



SECURITY AUDIT

KarenPsycho-Bot

Bot and Swap Script

CLIENT

KarenPsycho

PLATFORM

Fuel

LANGUAGE

Sway, TypeScript

REVIEW

December 2024

DOCUMENT

Document information

Project	KarenPsycho-Bot, Bot and Swap Script
Client	KarenPsycho
Platform	Fuel
Language	Sway, TypeScript
Repositories	D4Musketeers/KarenPsycho-SwapScript D4Musketeers/KarenPsycho-Bot
Commit	Not recorded in the original report
Methods	Manual Analysis, Functional Testing, Automated Testing
First review	24 to 31 December 2024
Updated code received	29 December 2024
Second review	31 December 2024

Document history

VERSION	DATE	CHANGE
1.0	December 2024	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 Storing User Private Keys in Plaintext Logs	8
	M-01 Incorrect amount_out Calculation After Fee Deduction in Sell Function	8
	M-02 No Upper Limit on Fee Percentage	9
	L-01 Missing Validation or Error Message for Invalid Token Address	10
07	Automated tests	11
A	Appendix: vulnerability checklist	12
	Disclaimer	13

Executive summary

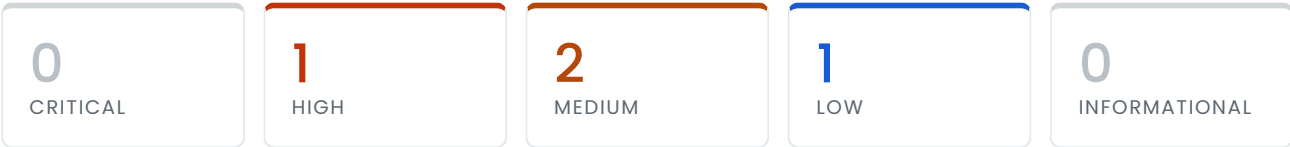
About KarenPsycho-Bot

A time-boxed security review of the KarenPsycho-Bot was performed by Safe Edges, focusing on the security aspects of the smart contract's implementation.

Summary of the review

Safe Edges reviewed the KarenPsycho trading bot and its Sway swap script on Fuel between 24 and 31 December 2024. Four issues were identified: one High, two Medium and one Low. The High issue exposed users' private keys in plaintext logs. The team fixed all four.

Findings by severity



Findings by status

STATUS	ISSUES
Resolved	4

Scope

Repositories	D4Musketeers/KarenPsycho-SwapScript D4Musketeers/KarenPsycho-Bot
Commit	Not recorded in the original report
Platform	Fuel
Language	Sway, TypeScript

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 Analysis, Functional Testing, Automated Testing.

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	Storing User Private Keys in Plaintext Logs	High	● Resolved
M-01	Incorrect amount_out Calculation After Fee Deduction in Sell Function	Medium	● Resolved
M-02	No Upper Limit on Fee Percentage	Medium	● Resolved
L-01	Missing Validation or Error Message for Invalid Token Address	Low	● Resolved

Detailed findings

H-01 Storing User Private Keys in Plaintext Logs

Severity **High** Status ● Resolved

STEPS TO REPRODUCE

1. Navigate to the "Export Private Key" page.
2. Export the private key, ensuring it is sent in an encrypted format.
3. Click the back button to exit the page.
4. Observe the HTTP request sent, which contains the plaintext private key.
5. Check the logs to confirm that the plaintext private key is stored.

IMPACT

Leaking Private key in logs

M-01 Incorrect amount_out Calculation After Fee Deduction in Sell Function

Severity **Medium** Status ● Resolved

DESCRIPTION

The amount_out value is not updated after fees are deducted in the sell function. While the fee is transferred correctly, the user's remaining balance (amount_out) does not reflect the deducted amount

```
if i == pools.len() - 1 {
    if !is_buy {
        let fee_deduction_amount = (amount_out * FEE_PERCENTAGE) / FEE_DENOMINATOR; // x%
        fee of the last output
        transfer(FEE_ADDR, asset_out, fee_deduction_amount);
    }
}
i += 1;
}
```

amounts_out

STEPS TO REPRODUCE

1. Perform a sell operation.
2. Observe the amount transferred to the user after the fee deduction.
3. Confirm that the amount_out does not account for the deducted fee.

IMPACT

- Wrong amount_out logic in sell

RECOMMENDATION

- Recalculate and update amount_out after the fee deduction

```
if i == pools.len() - 1 {
    if !is_buy {
        let fee_deduction_amount = (amount_out * FEE_PERCENTAGE) / FEE_DENOMINATOR;
        amount_out -= fee_deduction_amount;
        transfer(FEE_ADDR, asset_out, fee_deduction_amount);
    }
}
```

M-02 No Upper Limit on Fee Percentage

Severity **Medium** Status ● Resolved

DESCRIPTION

There is a lack of reentrancy guards in functions that perform external calls. The code does not consistently follow the Checks-Effects-Interactions (CEI) pattern.

```
let fee = (amount_in * FEE_PERCENTAGE) / FEE_DENOMINATOR; // x% fee
let mut amount_in_after_fee = amount_in;
if is_buy {
    // If it's a "buy", deduct the x% fee before doing the calculations
    amount_in_after_fee = amount_in - fee;
}
// fee calc end
```

IMPACT

- Users lose all their funds due to excessive fees.

RECOMMENDATION

- Introduce a maximum limit for FEE_PERCENTAGE (e.g., 5%).
- Validate fee percentages during configuration and runtime.

L-01 Missing Validation or Error Message for Invalid Token Address

Severity **Low** Status ● Resolved

DESCRIPTION

The system does not validate Ethereum token addresses when users attempt to buy eth tokens. This leads BOT stuck.

RECOMMENDATION

- Add ERROR HANDLING
- Provide clear and descriptive error messages for invalid inputs.

Automated tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

APPENDIX A

Vulnerability checklist

Vulnerability classes checked during this review, as listed in the original report.

✓ Arbitrary write to storage	✓ Upgradeable safety
✓ Centralization of control	✓ Using inline assembly
✓ Ether theft	✓ Style guide violation
✓ Improper or missing events	✓ Parallel Execution safety
✓ Logical issues and flaws	✓ UTXO Model Verification
✓ Arithmetic Computations Correctness	✓ FuelVM Opcodes
✓ Race conditions/front running	✓ Cross-Chain Interactions
✓ Re-entrancy	✓ Modular Design
✓ Malicious libraries	✓ Access Control Vulnerabilities
✓ Address hardcoded	✓ Denial of Service (DoS)
✓ Divide before multiply	✓ Oracle Manipulation
✓ Integer overflow/underflow	✓ Signature Replay Attacks
✓ ERC's conformance	✓ Improper Handling of External Calls
✓ Missing Zero Address Validation	✓ Proxy Storage Collision
✓ Revert/require functions	✓ Use of Deprecated Functions

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