Pashov Audit Group

stHYPE Security Review

November 10th 2025 - November 11th 2025

Contents

1. About Pashov Audit Group

Pashov Audit Group consists of 40+ freelance security researchers, who are well proven in the space - most have earned over $100k in public contest rewards, are multi-time champions or have truly excelled in audits with us. We only work with proven and motivated talent.

With over 300 security audits completed — uncovering and helping patch thousands of vulnerabilities — the group strives to create the absolute very best audit journey possible. While 100% security is never possible to guarantee, we do guarantee you our team's best efforts for your project.

Check out our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

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

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users
- **Medium** - leads to a moderate material loss of assets in the protocol or moderately harms a group of users
- **Low** - leads to a minor material loss of assets in the protocol or harms a small group of users

Likelihood

- **High** - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost
- **Medium** - only a conditionally incentivized attack vector, but still relatively likely
- **Low** - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive

4. About stHYPE

stHYPE is a Liquid Staking Protocol built for Hyperliquid HyperEVM that allows users to stake native HYPE tokens and receive stHYPE, a liquid staking derivative that accrues rewards while remaining tradable. It features modular staking through the Overseer contract, a rebasing stHYPE token, and a wrapped non-rebasing wstHYPE version for DeFi integrations.

5. Executive Summary

A time-boxed security review of the **ValantisLabs/sthype-contracts** repository was done by Pashov Audit Group, during which **Shurikenzer**, **0xAlix2**, **0xunforgiven** engaged to review **stHYPE**. A total of **4** issues were uncovered.

Protocol Summary	
Project Name	stHYPE
Protocol Type	Liquid Staking
Timeline	November 10th 2025 - November 11th 2025

Review commit hash:
- 0661f1972e3d02fcd923d1584022db5fa4ad621f (ValantisLabs/sthype-contracts)

Fixes review commit hash:
- 0478fc5a8a62da4f8e46ca0ef0d4030e4723abbe (ValantisLabs/sthype-contracts)

Scope

`StakingModuleExternalManagement.sol`

6. Findings

Findings count

Severity	Amount
Low	4
Total findings	**4**

Summary of findings

ID	Title	Severity	Status
[L-01]	`StakingModuleExternalManagement` does not expose `depositCap` and `amountDeposited`	Low	**Resolved**
[L-02]	ExternalManagement module uses the same storage slot as HyperCore StakingModule	Low	**Resolved**
[L-03]	Rebase does not have protection for sudden balance drops	Low	Acknowledged
[L-04]	`amountDeposited` is never decremented, blocking new deposits	Low	Acknowledged

Low findings

[L-01] `StakingModuleExternalManagement` does not expose `depositCap` and `amountDeposited`

`StakingModuleExternalManagement` defines internal storage variables `depositCap` and `amountDeposited` but does not provide any public or external getter functions for them.

Recommendations: Consider adding view functions to expose both fields.

[L-02] ExternalManagement module uses the same storage slot as HyperCore StakingModule

`StakingModuleExternalManagement` defines its storage under the same ERC-7201 namespace (`"stHYPE.storage.StakingModule"`) that is also used by `HyperCoreStakingModule` . This violates EIP-7201's requirement that each namespace must avoid collisions with others.

Recommendations:

Consider using a unique namespace for `StakingModuleExternalManagement` , to maintain proper storage isolation.

[L-03] Rebase does not have protection for sudden balance drops

Even though the stake account is trusted, sometimes the `rebase()` call might be delayed and run when the account's balance isn't up to date. This could make the system think the balance dropped a lot (for example, N%) when it really didn't, and mess up the rebase math. in `Overseer.sol` :

```
function rebase() external onlyRole(REBASER) {
    ...
    uint256 newSupply = getNewSupply();
    ...
}
```

`getNewSupply()` aggregates balances from all staking modules:

```
    function getNewSupply() public view returns (uint256) {
        ...
            totalBalance += IStakingModule(stakingModule).getTotalBalance();
        ...
    }
```

And in `StakingModuleExternalManagement.sol` , `getTotalBalance()` depends on the external `stakeAccount` , which may not always reflect the correct current balance:

```
    function getTotalBalance() external view override returns (uint256) {
        address stakeAccount = _getStakingModuleStorage().stakeAccount;
        // WARNING: This will be temporarily incorrect if either:
        // - stake account has not deposited the received HYPE into its staking balance (see
`deposit()`)
        // - the stake account's stake balance has been unstaked into spot balance,
        //   but not yet sent to this contract's HyperCore spot balance (see
`requestWithdraw()`)
        // It is important that `manager` (Overseer) does not call `rebase()` in these
scenarios,
        // so that stHYPE supply does not get adjusted incorrectly.
        return address(this).getHypeSpotBalance() * E10 + address(this).balance
            + stakeAccount.getHypeStakingBalance() * E10;
    }
```

Recommendation: Add a check that stops the rebase if the balance suddenly drops too much, unless the manager manually allows it with a special flag.

[L-04] `amountDeposited` is never decremented, blocking new deposits

In `deposit()`, the module increases `_getStakingModuleStorage().amountDeposited` by the deposited amount, but never decreases it when funds are withdrawn. Because the deposit cap check compares against the cumulative total, once the cap is reached, all future deposits will revert, even if the manager has withdrawn funds.

```
if (_getStakingModuleStorage().amountDeposited + amountEvm >
_getStakingModuleStorage().depositCap) {
    revert StakingModuleExternalManagement__deposit_ExceedsDepositCap();
}

// Update the amount deposited (measured in EVM decimals)
_getStakingModuleStorage().amountDeposited += amountEvm;
```

This behavior effectively locks the module from accepting new deposits after reaching the cap, even though capacity may be available again.

Recommendations

Consider decrementing the `amountDeposited` whenever a withdrawal is fulfilled.