Bluefin RFQ Smart Contract Audit

audited by Asymptotic

Summary

This is a security audit report of the Bluefin RFQ Smart Contract. We identified one medium-severity security vulnerability, 5 low-severity vulnerabilities, and made 3 advisory recommendations for improving code quality. The developers acknowledged and remediated all identified security vulnerabilities and integrated all but one of our advisory recommendations.

Legal Disclaimer

This report is subject to the terms and conditions agreed upon between Asymptotic and Bluefin in the Master Services Agreement.

Scope and Limitations

This security audit report focuses specifically on reviewing the Move smart contract code of Bluefin RFQ. The audit does not analyze or make any claims about other components of the system, including but not limited to:

- Frontend applications and user interfaces
- Backend services and infrastructure
- Off-chain components and integrations
- Deployment procedures and operational security
- Third-party dependencies and external services

The findings and recommendations in this report are limited to the Move code implementation and its immediate interactions within the Sui blockchain environment.

Creating proofs for specifications is on a “best effort” basis. We manually audited the code in instances where formal proofs were not technically feasible.

Files

Repo: https://github.com/fireflyprotocol
Commit: f298500

We inspected the following files:

- `sources/config.move`
- `sources/events.move`
- `sources/gateway.move`
- `sources/vault.move`

Legend

Issue severity

- **Critical** — Vulnerabilities which allow account takeovers or stealing significant funds, along with being easy to exploit.
- **High** — Vulnerabilities which either can have significant impact but are hard to exploit, or are easy to exploit but have more limited impact.
- **Medium** — Moderate risks with notable but limited impact.
- **Low** — Minor issues with minimal security implications.
- **Advisory** — Informational findings for security/code improvements.

M-1: Denial of service by flooding deposit with new coin types

Severity: Medium
Status: Confirmed
Keywords: deposit

Description

There is no privilege check for depositing, and the deposit function auto-adds any coin type it encounters. An attacker could create many coin types, flooding the system and potentially increasing the `supported_coin_types` beyond what Sui accepts. This issue also undermines the concept of a coin type being "supported" by the Vault.

Recommendation

Restrict the creation of coin types to the manager by making it a separate action and raise an `ETokenNotSupported` error on all other cases.

Remediation

✓ **Commit:** d1bb768

L-1: Smokescreen log flooding on deposit

Severity: Low
Status: Confirmed
Keywords: deposit

Description

There is no privilege check or minimum deposit amount, allowing an attacker to cheaply trigger a large number of Deposit events.

Remediation

✓ **Commit:** 202e867

L-2: set_manager incorrectly logs old manager

Severity: Low
Status: Confirmed
Keywords: `set_manager`

Description

The statement `let old_manager = manager;` incorrectly records the new manager instead of preserving the old manager.

Remediation

✓ **Commit:** 8fd53de

L-3: Anyone can create more vaults

Severity: Low
Status: Confirmed
Keywords: `create_rfq_vault`

Description

`create_rfq_vault` is public and does not take an admin cap, allowing anyone to create Vaults. This is contrary to the documentation of the function which states: “cap: Immutable reference to admin cap to ensure the caller is the Admin of the protocol.”

Remediation

✓ **Commit:** d1bb768

L-4: set_manager should also have a zero-address check

Severity: Low
Status: Confirmed
Keywords: `set_manager`

Description

In `create_rfq_vault`, there is a check ensuring that the manager is not the zero address (`assert!(manager != @0, EZeroAddress);`). The `set_manager` function should include a similar check.

Remediation

✓ **Commit:** 5cb05c7

L-5: get_coin_type and get_fee_coin_type are redundant

Severity: Low
Status: Confirmed
Keywords: `get_coin_type`, `get_fee_coin_type`

Description

The function `get_coin_type` appears to be idempotent. The method `type_name::into_string` simply returns the type's name without performing any special ASCII conversion. Converting from UTF8 to bytes and back to UTF8 is therefore redundant. Even if the conversion did change the value, it might be harmful by breaking the one-to-one mapping between the coin type keys in the Vault and the coin type included in the externally-signed Quote. However, a potential collision is prevented by `borrow_mut` failing with `EFieldTypeMismatch`.

Recommendation

1. Use the type name as a key directly, with no conversion.
2. For storing fees, use a wrapper object to store by type name.

Remediation

✓ **Commit:** f239ddb
Further cleanup in 40fc970.

A-1: withdraw cannot create coin types, dead code

Severity: Advisory
Status: Confirmed
Keywords: withdraw

Description

The code within the `withdraw` function intended to create a coin type is dead code. If the `amount` parameter is 0, the function aborts at `assert!(amount > 0, EZeroAmount)`. If the `amount` is greater than 0, the function will fail when attempting to create a new coin type because a new coin type cannot have a non-zero balance. The failure occurs in `internal_withdraw` during `balance::split`.

Remediation

✓ **Commit:** 1c0161c

A-2: Length validation on signature

Severity: Advisory
Status: Confirmed
Keywords: `verify_signature`

Description

In the `verify_signature` function, if the signature is not the correct length, the code aborts without providing a specific error message.

Remediation

✓ **Commit:** 3e5be42

A-3: Generate Quote objects on chain

Severity: Advisory
Status: Confirmed

Description

Generating the Quote object off-chain makes it possible to introduce subtle errors in the code, such as ensuring the uniqueness of the `Quote.id` string. The recommendation is to create the Quote object on-chain, leaving only the signature off-chain. The Quote could even be added to the list of quotes with a value of false.

Remediation

Remediation waived. Developers decided not to generate `Quote`s on-chain; it is not considered a security issue.

About the Auditor

Asymptotic provides a white-glove, formal-verification-based auditing service for Sui smart contracts.

- Analyze customer codebases to create mathematical proofs of security
- Use AI to accelerate specification writing, proof construction, and adjusting specs as code changes
- Focus on real-world security properties that matter for production systems

Asymptotic Team

Andrei Stefanescu

- Led verification of AWS cryptographic algorithms using SAW at Galois
- Created first Ethereum smart contract verification tool
- Three-time International Math Olympiad silver medalist
- PhD in Computer Science from UIUC with David J. Kuck Outstanding Thesis Award
- ACM SIGPLAN Distinguished Paper Award at OOPSLA 2016
- Pioneered K Framework for programming language semantics and verification

Cosmin Radoi

- Created GritQL, a language for large-scale code migration (backed by Founders Fund, 8VC)
- PhD in Computer Science from UIUC focusing on programming languages
- Multiple ACM SIGSOFT Distinguished Paper Awards at top conferences (ICSE, OOPSLA, FSE)
- IBM PhD Fellowship recipient and NSF SBIR Award winner
- Key contributor to K Framework for language semantics and verification
- Research expertise in parallelism, race detection, and performance optimization