Shogun Security Audit

Report Version 1.0

November 27, 2024

Conducted by **Hunter Security**

Table of Contents

1 About Hunter Security

Hunter Security is an industry-leading smart contract security auditing firm. Having conducted over 100 security audits protecting over $1B of TVL, our team always strives to deliver top-notch security services to the best DeFi protocols. For security audit inquiries, you can reach out on Telegram or Twitter at *@georgehntr*.

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 identify as many vulnerabilities as possible. Audits can reveal the presence of vulnerabilities **but cannot guarantee their absence**.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	High	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** - involves a small loss of funds or affects a core functionality of the protocol.
- **Low** - encompasses any unexpected behavior that is non-critical.

3.2 Likelihood

- **High** - a direct attack vector; the cost is relatively low compared to the potential loss of funds.
- **Medium** - only a conditionally incentivized attack vector, with a moderate likelihood.
- **Low** - involves too many or unlikely assumptions; offers little to no incentive.

3.3 Actions required by severity level

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

4 Executive summary

Hunter Security was engaged by Shogun to review their smart contract protocol.

Overview

Project Name	Shogun
Repository	https://github.com/jakesharpe777/ttx_shogun
Commit hash	01ef756d374387ea3bd25ce2fdc0f8cbe62046cd
Resolution	8598b41d6f130d87dbff73f7e1f53de3f93d7856
Repository	https://github.com/jakesharpe777/ttx_shogun_buyandburn
Commit hash	9aca6c8e888c3c923d3c7dbac6b7181ab7d15e05
Resolution	f0df27714a11ae2240116b1f3a000e044a559b7a
Methods	Manual review & testing

Scope

ttx_shogun/contracts/BurnInfo.sol
ttx_shogun/contracts/GlobalInfo.sol
ttx_shogun/contracts/Shogun.sol
ttx_shogun_buyandburn/contracts/BuyAndBurnShogun.sol

Issues Found

High risk	0
Medium risk	0
Low risk	2

5 Findings

5.1 Low

5.1.1 Price may be significantly different from the expected initial ratio

Severity: Low

Context: BuyAndBurnShogun.sol

Description: In *createInitialLiquidity*, if the pool has already been created and other liquidity providers have added tokens to it, the price may be significantly different from the planned 2_800_000_000:1_000_000 ratio and never actually get that initial liquidity position minted due to the 90% slippage check.

Furthermore, *createInitialPool* would not even get to setting the pool address in storage in case the pool has already been deployed by another address, resulting in a permanent Denial-of-Service for the buy and burn.

Recommendation: Consider making sure the pair does not get created before the initial liquidity is minted and that no other liquidity is provided to it, so that the BuyAndBurnShogun can set the initial price.

Resolution: Resolved.

5.1.2 Not considering principal penalty when burning stake

Severity: Low

Context: Shogun.sol

Description: The *enterAuctionTitanXStake* method in Shogun allows the caller to enter the auction by burning TitanX stake positions they have and matching the corresponding amount with liquid TitanX.

The problem is that when calculating the amount of tokens that has been burned from the stake, *info.titanAmount* is used in the following way:

```
amount += info.titanAmount;
```

What's missing, is the penalty for early stake burning calculated in the original TitanX code in *_calculatePrinciple*. Therefore, the actual amount burned and the one that is calculated for entering the auction do not align unfairly benefiting stakers who use that option.

Recommendation: Consider correctly calculating the actual amount of TitanX tokens burned by subtracting the applied penalty.

Resolution: Acknowledged. Intended behavior.

5.2 Informational

5.2.1 Typographical, non-critical or centralisation issues

Severity: Informational

Context: contracts/*

Description: The contracts contain one or more typographical mistakes, non-critical issues or centralisation vulnerabilities. In an effort to keep the report size reasonable, we enumerate these below:

1. The BuyAndBurnShogun cannot be used as a secondary BuyAndBurn contract for Shogun due to the calculations based on the timestamps set in *setShogunContractAddress*. If a new BuyAndBurn contract is considered to be set for Shogun, it should be re-audited and should start at a different time point than the genesis timestamp of Shogun.
2. Use Ownable2Step instead of Ownable.
3. The *dailyUpdate* function in BuyAndBurnShogun should be executed before any other external function just as in Shogun.
4. Use newer Solidity pragma version in the *shogun_ttx* repository.
5. The old *lpAddress* is not excluded from tax in *setNewLPAddress*.
6. Remove the second check in *enterAuctionTitanXStake* to prevent from failing when less than all stakes have been needed to be approved.
7. Consider applying the *onlyOwner* modifier to *createInitialPool* and *createInitialLiquidity* to control at what point these functions are called.
8. Merge *createInitialLiquidity* and *createInitialPool* with *setShogunContractAddress* to prevent unnecessary additional steps to the setup procedure.
9. Remove the option for the owner to renounce ownership and lose control of the parameters values.
10. Enforce a reasonable upper and lower limit on the following state variables in their setters to prevent Denial-of-Service: *s_capPerSwapTitanX*, *s_capPerSwapX28*, *s_maxIntervalAccumulation*
11. Use *OracleLibrary.getQuoteAtTick(meanTick, 1e18, X28, TITANX)* in *getTwapX28TitanX* to avoid potential overflow.
12. Remove the last line in *getMissedIntervals* in order not to mislead off-chain scripts or external contract callers.
13. Consider which functions in the BuyAndBurnShogun should be available while the initial liquidity has not been minted yet and add the necessary *s_initialLiquidityCreated* checks.

Recommendation: Consider fixing the above typographical mistakes, non-critical issues or centralisation vulnerabilities.

Resolution: Partially resolved.