Titan X Security Review

Version 1.0

November 6, 2023

Conducted by:
Georgi Georgiev (Gogo), Independent Security Researcher

Table of Contents

1 About Gogo

Georgi Georgiev, known as Gogo, is an independent security researcher experienced in Solidity smart contract auditing and bug hunting. Having conducted over 40 solo and team smart contract security reviews, he consistently aims to provide top-quality security auditing services. He serves as a smart contract auditor at Paladin Blockchain Security, where he has been involved in security audits for notable clients such as LayerZero, TraderJoe, SmarDex, and other leading protocols.

For security consulting, you can contact him on Twitter, Telegram, or Discord - *@gogotheauditor*.

2 Disclaimer

Audits are a time, resource and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to find as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

3 Risk classification

Severity	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 significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

3.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

3.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Titan X
Repository	https://github.com/jakesharpe777/ttx_buyandburn_v2_private
Commit hash	de6377a1bcff5d943d9eef73ccf92cf1b63553ab
Resolution	9b18b253375f46fb4e516d619ce348b4f08c809d
Methods	Manual review & automated testing

Scope

contracts/BuyAndBurnV2.sol

Issues Found

Governance risk	1
Critical risk	0
High risk	0
Medium risk	0
Low risk	1

5 Findings

5.1 Governance risk

5.1.1 The contract owner can brick all funds forever.

Severity: *Governance risk*

Context: BuyAndBurnV2.sol#L91-L108

Description: The following two functionalities were introduced in V2:

```
function setSlippage(uint256 amount) external {
    require(msg.sender == s_ownerAddress, "InvalidCaller");
    require(amount <= 50, "0-50_Only");
    s_slippage = amount;
}

function setBuynBurnInterval(uint256 amount) external {
    require(msg.sender == s_ownerAddress, "InvalidCaller");
    s_interval = amount;
}
```

The contract owner has the ability to configure the time interval gap between the calls to the `buynBurn` function. These two mechanisms were introduced to prevent any potential MEV bot or simply sandwich attacks. However, due to insufficient input validation, the contract owner (eventually compromised) has the following privileges:
1. set the `s_slippage` to 0%, which will block the contract as the swap won't be able to get executed
2. set the `s_slippage` to 50%, which may allow them to extract value by sandwiching transactions
3. set the `s_interval` to type(uint265).max in order to block the `buynBurn` function

Moreover, the contract owner is allowed to change the `s_capPerSwap` swap parameter to a very low value which could also make the `buynBurn` function unusable. However, a potential fix would limit the flexibility of this function.

Recommendation: Consider adding the input validation checks to prevent the above scenarios.

Resolution: Resolved. Checks for the first 3 cases were added. However, the last risk still remains.

5.2 Low risk

5.2.1 Tracking the total amount of WETH bought and burned can be incorrect.

Severity: *Medium risk*

Context: BuyAndBurnV2.sol#L134

Description: The contract tracks the total amount of WETH that have been used to buy and burn TITANX tokens from the corresponding UniswapV3 pool. The storage variable`s_totalWethBuyAndBurn` is updated in the `buynBurn` function. However, the actual amount used to perform the swap later may differ from the one initially declared.

Recommendation: Consider updating the `s_totalWethBuyAndBurn` variable in `uniswapV3SwapCallback` or `_swapWETHForTitan` instead of `buynBurn`.

Resolution: Resolved. The `s_totalWethBuyAndBurn` is now updated in `uniswapV3SwapCallback` using the amount delta returned from the UniswapV3 pool.