Three Sigma

Code Audit

D1 DISTRICT ONE

DistrictOne Social Space with Money Games

Disclaimer

Code Audit

DistrictOne Social Space with Money Games

Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

Table of Contents

Code Audit

DistrictOne Social Space with Money Games

Table of Contents

Summary

Code Audit

DistrictOne Social Space with Money Games

Summary

Three Sigma Labs audited DistrictOne in a 4 days engagement. The audit was conducted from 13-06-2024 to 18-06-2024.

Protocol Description

DistrictOne (D1) merges the excitement of money games with social interaction on Blast L2. It features five engaging activities: Linkup for reward earning and networking, Space Sprint for competitive visibility and earnings, Daily Rally to enhance engagement through gem collection and lotteries, SpaceShare for investing in spaces' success while earning from transactions, and the upcoming Battle Mode for head-to-head competition. D1 offers influencers and projects a dynamic platform for growth without entry barriers, leveraging gamified elements for enhanced community traction and engagement.

Scope

Code Audit

DistrictOne Social Space with Money Games

Scope

All the files inside the launch folder.

Assumptions

OpenZeppelin is secure.

Methodology

Code Audit

DistrictOne Social Space with Money Games

Methodology

To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit we report our findings using as a guideline Immunefi's vulnerability taxonomy, which can be found at immunefi.com/severity-updated/. The final classification takes into account the severity, according to the previous link, and likelihood of the exploit. The following table summarizes the general expected classification according to severity and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Severity / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Code Audit

DistrictOne Social Space with Money Games

Project Dashboard

Application Summary

Name	DistrictOne
Commit	1905aafc4da95b16577e0da99e3bf1a9c6b7eabe
Language	Solidity
Platform	Blast

Engagement Summary

Timeline	13-06-2024 to 19-06-2024
№ of Auditors	2
Review Time	4 days

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	4	2	2
Low	1	0	1

None	5	0	5

Category Breakdown

Suggestion	4
Documentation	0
Bug	5
Optimization	1
Good Code Practices	0

Code Maturity Evaluation

Code Audit

DistrictOne Social Space with Money Games

Code Maturity Evaluation

Code Maturity Evaluation Guidelines

Category	Evaluation
Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system.
Arithmetic	The proper use of mathematical operations and semantics.
Centralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Code Stability	The extent to which the code was altered during the audit.
Upgradeability	The presence of parameterizations of the system that allow modifications after deployment.
Function Composition	The functions are generally small and have clear purposes.
Front-Running	The system's resistance to front-running attacks.
Monitoring	All operations that change the state of the system emit events, making it simple to monitor the state of the system. These events need to be correctly emitted.
Specification	The presence of comprehensive and readable codebase documentation.
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage.

Code Maturity Evaluation Results

Category	Evaluation
Access Controls	Satisfactory. All functions had proper access control.
Arithmetic	Satisfactory. No rounding errors were found.
Centralization	Moderate. The owner has some privileges, such as changing fees or changing the exchange to swap OLE.
Code Stability	Satisfactory. The code was stable throughout the audit.
Upgradeability	Weak. The contracts are not upgradeable.
Function Composition	Satisfactory. Functionality was well split into helpers.
Front-Running	Satisfactory. The code has slippage protection.
Monitoring	Satisfactory. All events were correctly emitted.
Specification	Satisfactory. The code reflected the specification.
Testing and Verification	Satisfactory. The codebase had nearly 100% coverage.

Findings

Code Audit

DistrictOne Social Space with Money Games

Findings

3S-D1-M01

Claiming will fail for **`Ole`** and **`D1MemeToken`** if the overfunded **`ETH`** reverts

Id	3S-D1-M01
Classification	Medium
Severity	High
Likelihood	Low
Category	Bug
Status	Acknowledged

Description

`FairLauncher::claims()` reverts if the **`ETH`** claimed from overfunded presales fails. However, this means that **`Ole`** and **`D1MemeToken`** claiming will also revert, when this could be prevented.

Recommendation

Instead of reverting, the **`ETH`** could be sent to a protocol wallet and emit an event to refund users later.

The likelihood of happening is low because the protocol expects its users to be EOAs or multisig wallets, which don't have this problem.

3S-D1-M02

`FairLauncher::participate()` should allow specifying a minimum number of shares due to the overfund mechanism

Id	3S-D1-M02
Classification	Medium
Severity	Medium
Likelihood	Medium
Category	Bug
Status	Acknowledged

Description

Users receive discounted shares when they participate in **`FairLauncher::participate()`** but the amount raised has exceeded the hard cap set. Thus, users may be frontrunned when participating, receiving less shares than expected. Maliciously frontrunning on Blast is not possible because the sequencer is centralized and **`FIFO`**, but users may still be organically frontrunned.

Recommendation

Attackers can not exploit this but users may still be harmed. A minimum shares amount parameter would be advisable.

3S-D1-M03

`FairLauncher` inherits `BlastNoYieldAdapter` but will hold `ETH`

Id	3S-D1-M03
Classification	Medium
Severity	Medium
Likelihood	High
Category	Suggestion
Status	Addressed in #9211325.

Description

Blast earns yield for addresses that hold **`ETH`**. The **`FairLauncher`** contract will hold ETH due to the sales, but inherits **`BlastNoYieldAdapter`**, which does not configure the Blast address yield to claimable.

Recommendation

Inherit **`BlastAdapter`** instead of **`BlastNoYieldAdapter`** to claimed the yield from **`ETH`**.

3S-D1-M04

Any signature is valid before the issuer address is set

Id	3S-D1-M04
Classification	Medium
Severity	High
Likelihood	Low
Category	Bug
Status	Addressed in #9211325.

Description

The **`issuerAddress`** in the **`FairLauncher`** and **`TokenVault`** is not set in the constructor and the signature libraries do not check if **`ecrecover`** returns **`address(0)`**. Thus, before the **`issuerAddress`** is set, it is possible to forge signatures. This could be problematic for example in the **`TokenVault`** as there is no on chain balance tracking, so users may withdraw all the balance. However, this is not very likely because honest users will not deposit in the **`TokenVault`** if the **`issuerAddress`** has not been set.

Recommendation

Either set the **`issuerAddress`** in the constructor or revert if **`ecrecover()`** returns **`address(0)`**.

3S-D1-L01

`FairLauncher::newFairLaunch()` may be misconfigured

Id	3S-D1-L01
Classification	Low
Severity	Medium
Likelihood	Low
Category	Bug
Status	Acknowledged

Description

The `gasOperator` or `feesCfg` may not be set when users create a new fair launch, as they are not set in the constructor and `FairLauncher::newFairLaunch()` is permissionless, so users may snipe the creation of a new fair launch without `fees` or a gas operator. Additionally, the gas operator may not be changed after the `D1MemeToken` is created, so if it is `address(0)`, it will never be set.

Users have no incentive to set the correct `_minBoughtOle` because the only beneficiary is the fee recipient.

The recipient of the `lp`s may burn all the liquidity and steal all funds from the `Uniswap` pool. However, these recipients are whitelisted offchain and the ones that are not will not be displayed on the frontend.

The end time of the presale is not validated, so users can wait too long (possibly forever) for the presale to end to recover their ETH or proceed with the launch.

`UniswapV2` only accepts minting up to `type(uint112).max` tokens, but this number is not capped in the amount to lp, which would break the launch.

Recommendation

The protocol is going to place these checks in the offchain components to avoid these misconfigurations and not display such misconfigured launches.

3S-D1-N01

`Erc20Utils::safeTransferFrom()` assumes the tokens are transferred to **`this`**, which may not be the case

Id	3S-D1-N01
Classification	None
Category	Bug
Status	Acknowledged

Description

`Erc20Utils::safeTransferFrom()` checks the balance of **`address(this)`** before and after transferring, but it may transfer the tokens to a different address, in which case it would always return 0.

The function is not used in the codebase but may be in the future.

Recommendation

The balances should be checked of the **`to`** address, not **`address(this)`**.

3S-D1-N02

Deadline in **`Ole`** swap could be sent as a parameter to further prevent MEV

Id	3S-D1-N02
Classification	None
Category	Suggestion
Status	Acknowledged

Description

The deadline parameter to swap **`Ole`** is **`block.timestamp`**, so there is some room for MEV, although this is technically not possible on Blast as there is not a public mempool.

Recommendation

Send the deadline as a parameter.

3S-D1-N03

Invite fees in **`FairLauncher::_calInviteFees()`** may be 0, in which case storage update could be skipped

Id	3S-D1-N03
Classification	None
Category	Optimization
Status	Acknowledged

Description

In **`FairLauncher::_calInviteFees()`**, state updates are skipped if the **`directInviter`** is **`address(0)`**, but they could also be skipped if the amount is null, which happens after the hard cap is reached.

Recommendation

Skip storage slot updates if the amount is null.

3S-D1-N04

`D1MemeToken`s will be stuck whenever the merkle roots or the free claims are not fully claimed

Id	3S-D1-N04
Classification	None
Category	Suggestion
Status	Acknowledged

Description

In **`FairLauncher::newFairLaunch()`**, the total supply of minted **`D1MemeToken`** is equal to the **`presale + lp + freeClaim + airdrop`**. It's possible some tokens will never be claimed via **`airdrop`** or **`freeClaim`**, but there is no way to recover them. There is no significant impact as these tokens will just be burned and act as if the real supply is decreased.

Recommendation

The tokens could be reimbursed in some way that is fair but this would increase complexity.

3S-D1-N05

`TokenVault` allows withdrawals using offchain logic, so the offchain service must take into consideration reorgs

Id	3S-D1-N05
Classification	None
Category	Suggestion
Status	Acknowledged

Description

`TokenVault` does not track or set the withdrawal logic on chain, so care must be taken in the offchain components due to reorgs. If users deposit and are awarded a withdrawal in a short time frame, a reorg could happen which ends up removing the deposit, but the user still has the withdrawal signature.

Recommendation

The fix is in the offchain component, which is making sure there is a delay between a deposit and the ability of a user to withdraw. The protocol is going to be deployed on Blast, which is unlikely to have reorgs, but it is still a good practice.