses non-lucrative cost-of-execution to rewards ratio for the actor

5. Security Review Summary

The security review lasted 3 days with a total of 96 hours dedicated by 4 researchers from the Shieldify team.

Overall, the code is well-written. The audit report contributed by identifying one Low-severity issue for a misleading getter function, along with other informational findings.

The Honeypot team has done an excellent job on the development, demonstrating both expertise and dedication. Their responsiveness to questions from our team of researchers has been outstanding, showcasing their commitment to collaboration and excellence.

5.1 Protocol Summary

Project Name	**Honeypot**
Repository	HoneyPot-MemeLaunchPad-Algebra
Type of Project	DeFi, Proof-of-Liquidity (PoL) Accelerator
Audit Timeline	3 days
Review Commit Hash	2c0dceb9023ff6e4565425f8ecd1d8cf72b359a9
Fixes Review Commit Hash	68e1646f1cd192f85a966eaa2d1e2b4417c65682

5.2 Scope

The following smart contracts were in the scope of the security review:

File	nSLOC
contracts/hpot/core/pot2pump/Pot2PumpFactory.sol	91
contracts/hpot/core/pot2pump/Pot2PumpPair.sol	73
contracts/hpot/core/pot2pump/interfaces/IPot2PumpFactory.sol	16
contracts/hpot/core/pot2pump/interfaces/IPot2PumpPair.sol	6

contracts/hpot/core/berascout/BeraScoutFactory.sol	154
contracts/hpot/core/berascout/BeraScoutPair.sol	153
contracts/hpot/core/berascout/interfaces/IBeraScoutFactory.sol	31
contracts/hpot/core/berascout/interfaces/IBeraScoutPair.sol	5
Total	**529**

6. Findings Summary

The following number of issues have been identified, sorted by their severity:

- **Low** issues: **1**
- **Informational** issues: **2**

ID	Title	Severity	Status
[L-01]	`_isCycleEnded()` Should Also Return True if The Pair is Successfully Deployed or Failed	Low	Fixed
[I-01]	`pairs[launchedToken]` Is Set Twice in `BaseFactory`	Informational	Fixed
[I-02]	Potential Risks Involving `depositRaisedToken()` Function	Informational	Acknowledged

7. Findings

[L-01] `_isCycleEnded()` Should Also Return True if The Pair is Successfully Deployed or Failed

Severity

Low Risk

Description

In `BeraScoutPair.sol`, the cycle is set as ended if the current time is past the block timestamp and `PairState == Status.Processing`.

File: contracts/hpot/core/berascout/BeraScoutPair.sol#L185

```
function _isCycleEnded() public view returns (bool) {
  return block.timestamp > endTime && PairState == Status.Processing;
}
```

If the `PairState` is `Status.Failed` or `Status.Success`, it also means the cycle has ended but `_isCycleEnded()` will return false, which is misleading.

Impact

Misleading getter function.

Recommendation

In case the cycle fails or is successful, it also means the cycle has ended as well:

```
function _isCycleEnded() public view returns (bool) {
+ if (PairState == Status.Failed || PairState == Status.Success) return
    true;
  return block.timestamp > endTime && PairState == Status.Processing;
}
```

Team Response

Fixed.

[I-01] `pairs[launchedToken]` Is Set Twice in `BaseFactory`

Severity

Informational

Description

When `_setupTokenAndPair()` is called in BaseFactory, `pairs[launchedToken]` is set two times.

```
  BaseLaunchToken(launchedToken).mint(pair, launchedTokenSupply);

> pairs[launchedToken] = pair;
> pairs[launchedToken] = pair;

  allPairs.push(pair);
```

Recommendation

Remove one instance of `pairs[launchedToken] = pair;`.

Team Response

Fixed.

[I-02] Potential Risks Involving `depositRaisedToken()` Function

Severity

Informational

Description

In `Pot2PumpPair.sol` and `BeraScoutPair.sol`, the `depositRaisedToken()` function requires tokens to be deposited into the pair contract before updating the depositor's `raisedTokenDeposi` mapping. This function is similar to the UniswapV2 `pair` and `mint()` functions.

Potential risks involved:

1. Contracts that interact with `depositRaisedToken()` must ensure that the deposited amount equals the `amount` parameter when calling `depositRaisedToken(address depositor, uint256 amount)`.
2. User that wants to interact with the contract directly must know that they have to deposit tokens into the pair contract and simultaneously call `depositRaisedToken()`, otherwise someone can frontrun the `depositRaisedToken()` call and claim the `raisedTokenDeposit` for themselves.
3. Note that if external contracts interact with `depositRaisedToken()` and set a fee, the fee can be circumvented by the user directly calling `depositRaisedToken()`.

Team Response

Acknowledged.

shieldify

Thank you!