Poof Policy - grandmAIsters Audit Report

Prepared by KupiaSec

Version 1.0

Auditors

KupiaSec

Aug 11th, 2025

Contents

1 About KupiaSec

KupiaSec is a team of Web3 security experts that operates with transparency and a meritocratic spirit.

KupiaSec executes the modified **MPA** model for the Private Audits, a.k.a. **Solo Audit by a Lead + Internal Competition + Mitigation Review**.

- Solo Audit by a Lead

  KupiaSec assigns a senior auditor as a Lead Auditor based on the protocol category. The Lead Auditor is responsible for the first phase and will be the main point of contact for the client. The Lead Auditor shares the analysis and findings with the team.

- Internal Competition

  KupiaSec assigns 5~7 assist auditors to conduct the second phase. The auditors compete to find the most issues and the best solutions. This phase ensures the protocol goes through a rigorous review process by "many eyes" in a competitive environment.

- Mitigation Review

  After the protocol team has fixed the issues, KupiaSec conducts a final review to ensure all the issues are resolved.

2 Disclaimer

The KupiaSec team makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4 Protocol Summary

Poof team developed a Game protocol using Tarobase and Trying to judge whether an extra safeguard that can be added post launch or if the idea is that apps launched on poof should be policy audited before mainnet. Context for the audit:

- Users mint tokens tied to AI game agents, each with bonding curve funded with SOL
- Before each game, a small amount of SOL is locked from both agents' curves
- At game resolution, SOL shifts from the loser's curve to the winner's curve
- Only the token creator can claim SOL from their agent's curve

5 Audit Scope

https://pastebin.com/4j6uFhT3

Summary of Commits

Project Name	Poof Policy - grandmAIsters
Repository	4j6uFhT3
Initial Commit	N/A...
Mitigation Commit	N/A...

6 Executive Summary

KupiaSec executed a modified Multi-Phase Audit model, a.k.a. **Solo Audit by a Lead + Internal Competition + Mitigation Review**.

Auditor conducted the audit as the Lead Auditor and 5 auditors competed in the second phase.

Execution Timeline

Phase-1: Audit by a Lead	Aug 4th - Aug 5th
Phase-2: Internal Competition	Aug 5th - Aug 7th
Initial Report Delivery	Aug 8th, 2025
Phase-3: Mitigation Review	N/A
Final Report Delivery	Aug 11th, 2025

Issues Found

Critical Risk	0
High Risk	4
Medium Risk	0
Low Risk	2
Informational	1
Gas Optimizations	0
Total Issues	7

Summary of Findings

[H-1] `gameTokens` is updated incorrectly when token creator claims the reward in the partial claim	Resolved

[H-2] `virtualTokenReserves` is updated incorrectly when token creator claims the reward in the full claim	Resolved
[H-3] `gameTokens` and `virtualTokenReserves` are not updated correctly when SOL is withdrawn from the bonding curve during the resolution of the game and claim of the reward	Resolved
[H-4] No limitation of inital SOL amount in the creation of agent	Resolved
[L-1] Some fees are not accounted	Acknowledged
[L-2] pecision loss	Acknowledged
[I-1] Use the Pump.fun bonding curve parameters	Acknowledged

7 Findings

7.1 High

7.1.1 `gameTokens` is updated incorrectly when token creator claims the reward in the partial claim

Description: A token creator can claim their reward when `gameTokens < 0`. A smaller negative value indicates that token creators can claim greater rewards. After claiming their rewards, the `gameTokens` should be increased. However, it deducts the clamied tokens from the original `gameTokens` which make it smaller negative.

```
&& @DocumentPlugin.updateField(
        /agents/$agentId,
        'gameTokens',
        get(/agents/$agentId).gameTokens - @BondingCurvePlugin.getTokensOutProduct(
                            (get(/agents/$agentId).actualSolReserves -
                            ↪  get(/agents/$agentId).lockedLiquidity),
                            get(/agents/$agentId).virtualSolReserves,
                            get(/agents/$agentId).virtualTokenReserves,
                            get(/agents/$agentId).actualSolReserves,
                            get(/agents/$agentId).actualTokenReserves) )
```

Impact: High. Malicious token creators can claim more rewards than they are entitled to.

Recommended Mitigation: It is recommended to modify the code as follows:

```
&& @DocumentPlugin.updateField(
        /agents/$agentId,
        'gameTokens',
-       get(/agents/$agentId).gameTokens - @BondingCurvePlugin.getTokensOutProduct(
+       get(/agents/$agentId).gameTokens + @BondingCurvePlugin.getTokensOutProduct(
                            (get(/agents/$agentId).actualSolReserves -
                            ↪  get(/agents/$agentId).lockedLiquidity),
                            get(/agents/$agentId).virtualSolReserves,
                            get(/agents/$agentId).virtualTokenReserves,
                            get(/agents/$agentId).actualSolReserves,
                            get(/agents/$agentId).actualTokenReserves) )
```

GrandmAIsters:: Fixed

KupiaSec: Confirmed the fix

7.1.2 `virtualTokenReserves` is updated incorrectly when token creator claims the reward in the full claim

Description: A token creator can claim their reward when `gameTokens < 0`. After claiming their rewards, the `virtualTokenReserves` should be increased as it should receive tokens. However, it adds `gameTokens` which is smaller than `0`.

```
&& @DocumentPlugin.updateField(
    /agents/$agentId,
    'virtualTokenReserves',
    get(/agents/$agentId).virtualTokenReserves + get(/agents/$agentId).gameTokens)
```

Impact: High. The bonding curve operates differently than expected. Loss of funds to traders.

Recommended Mitigation: It is recommended to modify the code as follows:

```
&& @DocumentPlugin.updateField(
    /agents/$agentId,
    'virtualTokenReserves',
-   get(/agents/$agentId).virtualTokenReserves + get(/agents/$agentId).gameTokens)
+   get(/agents/$agentId).virtualTokenReserves - get(/agents/$agentId).gameTokens)
```

GrandmAIsters: Fixed

KupiaSec: Confirmed the fix

7.1.3 `gameTokens` and `virtualTokenReserves` are not updated correctly when SOL is withdrawn from the bonding curve during the resolution of the game and claim of the reward

Description: When games are resolved, if an agent receives SOL as a reward from another agent, the calculation of `gameTokens` is performed correctly based on the bonding curve. This is because SOL is deposited into the bonding curve, and `gameTokens` should be calculated using the `getTokensOutProduct()` function, similar to a buy operation.

```
    && @DocumentPlugin.updateField(
          /agents/@newData.blackAgentId,
          'gameTokens',
          get(/agents/@newData.blackAgentId).gameTokens
            - @BondingCurvePlugin.getTokensOutProduct(
                ( (@newData.blackAgentReceivedSolAmount * 990) / 1000 ) -
                ↪  @newData.whiteAgentReceivedSolAmount,
                get(/agents/@newData.blackAgentId).virtualSolReserves,
                get(/agents/@newData.blackAgentId).virtualTokenReserves,
                get(/agents/@newData.blackAgentId).actualSolReserves,
                get(/agents/@newData.blackAgentId).actualTokenReserves))
```

However, if the agent loses, it calculates the `gameTokens` incorrectly. In this case, SOL is withdrawn from the bonding curve, this operation should be treated as a sell operation. Currently, the implementation incorrectly uses the same `getTokensOutProduct()` function to calculate `gameTokens`, leadig erroneous results. So the calculation becomes incorrect. Since this amount is calculated incorrectly, 'gameTokens' and `virtualTokenReserves` are updated to incorrect value.

This also happens when token creators partially claim their rewards using `gameTokens`.

```
  && @DocumentPlugin.updateField(
        /agents/$agentId,
        'virtualTokenReserves',
        get(/agents/$agentId).virtualTokenReserves + @BondingCurvePlugin.getTokensOutProduct(
                                  (get(/agents/$agentId).actualSolReserves -
                                  ↪  get(/agents/$agentId).lockedLiquidity),
                                  get(/agents/$agentId).virtualSolReserves,
                                  get(/agents/$agentId).virtualTokenReserves,
                                  get(/agents/$agentId).actualSolReserves,
                                  get(/agents/$agentId).actualTokenReserves) )
```

Also in this case, since SOL is withdrawn from the bonding curve, this operation should be treated as a sell operation.

Impact: High. The bonding curve operates differently than expected. Loss of funds to traders.

Recommended Mitigation: It is recommended to implement a new `getTokensInProduct()` function that accurately returns the amount of tokens based on the output SOL, in accordance with the bonding curve. Then, use this function instead of the `getTokensOutProduct()` function.of Output Sol.

GrandmAIsters: Fixed

KupiaSec: Confirmed the fix

7.1.4 No limitation of inital SOL amount in the creation of agent

Description: In the GrandmAIsters platform, token creators should provide at least 0.2 SOL. However, in the current implementation, there is no check for this and only pays 1% of initial SOL amount as fee.

Impact: High. There is loss of funds to the protocol. Furthermore, as attackers can easily create tokens with small amount of SOL, they can easily DoS the creation process.

Proof of Concept: Recommended Mitigation:

It is recommended to change the policy to reflect this.

```
-   "create": "(@newData.lockedLiquidity == null || get(/admins/@user.address) != null) &&
↪   @newData.initialSolAmount > 0 && @newData.initialSolAmount < 50000000000 && @newData.gameTokens ==
↪   null"
+   "create": "(@newData.lockedLiquidity == null || get(/admins/@user.address) != null) &&
↪   @newData.initialSolAmount > 200000000 && @newData.initialSolAmount < 50000000000 &&
↪   @newData.gameTokens == null"
```

GrandmAIsters: Fixed

KupiaSec: Confirmed the fix

7.2 Low

7.2.1 Some fees are not accounted

Description: The GrandmAIsters platform applies the following fee.

Action	Fee
On Agent Creation	1% + 0.001 SOL
At the End of Games	1% of Transferred SOL + 0.001 SOL
Buying and Selling Agent Tokens on grandmAIsters	1.5% + 0.5% Creator Fee

However, in the current implementation, it only applies 1% of Transferred SOL as fee in the agent creation and games.

Impact: Low. Users pay less fee and the protocol faces loss of funds.

Recommended Mitigation: It is recommended to change the policy to reflect this fee amount.

GrandmAIsters: Acknowledged, won't fix

KupiaSec:

7.2.2 pecision loss

Description: In the hooks of the sellToken, agentId transfers 0.4, 0.5, 1.1% of `getSolOutProduct` to the specific address and 98% to seller, and subtracts `getSolOutProduct` from the `actualSolReserves` and `virtualSolReserves`. There is the precision loss to calculate the transfer amount. Let's consider the following scenario:

- getSolOutProduct = 900
- 98%: 900 * 980 // 1000 = 882
- 0.4%: 900 * 4 // 1000 = 3
- 0.5%: 900 * 5 // 1000 = 4
- 1.1%: 900 * 11 // 1000 = 9
- 882 + 3 + 4 + 9 = 898 `actualSolReserves` and `virtualSolReserves` are decreased as 900 even though 898 sol is transferred. As a result, 2 sol is locked in the agent forver. There is the same vulnerability in the resolvedGames

Impact: High.

Recommended Mitigation: It is recommended to update the `actualSolReserves` and `virtualSolReserves` by considering the precision loss.

GrandmAIsters: Acknowledged, won't fix

KupiaSec:

7.3 Informational

7.3.1 Use the Pump.fun bonding curve parameters

Description: Pump.fun uses `1073000000000000` as `initialTokenReserves` rather than `1073000191000000` for its bonding curve.

Impact: Informational

Recommended Mitigation: It is recommended to change some constants in the policy.

GrandmAIsters: Acknowledged, won't fix

KupiaSec: