PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero (sveSTG)

07 March 2023

paladinsec.co   info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for LayerZero's sveSTG contract on the Ethereum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	LayerZero
URL	https://layerzero.network/
Platform	Ethereum
Language	Solidity
Preliminary Contracts	https://github.com/LayerZero-Labs/stargate-dao/pull/17/commits/dd223e4f7e702c9a4b860471c202f82dc3d6510b
Resolution 1	https://github.com/LayerZero-Labs/stargate-dao/blob/eebd614e07f87072a9d6e0e429afabb72bd45eec/contracts/sveSTG.sol
Final Contracts	https://github.com/LayerZero-Labs/stargate-dao/blob/c47af2a65db90a5141a57ace6df5ad0faaaccd7a/contracts/sveSTG.sol

1.2 Contracts Assessed

Name	Contract	Live Code Match
sveSTG	0xd56E00A493eD90C490a54cf58cc5a713556bfdBB	✓ MATCH
ERC20	Dependency	✓ MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	0	-	-	-
● Medium	2	2	-	-
● Low	1	-	-	1
● Informational	2	2	-	-
Total	**5**	**4**	**-**	**1**

Classification of Issues

Severity	Description
● High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
● Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
● Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
● Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 sveSTG

ID	Severity	Summary	Status
01	MEDIUM	`totalSupply()` represents the underlying balance sum instead of the sum of `balanceOf()`	✓ RESOLVED
02	MEDIUM	Contract grants excessive balance before Fri Mar 17 2023	✓ RESOLVED
03	LOW	`sveSTG` defines a month as 30 days while `veSTG` defines this differently	ACKNOWLEDGED
04	INFO	Typographical errors	✓ RESOLVED

1.3.2 ERC20

ID	Severity	Summary	Status
05	INFO	Forking ERC20 is unnecessary	✓ RESOLVED

2 Findings

2.1 sveSTG

`sveSTG` allows for the Stargate governance to grant voting rights to wallets which have STG tokens which are undergoing vesting. Governance can grant `sveSTG` voting rights to wallets via the `mint` function. These voting rights are for all purposes equal to `veSTG` voting rights within snapshot.org votes. Transfers of the `sveSTG` tokens are disabled, similar to `veSTG`.

`sveSTG` is meant to give users voting rights as if they were locking their locked balance until 2 years after the vesting start date. Unlocked tokens will not be eligible for `sveSTG` voting power. This can be nicely symbolized in an equation which holds for any timestamp for the vesting addresses (eg. for Alice vesting X coins):

```
sveSTG.balanceOf(alice | A) + veSTG.balanceOf(alice | A) =
veSTG.balanceOf(alice | B)
```

A and B being mathematical conditions (not bitwise operations!):

- A: Alice vests X tokens, whenever they unlock, she immediately stakes them into veSTG with expiry March 17th, 2 years future
- B: Alice instead just straight up stakes X STG tokens at the start with expiry on March 17th, 2 years future

<u>It should be noted that Paladin did not have access to the vesting contract of these users at the time of this audit.</u> We were therefore unable to validate the exploit vector where there might be a single second where a vesting lot unlocks and the voting power for that month is effectively doubled. We strongly recommend that the team validate this.

2.1.1 Privileged Functions

- `mint`
- `transferOwnership`
- `renounceOwnership`

2.1.2 Issues & Recommendations

Issue #01	`totalSupply()` represents the underlying balance sum instead of the sum of `balanceOf()`
Severity	● MEDIUM SEVERITY
Description	It is common expectation that EIP20 implementations adhere to the following property: `sum(balanceOf(user) for all users) = totalSupply()` This property has been violated within the `sveSTG` token, causing the `totalSupply()` value to be a gross misrepresentation of the current sum of balances. This is because `totalSupply` represents the underlying balances compared to the actual present `sveSTG` balances. It could be noted that approvals are also misrepresented but since transfers are disabled, this really does not matter.
Recommendation	Consider overriding `totalSupply`. Consider moving the `balanceOf` logic to a pure function such as `underlyingToSVE(uint256 underlyingAmount)` or similar. This function can then be reused to map the user balances and the total supply.
Resolution	✓ RESOLVED The client has implemented the recommended change.

Issue #02	Contract grants excessive balance before Fri Mar 17 2023
Severity	● MEDIUM SEVERITY
Description	The contract appears to grant an excessive balance to users if the `balanceOf` function is called before Friday 17 March.
Recommendation	Consider capping the decay factors to at most a 100% cap.
Resolution	✓ RESOLVED A cap has been introduced to `remainingSeconds`.

Issue #03	sveSTG defines a month as 30 days while veSTG defines this differently
Severity	● LOW SEVERITY
Description	The sveSTG contract accounts for months of 30 days. The voting duration is defined as 30 days * 36, symbolizing 3 years. However, within veSTG, this duration is defined as 3 * 365 days, which results in a different number of seconds. This leads to a slight bias in the vesting curve compared to veSTG.

Recommendation	Consider whether this is a concern, if so, it might be necessary to account for the sveSTG decay using `3 * 365` days seconds from the lock time as the expiry time.
	
	`uint public constant MAX_LOCK_TIME = 3 * 365 * 86400;`
	
	`uint public constant VEST_END_TIME = 1679011200 + MAX_LOCK_TIME;`
	
	It should be noted that `remainingMonths` should likely still use the old timeframe.

Resolution	● ACKNOWLEDGED The client has indicated they wish to retain 30 days behavior for now.

Issue #04	Typographical errors
Severity	● INFORMATIONAL

Description	We have consolidated the typographical issues into a single issue to keep the report brief and readable.
	
	Line 20
	`function _beforeTokenTransfer(address from, address, uint) internal virtual override {`
	
	Line 24
	`function balanceOf(address account) public view virtual override returns (uint256) {`
	
	The `virtual` modifier is unnecessary in both lines.
	
	The contract uses both `uint256` and `uint` interchangeably. It would be cleaner from a readability/code quality perspective to stick to a single one (we personally prefer to stick to `uint256`).

Recommendation	Consider fixing the typographical errors.
Resolution	✓ RESOLVED Most of these errors have been fixed.

2.2 ERC20

`ERC20` is a minor fork from the canonical OpenZeppelin v4.5.0 ERC20 implementation. The sole change which has been made was marking the balance as `internal` to allow it to be used within `sveSTG`.

2.2.1 Issues & Recommendations

Issue #05	Forking ERC20 is unnecessary
Severity	INFORMATIONAL
Description	Forking the `ERC20` contract is unnecessary as the underlying balance can be fetched using `super.balanceOf(addr)`.
Recommendation	Consider removing this file and use `super.balanceof(addr)` instead.
Resolution	✓ RESOLVED The client has implemented the recommended change.

PALADIN

BLOCKCHAIN SECURITY