



**PALADIN**  
BLOCKCHAIN SECURITY

# Smart Contract Security Assessment

Final Report

For LayerZero (veSTG)

07 March 2023



[paladinsec.co](https://paladinsec.co)



[info@paladinsec.co](mailto:info@paladinsec.co)

# Table of Contents

Table of Contents	2
Disclaimer	3
1 Overview	4
1.1 Summary	4
1.2 Contracts Assessed	4
1.3 Findings Summary	5
1.3.1 VotingEscrow	6
2 Findings	7
2.1 VotingEscrow	7
2.1.1 Privileged Functions	8
2.1.2 Issues & Recommendations	9



# 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 advocacy 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 veSTG 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

<b>Project Name</b>	LayerZero
<b>URL</b>	<a href="https://layerzero.network/">https://layerzero.network/</a>
<b>Platform</b>	Ethereum
<b>Language</b>	Solidity
<b>Preliminary Contracts</b>	<a href="https://github.com/LayerZero-Labs/stargate-dao/pull/16/commits/6692e0e96777f55ca5164d0a288e65dda747778a">https://github.com/LayerZero-Labs/stargate-dao/pull/16/commits/6692e0e96777f55ca5164d0a288e65dda747778a</a>
<b>Final Contracts</b>	<a href="https://github.com/LayerZero-Labs/stargate-dao/blob/2830a8614d4cf516ee4105e2ff634f2053d2cdb5/contracts/VotingEscrow.sol">https://github.com/LayerZero-Labs/stargate-dao/blob/2830a8614d4cf516ee4105e2ff634f2053d2cdb5/contracts/VotingEscrow.sol</a>

## 1.2 Contracts Assessed

Name	Contract	Live Code Match
VotingEscrow	0x95Df20d5F5D7C92BA929e5C41e0c20eA4B59476A	 MATCH

## 1.3 Findings Summary

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

### 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 VotingEscrow

ID	Severity	Summary	Status
01	HIGH	create_lock_for is fundamentally flawed as _create_lock has been incorrectly refactored to still fetch the locked balance of msg.sender	✓ RESOLVED
	HIGH	Griefing: An exploiter can grief new wallets by seeding them with a small 3 year lock, preventing these wallets from locking their tokens for any time shorter than 3 years	✓ RESOLVED

## 2 Findings

---

### 2.1 VotingEscrow

VotingEscrow is used as the main staking contract for the governance token to generate voting power. It is inspired by and extremely similar to veCRV (Curve's voting-escrow implementation) but in Solidity (Curve's version is written in Vyper).

When people stake their governance tokens, they receive voting power over time. Stakes can be locked up for between one month and 3 years. If a stake is locked up for four years, the full amount of voting tokens is granted. Accordingly, this amount is reduced according to the shortened staking duration.

Once staked, the voting power slowly decays over the token lock period, which means that the voting power balance of all users is constantly decreasing. To regain their voting power, users can extend the lock period of their tokens.

From a high level perspective, voting power is directly proportional to the remaining duration of stakes being locked up, which means that those with the longest locked stake in the protocol will have the largest voting power, which could be considered an ideal incentive alignment.

To establish this mechanism technically, Curve has introduced an elaborate design of recording all points where users create deposits. The end of the deposit is always aligned to 1 week periods and contains the decay factor of that user. When a period expires, all 1 week periods are iterated to ensure that the total supply and different balances are properly adjusted. These 1 week periods are internally called epochs.

If necessary, governance can mark the contract as unlocked to allow everyone to withdraw their underlying Stargate tokens prematurely.

Finally, locks can exclusively be created for EOAs and governance-approved contracts.

This audit is an incremental audit. This audit solely covers the changes introduced from the original stargate voting escrow token into the new voting escrow token. Any issues within the original token, which might still be present in the current token, are therefore **not covered** by this audit.

Even though Paladin did not audit the original VotingEscrow contract, it was audited by other auditors and we recommend reading those audits to get a sense of the risks in the original contract, as it does have some limitations.

In-scope changes: <https://github.com/LayerZero-Labs/stargate-dao/pull/16/commits/6692e0e96777f55ca5164d0a288e65dda747778a>



The main feature which has been introduced is the ability to create locks for others. This is necessary as Layer Zero is migrating to new tokens and they would like to re-create all locks for their users.

## 2.1.1 Privileged Functions

- `add_to_whitelist`
- `remove_from_whitelist`
- `unlock`
- `create_lock_for`
- `transferOwnership`
- `renounceOwnership`
- `changeController [ controller ]`



## 2.1.2 Issues & Recommendations

Issue #01	<b>create_lock_for</b> is fundamentally flawed as <b>_create_lock</b> has been incorrectly refactored to still fetch the locked balance of <b>msg.sender</b>
Severity	 HIGH SEVERITY
Location	<u>Line 357</u> <code>LockedBalance memory _locked = locked[msg.sender];</code>
Description	<p>The Stargate team refactored the VotingEscrow token to allow them to create locks for other wallets. They added this functionality because they need to migrate all users' locks to the new contract. However, while refactoring the codebase to include the option to change the recipient address, a line of code still references <b>msg.sender</b> as the lock owner, eg. the calling wallet.</p> <p>This fundamentally messes up the contract's business logic and the locked balances of wallets could be wiped out because the locked amount of <b>msg.sender</b> is copied into the locked amount. This is a critical bug of high impact.</p>
Recommendation	Consider using the <b>_addr</b> balance instead:  <u>Line 357</u> <code>LockedBalance memory _locked = locked[_addr];</code>
Resolution	 RESOLVED

**Issue #02**

**Griefing: An exploiter can grief new wallets by seeding them with a small 3 year lock, preventing these wallets from locking their tokens for any time shorter than 3 years**

**Severity** **HIGH SEVERITY****Description**

The new VotingEscrow code allows anyone to create a lock for another party. This was originally not permitted in Curve's code as it would allow people to grief wallets.

Example:

1. Bob wants to lock his tokens for 3 months
2. Alice notices this in the mempool
3. Alice frontruns and creates a 3 year lock for Bob using 0.0001 tokens
4. Bob's transaction fails and he is forced to move to a new wallet to try again (where Alice could be still waiting for him)

This example can be circumvented with flashbots, but of course Alice can be proactive and DoS all wallets with significant amounts of Stargate tokens.

**Recommendation**

There are several ways to address this:

1. Implement an allowlist for contracts which are allowed to create locks for users
2. Implement an approval flow where users must approve this action
3. Implement a `burn_and_create_lock` function which burns the current lock

Given the business requirements of Stargate, we recommend a temporary privileged role that is allowed to create locks which will eventually be renounced (eg. solution 1). The other solutions are more complex or do not satisfy the client's requirements.

**Resolution** **RESOLVED**

The client has chosen to go for solution 1 and has made the function `onlyOwner`.



**PALADIN**  
BLOCKCHAIN SECURITY