



SMART CONTRACT SECURITY AUDIT

Griffy

Prediction Market

CLIENT

Griffy

PLATFORM

Fuel

LANGUAGE

Sway, TypeScript

REVIEW

October 2024

DOCUMENT

Document information

Project	Griffy, Prediction Market
Client	Griffy
Platform	Fuel
Language	Sway, TypeScript
Repository	Not recorded in the original report
Commit	Not recorded in the original report
Methods	Not recorded in the original report
Report date	October 2024

Document history

VERSION	DATE	CHANGE
1.0	October 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
	M-01 Missing validation in Commission Calculation	8
	M-02 Missing validation in FEES Calculation	8
	I-01 ZERO_B256 deprecated	9
	Disclaimer	10

Executive summary

About Griffy

Griffy is a decentralized prediction market platform built on the Fuel chain, designed to harness the power of blockchain technology for innovative forecasting. In decentralized prediction markets like Griffy, users can buy and sell prediction shares on the outcomes of various events, ranging from elections and sports to market trends. This is akin to trading in a stock market, where the share price reflects the collective probability of an event's outcome.

Summary of the review

Safe Edges reviewed Griffy, a prediction market on Fuel. Three issues were identified: two Medium and one Informational. The Medium issues were a commission percentage with no upper limit and a missing check for fees consuming the whole requested amount. Griffy fixed all three.

Findings by severity



Findings by status

STATUS	ISSUES
● Resolved	3

Scope

Repository	Not recorded in the original report
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: 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
M-01	Missing validation in Commission Calculation	Medium	● Resolved
M-02	Missing validation in FEES Calculation	Medium	● Resolved
I-01	ZERO_B256 deprecated	Informational	● Resolved

Detailed findings

M-01 Missing validation in Commission Calculation

Severity **Medium** Status ● Resolved

DESCRIPTION

The variable COMMISSION_PERCENTAGE is currently defined as `u64 = 0`. This likely implies that it will be set later:

```
COMMISSION_PERCENTAGE: u64 = 0,
```

There is no explicit maximum limit set for the COMMISSION_PERCENTAGE. This can lead to potential issues where, if not properly validated, the commission could reach unreasonable or unintended levels, such as 100%.

RECOMMENDATION

Add a Maximum Commission Check: Implement a validation that ensures COMMISSION_PERCENTAGE does not exceed a reasonable threshold, such as 20% to 30%, depending on the business logic

M-02 Missing validation in FEES Calculation

Severity **Medium** Status ● Resolved

DESCRIPTION

There is a missing validation check in the code for scenarios where the entire requested amount is consumed by the fees. Specifically, the current logic includes a check for:

```
const userUtxo: Resource[] = await userWallet.getResourcesToSpend([
  [bn(1), VRF_ADDRESS],
  [req.amount * DECI_MULTIPLIER, TTD_ADDRESS],
]);

const amountYesNo = req.amount * DECI_MULTIPLIER - FEES;
```

However, no additional check exists for cases where the result of:

```
const amountYesNo = req.amount * DECI_MULTIPLIER - FEES;
```

becomes zero. In this case, if `req.amount * DECI_MULTIPLIER` equals `FEES`, the calculated `amountYesNo` would become 0. This suggests that the entire requested amount is consumed by the fees, effectively leaving no remaining amount for the operation.

I-01 ZERO_B256 deprecated

Severity Informational Status ● Resolved

DESCRIPTION

Use of the ZERO_B256 is deprecated as per the updated Sway language standards.

Reference: https://fuellabs.github.io/sway/master/std/constants/constant.ZERO_B256.html

```
- const zero_address = Address::from(b256::zero());
```

RECOMMENDATION

It is advised to use `b256::zero()` instead

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