Tigris Trade - xTig Audit Report

Prepared by bytes032

Contents

1 About bytes032

bytes032 is an independent smart contract security researcher.

His knack for identifying smart contract security vulnerabilities in a range of protocols is more than a skill; it's a passion. Committed to enhancing the blockchain ecosystem, he invests his time and energy into thorough security research and reviews. Want to know more or collaborate? bytes's always up for a chat about all things security.

Feel free to reach out on X.

2 Protocol Summary

Tigris Trade is a leveraged trading platform that utilizes price data signed by oracles off-chain to provide atomic trades and real-time pair prices.

Open positions are minted as NFTs, making them transferable. Tigris is governed by Governance NFT holders.

The oracle aggregates real-time spot market prices from CEXs and sign them. Traders include the price data and signature in the trade txs.

For people that want to provide liquidity, they can lock up tigAsset tokens (such as tigUSD, received by depositing the appropriate token into the stablevault) for up to 365 days.

They will receive trading fees through an allocation of Governance NFTs, which get distributed based on amount locked and lock period.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of {X} according to the specific commit. Any modifications to the code will require a new security review.

3 Risk Classification

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

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Overview

Project	Tigris Trade - xTig
Repository	Contracts
Commit	eb265362ad33...
Date	June 2023

Issues Found

Severity	Count
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	2
Informational	0
Gas Optimizations	0
Total Issues	**2**

Summary of Findings

Title	Status
[L-1] Whitelisting reward tokens without an unwhitelisting mechanism	Resolved
[L-2] Incompatibility Issue with Solidity Version 0.8.20	Resolved

5 Findings

5.1 Low Risk

[L-1] Whitelisting reward tokens without an unwhitelisting mechanism

Context: xTig.sol

Impact

There's a lack of flexibility and control in managing the whitelisted tokens. This can not directly lead to financial loss and can only complicate token management in specific circumstances and create situations where unwanted tokens cannot be removed from the rewards pool.

Description

The whitelistReward function is used to whitelist reward tokens.

```
function whitelistReward(address _rewardToken) external onlyOwner {
    require(!rewardTokens.get(_rewardToken), "Already whitelisted");
    rewardTokens.set(_rewardToken);
    emit TokenWhitelisted(_rewardToken);
}
```

However, the contract currently lacks a mechanism to unwhitelist tokens. This means that once a token is whitelisted, it remains so indefinitely, unless the entire contract is redeployed.

In a scenario where the team wants to cease the use of a particular reward token, the contract offers no simple means to achieve this. Redeployment of the contract, especially when there are accrued rewards, would be potentially disruptive.

Recommendation

Implement an `unwhitelistReward` function in the xTIG contract that allows the removal of tokens from the whitelist. This function should only be callable by the contract owner, similar to the `whitelistReward` function. This would improve the contract's flexibility and provide better control over the token management.

The `unwhitelistReward` function could look something like this:

```
function unwhitelistReward(address _rewardToken) external onlyOwner {
    require(rewardTokens.get(_rewardToken), "Token not whitelisted");
    rewardTokens.remove(_rewardToken);
    emit TokenUnwhitelisted(_rewardToken);
}
```

[L-2] Incompatibility Issue with Solidity Version 0.8.20

Context: xTig.sol

Impact

The choice of Solidity version 0.8.20 makes the potentially deployed contract non-functional due to opcode incompatibility.

Description

In the xTIG contract, Solidity version 0.8.20 is used as specified in the pragma directive:

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.20;
```

However, due to an opcode incompatibility issue (`push0` opcode is not supported yet), the contract compiles but is non-functional when deployed.

This is because Solidity version 0.8.20 or higher can only be used with an EVM (Ethereum Virtual Machine) version lower than the default `shanghai`. The incompatible `push0` opcode will soon be supported, but it currently makes contracts compiled with Solidity 0.8.20 non-functional.

The compatibility issues mean that the entire contract fails to function as expected, making it impossible to perform any operations on it. The EVM version setting can be adjusted in solc as per the instructions here or in Hardhat as mentioned here.

Versions up to and including 0.8.19 are fully compatible, meaning contracts written in these versions will not face this opcode issue.

Recommendation

Downgrade the Solidity version to 0.8.19 or below to avoid the opcode incompatibility issue until the `push0` opcode support is officially implemented.

Alternatively, you could change the EVM version to a lower one than the default 'shanghai' when compiling the contract