Omnichain Security Audit

Report Version 1.0

March 5, 2025

Conducted by **Hunter Security**

Table of Contents

1 About Hunter Security

Hunter Security is an industry-leading smart contract security company. Having conducted over 100 security audits protecting over $1B of TVL, our team delivers 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 Findings

4.1 Medium

4.1.1 Bridging could be temporarily DoS-ed

Severity: Medium

Description: When bridging E280 tokens from Ethereum to Base, the following code snippet is used:

```
balance = _processIncentiveFee(E280, balance, incentiveFeeBps);

SendParam memory params = SendParam({
    // ...
    amountLD: balance,
    minAmountLD: balance, // @audit dust amount not cleared
    // ...
});
```

The problem here is that the following check in *_debit* would cause the *send* to revert if the balance of the contract includes any dust amount (such as 1 wei):

```
  amountSentLD = _removeDust(_amountLD);
  amountReceivedLD = amountSentLD;

  if (amountReceivedLD < _minAmountLD) {
      revert SlippageExceeded(amountReceivedLD, _minAmountLD);
  }
```

Recommendation: Consider removing the dust amount from the balance in the *bridge* function.

Resolution: Resolved.

4.1.2 Users can avoid paying fee on transfer

Severity: Medium

Description: Consider the following scenario:

1. Alice wants to transfer 1,000,000 S88 tokens from my account to Bob on Ethereum
2. Tax would be 4%, i.e. 40,000 S88 tokens
3. However, Alice can just send these tokens from her Ethereum account to Bob on Base
4. Then Bob can transfer it to himself from Base to Ethereum
5. That way they avoid paying the 4% fee on transfer

Recommendation: Consider either not allowing users to do cross-chain transfers among each other (i.e. allow only to their own address) or applying a tax on cross-chain transfers.

Resolution: Acknowledged. Accepted risk.

4.1.3 Incorrect constant addresses used

Severity: Medium

Description: Several constant addresses currently refer to non-existent contracts. Also, the HLX address is used instead of ELMNT in S88 Ethereum constants.

Recommendation: Consider using the correct constant addresses for deployed contracts.

Resolution: Acknowledged. The correct values would be added upon deployment.

4.2 Low

4.2.1 Cross-chain gas limit may be insufficient

Severity: Low

Description: A default gas limit for the LayerZero receive function (that will be involved on the destination chain during a cross-chain transfer) is set at 75000 gas units. A setter method is implemented which allows the contract owner to adjust this limit.

The problem is that if the passed gas limit is too low, users might not receive their tokens on the destination chain. They would have to manually execute the transaction on the destination chain if this happens.

Recommendation: Consider implementing a minimum gas limit constraint enforced in the setter function as well as thoroughly testing what value should be used.

Resolution: Resolved.

4.2.2 Potential overflow

Severity: Low

Description: The following function is present in all token contracts:

```
function _applyTax(
    uint256 amount,
    uint16 taxBps
) internal pure returns (uint256 taxAmount, uint256 amountAfterTax) {
    unchecked {
        taxAmount = (amount * taxBps) / 100_00;
        amountAfterTax = amount - taxAmount;
    }
}
```

The problem is that if the passed *amount* is greater than *type(uint256).max / taxBps*, silent overflow will happen making the *taxAmount* lower than intended.

Recommendation: Consider removing the *unchecked* box.

Resolution: Resolved.

4.3 Informational

4.3.1 Typographical mistakes and non-critical issues

Severity: Informational

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

1. Several custom errors are defined but never used.
2. Test values present in the code - *BASE_DST_EID*.
3. S88 token on the Base chain currently named E280.
4. Incorrect values used for E280 operations in S88LPDepositor - *ElmntSwap* event emitted in *swapE280* as well as wrong storage variables updated in E280 setters.
5. *_addressToBytes32* unused in S88LPDepositor.
6. Token allowance not reset in *inject* after liquidity is added.
7. When using *setTaxDistributor* and upon construction, the *whitelistTo* should also be updated. Otherwise, if a transfer is made to the tax distributor, it would charge a fee which would be sent to the same address leading to unnecessary second transfer.
8. *distribute* does not need to be *payable*.
9. Consider always using *feeOnTransfer = true*. Otherwise, if misconfigured, unexpected behavior may occur.

Recommendation: Consider fixing the above typographical mistakes and non-critical issues.

Resolution: Resolved. Only pt. 2 remained unchanged.