



SMART CONTRACT SECURITY AUDIT

Zap

Wallet Contracts

CLIENT

Zap

PLATFORM

Fuel

LANGUAGE

Sway

REVIEW

March 2025

DOCUMENT

Document information

Project	Zap, Wallet Contracts
Client	Zap
Platform	Fuel
Language	Sway
Repository	Not recorded in the original report
Commit	Not recorded in the original report
Methods	Not recorded in the original report
Report date	March 2025

Document history

VERSION	DATE	CHANGE
1.0	March 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 Asset validation is not properly implemented in Module 5	8
	H-02 Wallet Initialization Race Condition	8
	L-01 Zero Address Check Missing in Ownership Transfer	9
	L-02 Unused Variable upperb_ovf	10
	I-01 Contract Pause Check Missing on Version Update	11
	Disclaimer	12

Executive summary

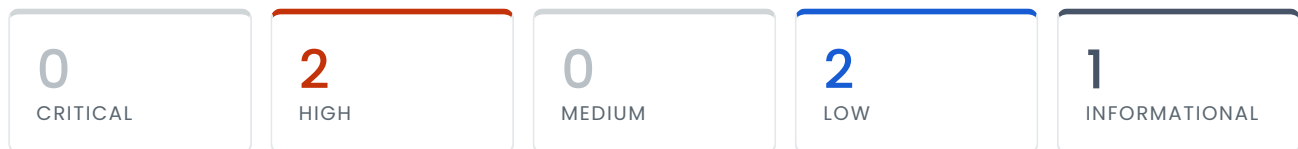
About Zap

Zap's wallet contracts on Fuel let an EVM address own a Fuel wallet, initialised with a nonce asset and with its functionality split across modules.

Summary of the review

Safe Edges reviewed Zap's Sway wallet contracts on Fuel. Five issues were identified: two High, two Low and one Informational. The High issues were an asset check in Module 5 that required both outputs to be the base asset, blocking native asset transfers, and a race in `initialize_wallet` that lets parallel transactions mint duplicate nonce assets for the same wallet. H-01 was fixed with a published patch; the original records the remaining issues as needing fixes.

Findings by severity



Findings by status

STATUS	ISSUES
● Resolved	1
● Not recorded	1
● Open	3

Scope

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

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: Not recorded in the original report.

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	Asset validation is not properly implemented in Module 5	High	● Resolved patch
H-02	Wallet Initialization Race Condition	High	● Not recorded
L-01	Zero Address Check Missing in Ownership Transfer	Low	● Open
L-02	Unused Variable upperb_ovf	Low	● Open
I-01	Contract Pause Check Missing on Version Update	Informational	● Open

Detailed findings

H-01 Asset validation is not properly implemented in Module 5

Severity **High** Status ● Resolved patch

OVERVIEW

In `main.sw`, Module 5, at line 243, a logic check ensures that the first output is strictly `FUEL_BASE_ASSET`:

```
assert(output_coin_asset_id(0).unwrap() == FUEL_BASE_ASSET);
```

However, at line 241, another assertion enforces that the second output must also be `FUEL_BASE_ASSET`:

```
assert(output_coin_asset_id(1).unwrap() == FUEL_BASE_ASSET);
```

This restriction mandates that both outputs must be the Fuel base asset, potentially blocking legitimate transactions that intend to transfer native assets instead.

IMPACT

- **Transaction Rejection:** If a user intends to transfer a native asset other than `FUEL_BASE_ASSET`, all the transaction fails

RECOMMENDATION

- Modify the assertion logic to correctly differentiate between base asset and AssetId transfers.

H-02 Wallet Initialization Race Condition

Severity **High** Status ● Not recorded

OVERVIEW

The wallet is built on **Fuel**, which supports **parallel transaction execution**. However, a critical issue exists in the `initialize_wallet` function: **storage is updated after minting the nonce asset**. This allows two transactions (**Tx1** and **Tx2**) to be processed in parallel, both creating **identical nonce assets** with the same wallet address.

VULNERABLE CODE SNIPPET

```
fn initialize_wallet(
    master_addr: Address,
    owner_evm_addr: EvmAddress,
    initdata: InitData,
) -> EvmAddress {
    require(!_is_paused(), "Contract is paused");

    match initdata {
        InitData::InitModules(base_mods) => {
            require(storage.can_initialize.read(), "Wallet initialization is not
            let key = get_key1(owner_evm_addr, master_addr);
            require(
                !check_initialized(key),
                "Wallet already has Nonce, if error mint assets individually"
            );
            let (nonce_tfr_amt, nonce_assetid) = mint_nonce_asset(owner_evm_addr
            storage.v1_map.insert(key, nonce_assetid);
        }
    }
}
```

EXPLOIT SCENARIO

1. **Parallel Transactions Execution:** An attacker sends **Tx1** and **Tx2** simultaneously, calling `initialize_wallet()`.
2. **Bypassing the Initialization Check:** `check_initialized()` executes before storage updates, returning false for both transactions.
3. **Double Minting Occurs:** Tx1 mints a nonce asset. Tx2 mints another nonce asset before storage updates.

IMPACT

- **Duplicate Wallet Initializations:** Two separate nonce assets exist for the same wallet address.
- **Signature Replay Attacks:** This flaw impacts Module 1 and Module 2, enabling attackers to abuse wallet

RECOMMENDATION

- **Immediate Fix:** Update storage before minting the nonce asset.

L-01 Zero Address Check Missing in Ownership Transfer

Severity Low Status Open

OVERVIEW

The `transfer_ownership()` function does not validate whether the new owner's address is a zero address before transferring ownership.

VULNERABLE CODE SNIPPET

```
#[storage(read, write)]
fn transfer_ownership(new_owner: Identity) {
    require_owner();
    storage.owner.write(State::Initialized(new_owner));
}
```

IMPACT

- If the ownership is transferred to a zero address, recovery is impossible.
- The contract becomes permanently inaccessible, leading to a loss of administrative control.

RECOMMENDATION

Fix required:

- Introduce a validation check preventing a zero address assignment.
- Implement event logging for ownership changes to enhance security monitoring.

L-02 Unused Variable upperb_ovf

Severity Low Status ● Open

OVERVIEW

The variable `upperb_ovf` is declared but never used, resulting in unnecessary computation and memory allocation.

VULNERABLE CODE SNIPPET

```
let upperb_ovf = is_overflow_u64(upper_bound_bn); //@audit unused variable
if actual_amount_bn < lower_bound_bn || actual_amount_bn > upper_bound_bn {
    return false;
}
return true;
```

RECOMMENDATION

Fix required:

- Remove the unused variable or integrate it properly within the function logic.

I-01 Contract Pause Check Missing on Version Update

Severity Informational Status ● Open

OVERVIEW

A function responsible for contract updates does not verify whether the contract is paused before executing an update.

RECOMMENDATION

Fix required:

- Implement a pause check before allowing a contract version update to proceed.

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