



SMART CONTRACT SECURITY AUDIT

Swaylend

Oracle Integration

CLIENT

Swaylend

PLATFORM

Fuel

LANGUAGE

Sway

REVIEW

July 2025

DOCUMENT

Document information

Project	Swaylend, Oracle Integration
Client	Swaylend
Platform	Fuel
Language	Sway
Repository	Not recorded in the original report
Commit	Not recorded in the original report
Methods	Manual code review, automated analysis
Report generated	25 July 2025

Document history

VERSION	DATE	CHANGE
1.0	25 July 2025	Original report issued View original report
1.1	October 2026	Reformatted into the Safe Edges standard template. Findings unchanged

This report was reformatted into the Safe Edges standard template in October 2026. Findings, severities, recommendations and code references are unchanged from the original report, which is preserved unchanged in the [Safe Edges Audit Portfolio on GitHub](#).

CONTENTS

Contents

01	Executive summary	3
02	Scope	4
03	Methodology	5
04	Severity and status	6
05	Findings summary	7
06	Detailed findings	8
	H-01 Missing Funds Validation in update_price_feeds	8
	M-01 Missing Staleness Checks in Oracle Queries	9
	M-02 Missing Price Feed Validation	10
	L-01 Decimal Mismatch May Cause Function Revert or Miscalculation	11
07	Recommendations	12
08	Conclusion	12
	Disclaimer	13

Executive summary

About Swaylend

Swaylend is a lending protocol on the Fuel network. This audit focused on the oracle integration and price feed management functionality within the Swaylend lending market contract.

Summary of the review

Safe Edges reviewed the oracle integration and price feed management in Swaylend's lending market contract. Four issues were identified: one High, two Medium and one Low. The High issue allowed anyone to update Pyth price feeds using the market's own balance instead of their own funds. All four issues were fixed by the development team and validated by Safe Edges, with fix commits recorded against each finding.

Findings by severity



Findings by status

STATUS	ISSUES
● Resolved	4

Scope

Repository	Not recorded in the original report
Commit	Not recorded in the original report
Platform	Fuel
Language	Sway

FOCUS AREAS

- Pyth oracle integration
- Price feed update mechanisms
- Oracle data validation
- Fund management for oracle operations
- Decimal scaling logic in collateral and borrowing calculations

Only the code at the commit above was reviewed. Changes made after this commit are not covered by this report.

Methodology

Methods recorded for this engagement: Manual code review, automated analysis.

Manual review

Auditors read the in-scope code line by line, tracing control flow, state changes and trust boundaries against the intended behaviour described by the client.

Threat modelling

We map privileged roles, external dependencies and value flows, then test the assumptions the code makes about each of them.

Functional and automated testing

Where tooling exists for the language and chain, we run static analysis and targeted tests to confirm suspected issues and check edge cases.

Fix review

After the client addresses the findings, we review the updated code and record the final status of each issue.

Severity and status

Severity levels

SEVERITY	DEFINITION
Critical	Direct loss or permanent lock of funds, or full compromise of the protocol, with few or no preconditions. Must be fixed before deployment.
High	Significant loss of funds, broken core functionality or manipulation of critical state under realistic conditions. Should be fixed before deployment.
Medium	Limited loss or degraded functionality, or issues that need specific conditions or privileged access to exploit. Should be fixed.
Low	Minor impact, low likelihood, or deviations from best practice. Fix when practical.
Informational	Code quality, documentation, gas and style observations with no direct security impact.

Status definitions

STATUS	DEFINITION
Resolved	The issue was fixed and the fix was verified.
Partially resolved	The risk was reduced but not fully removed.
Acknowledged	The client accepted the issue without changing the code.
Open	The issue was not addressed at the time of the report.
Not recorded	The original report did not record a resolution status for this issue.

Findings summary

ID	TITLE	SEVERITY	STATUS
H-01	Missing Funds Validation in update_price_feeds	High	● Resolved
M-01	Missing Staleness Checks in Oracle Queries	Medium	● Resolved
M-02	Missing Price Feed Validation	Medium	● Resolved
L-01	Decimal Mismatch May Cause Function Revert or Miscalculation	Low	● Resolved

Detailed findings

H-01 Missing Funds Validation in `update_price_feeds`

Severity **High** Status ● Resolved

DESCRIPTION

The `update_price_feeds` function responsible for updating Pyth oracles failed to validate that the funds sent by users matched the required `update_fee` parameter. This vulnerability allowed attackers to update price feeds using the market's internal balance instead of their own funds, resulting in direct fund loss for depositors.

TECHNICAL DETAILS

```
#[storage(read, write), payable]
fn update_price_feeds(update_data: Vec<Bytes>) {
    require(
        msg_asset_id() == AssetId::base(),
        PythError::FeesCanOnlyBePaidInTheBaseAsset,
    );
    let required_fee = total_fee(total_number_of_updates, storage.single_update_fee);
    require(msg_amount() >= required_fee, PythError::InsufficientFee);
}

#[payable, storage(read)]
fn update_price_feeds_internal(update_fee: u64, update_data: Vec<Bytes>) {
    let contract_id = storage.pyth_contract_id.read();
    require(contract_id != ZERO_B256, Error::OracleContractIdNotSet);
    let oracle = abi(PythCore, contract_id);
    oracle.update_price_feeds {
        asset_id: FUEL_ETH_BASE_ASSET_ID, coins: update_fee
    }(update_data);
}
```

IMPACT

- Direct fund loss from market reserves
- Exploitation by malicious actors to drain protocol funds
- Potential for repeated attacks until market balance depletion

RISK SCORE

BVSS: `A0:A/AC:L/AX:M/C:N/I:N/A:N/D:C/Y:N/R:N/S:C` (8.4)

REMEDIATION

Proper validation was added to ensure the user-sent amount matches or exceeds `update_fee` and the asset used is correct.

Fix commit: `622bd073b48cfc211e8b7d6d0474186c61903e6f`

M-01 Missing Staleness Checks in Oracle Queries

Severity **Medium** Status ● Resolved

DESCRIPTION

The protocol failed to implement staleness checks when querying Pyth price feeds, allowing the use of outdated price data that may not reflect current market conditions.

TECHNICAL DETAILS

```
pub struct Price {
    pub confidence: u64,
    pub exponent: u32,
    pub price: u64,
    pub publish_time: u64,          // TAI64 timestamp
}

#[storage(read)]
fn get_price_internal(price_feed_id: PriceFeedId) -> Price {
    let contract_id = storage.pyth_contract_id.read();
    require(contract_id != ZERO_B256, Error::OracleContractIdNotSet);
    let oracle = abi(PythCore, contract_id);
    let price = oracle.price(price_feed_id);
    price
}
```

IMPACT

- Market manipulation through exploitation of stale prices
- Incorrect valuation of collateral and borrowed assets
- Risk of liquidations based on outdated pricing

RISK SCORE

BVSS: A0:A/AC:M/AX:L/C:N/I:N/A:M/D:M/Y:N/R:N/S:C (5.2)

REMEDIATION

Implemented time validation checks to reject stale prices.

Fix commit: 622bd073b48cfc211e8b7d6d0474186c61903e6f

M-02 Missing Price Feed Validation

Severity **Medium** Status ● Resolved

DESCRIPTION

The oracle price retrieval function lacked validation of returned values such as zero price, high confidence intervals, or invalid exponents.

TECHNICAL DETAILS

```
#[storage(read)]
fn get_price_internal(price_feed_id: PriceFeedId) -> Price {
    let contract_id = storage.pyth_contract_id.read();
    require(contract_id != ZERO_B256, Error::OracleContractIdNotSet);
    let oracle = abi(PythCore, contract_id);
    let price = oracle.price(price_feed_id);
    price
}
```

IMPACT

- Acceptance of invalid or zero price data
- Inaccurate financial calculations and liquidations

RISK SCORE

BVSS: A0:A/AC:L/AX:L/C:N/I:L/A:L/D:L/Y:N/R:N/S:C (4.7)

REMEDIATION

Sanity checks added:

- Non-zero price
- Acceptable confidence ratio
- Reasonable exponent validation

Fix commit: 622bd073b48cfc211e8b7d6d0474186c61903e6f

L-01 Decimal Mismatch May Cause Function Revert or Miscalculation

Severity **Low** Status ● Resolved

DESCRIPTION

Functions such as `collateral_value_to_sell()` and `available_to_borrow()` failed when the base token had more decimals than the collateral token, either reverting or producing incorrect calculations.

TECHNICAL DETAILS

```
// Example causing underflow:
let scale = u256::from(10_u64).pow(
    collateral_configuration.decimals - storage.market_configuration.read().base_token_decimals,
);

// Example causing wrong borrow limit:
let scale = u256::from(10_u64).pow(
    collateral_configuration.decimals + price_exponent -
    market_configuration.base_token_decimals,
);
borrow_limit += amount * price / scale;
```

IMPACT

- External integrations (bots, APIs) revert unexpectedly
- Borrowing limits underestimated
- Potential market inefficiency or minor griefing

RISK SCORE

BVSS: A0:A/AC:H/AX:M/C:N/I:L/A:L/D:L/Y:N/R:N/S:L (2.1)

REMEDIATION

Implemented conditional logic to compare and compute proper scale depending on which decimal is greater, including fallback if division is not needed.

Fix commit: 9132747331188b86dd8cbf9a1ca37b811d08dddb

Recommendations

Immediate actions

1. Oracle Monitoring: Implement real-time monitoring of price anomalies.
2. Circuit Breakers: Add fallback mechanisms to halt operations on extreme deviation.
3. Decimal Handling Checks: Normalize decimal logic across markets for consistency.

Long-term improvements

1. Test Coverage: Expand unit and fuzz testing for edge cases.
2. Documentation: Update oracle and pricing behavior documentation.
3. Scheduled Reviews: Perform periodic re-audits post-upgrades.

Conclusion

The audit identified four total vulnerabilities, including one high, two medium, and one low severity issue. All vulnerabilities have been fixed and validated by the Safe Edges team. Swaylend's oracle and financial logic now include robust validation and fallback measures to protect user funds.

Safe Edges recommends conducting a follow-up security review within six months or immediately after significant code changes to maintain a strong security posture.

Disclaimer

This report is based on the code at the commit listed in the Scope section and on information the client provided during the review. It reflects Safe Edges' assessment at the time of the review and is not a guarantee that the code is free of vulnerabilities.

Changes made after the reviewed commit, deployment configuration, off-chain components and third-party dependencies are outside the scope of this report unless stated otherwise.

Security is ongoing. We recommend continued testing, monitoring and bug bounty coverage alongside any audit.

This report is not investment advice and is not an endorsement of the project, its team or any token.



Safe Edges is a Web3 security firm and official security partner for the Fuel, ENI and XI EcoChain ecosystems. We work across smart contract audits, Rust and WASM reviews, ZK circuit review, AI and MCP security, and penetration testing.

200+

PROJECTS AUDITED

\$4B+

TVL SECURED

0

POST-AUDIT HACKS