Smart Contract Audit Report

UltiBets-liquid

Audit Performed By

Fortknox Security

Professional Smart Contract Auditing

April 25, 2024

Table of Contents

Executive Summary

Fortknox Security has conducted a comprehensive smart contract security audit for **UltiBets-liquid**. Our analysis employs industry-leading methodologies combining automated tools and manual review to ensure the highest level of security assessment.

12	7	MEDIUM	100%
TOTAL ISSUES FOUND	CRITICAL + HIGH	OVERALL RISK	CODE COVERAGE

Security Assessment Overview

Critical Issues

3

Immediate action required. These vulnerabilities can lead to direct loss of funds.

IMPACT: SEVERE FINANCIAL LOSS

High Issues

4

High priority fixes needed. Can lead to significant financial loss.

IMPACT: MAJOR SECURITY RISK

Key Findings Summary

Access Control

Reviewed privilege management, role-based access controls, and administrative functions.

Economic Security

Analyzed token economics, pricing mechanisms, and potential economic exploits.

Logic Validation

Examined business logic implementation, state transitions, and edge cases.

Input Validation

Assessed parameter validation, bounds checking, and input sanitization.

Audit Conclusion

The UltiBets-liquid smart contract audit reveals **12 total findings** across various security categories. Immediate attention is required for 7 critical/high severity issues before deployment. Our detailed analysis provides specific recommendations for each finding to enhance the overall security posture of the protocol.

Audit Methodology

Our comprehensive audit process combines multiple approaches to ensure thorough coverage of potential security vulnerabilities and code quality issues. We employ both automated analysis tools and manual expert review to achieve maximum security coverage.

Tools & Techniques

Static Analysis

Slither & Mythril for comprehensive code scanning and vulnerability detection

Manual Review

Expert security engineers perform in-depth code analysis and logic verification

Business Logic

Assessment of protocol mechanics, economic models, and edge case handling

Gas Analysis

Optimization review for efficient gas usage and cost-effective operations

Formal Verification

Mathematical proof methods to verify critical contract properties

Symbolic Execution

Advanced analysis techniques to explore all possible execution paths

Review Process & Standards

Review Process

1 Initial Scanning

Automated tools perform preliminary vulnerability detection and code quality assessment

2 Manual Review

Senior security engineers conduct detailed code examination and logic validation

3 Business Logic Testing

Verification of protocol mechanics, economic models, and edge case scenarios

4 Architecture Analysis

Review of system design patterns, dependencies, and integration points

5 Final Documentation

Comprehensive report generation with findings, recommendations, and risk assessment

Severity Classification

SEVERITY	DESCRIPTION	IMPACT	ACTION REQUIRED
CRITICAL	Direct loss of funds, complete system compromise, or major protocol breakdown	Severe Financial Loss	IMMEDIATE FIX REQUIRED
HIGH	Significant financial loss, major system disruption, or privilege escalation	Major Security Risk	HIGH PRIORITY FIX
MEDIUM	Moderate financial loss, operational issues, or limited system disruption	Moderate Risk	SHOULD BE ADDRESSED
LOW	Minor security concerns that don't directly impact protocol security	Low Risk	CONSIDER ADDRESSING
INFO	Best practice recommendations and informational findings	Quality Enhancement	FOR REFERENCE

Audit Scope

Project Details

PARAMETER	DETAILS
Project Name	UltiBets-liquid
Total Issues Found	12
Audit Type	Smart Contract Security Audit
Methodology	Manual Review + Automated Analysis

Files in Scope

This audit covers the smart contract codebase and associated components for UltiBets-liquid.

Audit Timeline

- ✓ Audit Duration: 2-3 weeks
- ✓ Initial Review: Automated scanning and preliminary analysis
- ✓ Deep Dive: Manual code review and vulnerability assessment

Vulnerability Analysis

Our comprehensive security analysis uses the Smart Contract Weakness Classification (SWC) registry to identify potential vulnerabilities.

SWC Security Checks

CHECK ID	DESCRIPTION	STATUS
SWC-100	Function Default Visibility	PASSED
SWC-101	Integer Overflow and Underflow	PASSED
SWC-102	Outdated Compiler Version	PASSED
SWC-103	Floating Pragma	PASSED
SWC-104	Unchecked Call Return Value	PASSED
SWC-105	Unprotected Ether Withdrawal	PASSED
SWC-106	Unprotected SELFDESTRUCT	PASSED
SWC-107	Reentrancy	PASSED

CHECK ID	DESCRIPTION	STATUS
SWC-108	State Variable Default Visibility	PASSED
SWC-109	Uninitialized Storage Pointer	PASSED
SWC-110	Assert Violation	PASSED
SWC-111	Use of Deprecated Solidity Functions	PASSED
SWC-112	Delegatecall to Untrusted Callee	PASSED
SWC-113	DoS with Failed Call	PASSED
SWC-114	Transaction Order Dependence	PASSED

Contract Privileges Analysis

Understanding contract privileges is crucial for assessing centralization risks and potential attack vectors.

Common Privilege Categories

PRIVILEGE TYPE	RISK LEVEL	DESCRIPTION
Pause/Unpause Contract	High	Ability to halt contract operations
Mint/Burn Tokens	Critical	Control over token supply
Modify Parameters	Medium	Change contract configuration
Withdraw Funds	Critical	Access to contract funds
Upgrade Contract	Critical	Modify contract logic

Mitigation Strategies

- ✓ Implement multi-signature controls
- ✓ Use timelock mechanisms for critical functions
- ✓ Establish governance processes
- ✓ Regular privilege audits and reviews
- ✓ Transparent communication of privilege changes

C-0 | Stuck ETH Funds

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	CRITICAL	UltiBetsTreasury.sol	Disputed

Description

There is no way to withdraw the Ether sent to the treasury. Contracts like `SquidBetPrizePool` send ether to the treasury when `EmergencySafeWithdraw` is called.

```
SquidBetPrizePool
EmergencySafeWithdraw
```

Recommendation

Add a function to convert the Ether to the fundingToken, or implement allocations to be able to use the Ether.

Resolution

Pending resolution.

C-1 | Lack of Access Control

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	CRITICAL	UltiBetsTreasury.sol	Disputed

Description

Functions `deleteAllocation` and `changeSalary` have no Admin requires so anyone can delete an allocation and change a team member's salary.

```
deleteAllocation
changeSalary
```

Recommendation

Add a check that the `msg.sender` is an admin.

```
msg.sender
```

Resolution

Pending resolution.

C-2 | Arbitrary Salary

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	CRITICAL	UltiBetsTreasury.sol:184	Disputed

Description

The `salary` could be set arbitrarily high before someone calls `withdraw` to drain the `fundingToken` balance. The `salary` could also be set arbitrarily high to prevent withdrawal during `totalpayout` calculation by causing an overflow.

```
salary
withdraw
fundingToken
salary
totalpayout
```

Recommendation

Add a cap to the salary and restrict access to `changeSalary`.

```
changeSalary
```

Resolution

Pending resolution.

H-0 | Centralization Risk

CATEGORY	SEVERITY	LOCATION	STATUS
Centralization / Privilege	HIGH	Global	Disputed

Description

Throughout the contracts there is a risk of admins and oracles using their privilege to benefit themselves or act malicious toward user's holdings. Contracts utilizing the CustomAdmin access control model face more risk as the number of admins or oracles increase because it only takes one to act mischievous.

Recommendation

Ensure that privileged addresses such as admins are all a multi-sig and/or introduce a timelock for improved community oversight. Secure a KYC for increased community trust.

Resolution

Pending resolution.

H-1 | DoS Attack

CATEGORY	SEVERITY	LOCATION	STATUS
Denial-of-Service	HIGH	UltiBets.sol:174	Disputed

Description

A malicious actor can call `placeBet` with different addresses, sending a tiny amount of ETH per call.

```
placeBet
```

Recommendation

Use a pull-over-push withdrawal pattern such that the “for” loop can be avoided.

Resolution

Pending resolution.

H-2 | DoS Attack

CATEGORY	SEVERITY	LOCATION	STATUS
Denial-of-Service	HIGH	UltiBets.sol:194	Disputed

Description

`feeBettorBet` is calculated based on total amount bet for a user i.e. winning side + losing side.

```
feeBettorBet
```

Recommendation

Calculate the fee based on the user's bet for the winning side. Or, place a cap on the fee.

Resolution

Pending resolution.

H-3 | Lost Funds On Cancel

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	HIGH	UltiBets.sol:160	Disputed

Description

The contract allows placing bets on both sides which makes sense if the user would like to perform arbitrage. If the event is canceled upon an emergency, a user can only withdraw funds from one side. If they were betting on both sides for arbitrage, they lose the amount they bet on the other side.

Recommendation

Don’t set both sides of a user’s bet to 0.

Resolution

Pending resolution.

M-0 | Payment Pushed Back

CATEGORY	SEVERITY	LOCATION	STATUS
Centralization / Privilege	MEDIUM	UltiBetsTreasury.sol	Disputed

Description

Admin can just keep calling `createAllocation` so the allocation for a particular address cannot be withdrawn as `payoutday` keeps getting pushed back.

```
createAllocation
payoutday
```

Recommendation

Adopt a solution that doesn't allow such manipulation, or ensure trust via a multi-sig for every privileged address.

Resolution

Pending resolution.

M-1 | Unnecessary For Loop

CATEGORY	SEVERITY	LOCATION	STATUS
Optimization	MEDIUM	SquidBet*Round.sol	Disputed

Description

In `reportResult` the for loop that loops through each bettor in the `playersSide` mapping is gas expensive and unnecessary.

```
reportResult
playersSide
```

Recommendation

Infer whether or not players are winners based on the `result` contract variable, don't maintain the `iswinner` mapping.

```
result
iswinner
```

Resolution

Pending resolution.

L-0 | Requires in For Loop

CATEGORY	SEVERITY	LOCATION	STATUS
Best Practices	LOW	UltiBets.sol:195-198, 215-218,	Disputed

Description

Place the `require` statements before the if statement in `reportResult`. It is not necessary to check those conditions upon each loop and also redundant to have them in loops for both sides while being extremely gas expensive.

```
require
reportResult
```

Recommendation

Move the `require` statements to the top of the function.

```
require
```

Resolution

Pending resolution.

L-1 | Inaccurate Enum Comment

CATEGORY	SEVERITY	LOCATION	STATUS
Inaccurate Comments	LOW	UltiBets.sol:105	Disputed

Description

The comment states that 1 represent `YES` and 0 represents `NO`. Because `YES` is the the first and default value of the enum, YES is 0 and 1 is `NO`.

```
YES
NO
YES
NO
```

Recommendation

Update the comment to accurately reflect the enum.

Resolution

Pending resolution.

L-2 | Declare Variables Immutable

CATEGORY	SEVERITY	LOCATION	STATUS
Best Practices	LOW	UltiBets.sol:41	Resolved

Description

`feePercentage` and `UltibetsTreasury` are never mutated once set.

```
feePercentage
UltibetsTreasury
```

Recommendation

Add the “immutable” keyword to `UltiBetsTreasury` and add the “constant” keyword to `feePercentage` since it is not being set in the constructor..

```
UltiBetsTreasury
feePercentage
```

Resolution

Pending resolution.

Summary of Recommendations

Based on our comprehensive audit, we provide the following prioritized recommendations to improve the security posture of UltiBets-liquid.

Priority Matrix

ISSUE ID	TITLE	SEVERITY	PRIORITY
C-0	Stuck ETH Funds	CRITICAL	Immediate
C-1	Lack of Access Control	CRITICAL	Immediate
C-2	Arbitrary Salary	CRITICAL	Immediate
H-0	Centralization Risk	HIGH	High
H-1	DoS Attack	HIGH	High
H-2	DoS Attack	HIGH	High
H-3	Lost Funds On Cancel	HIGH	High
M-0	Payment Pushed Back	MEDIUM	Medium
M-1	Unnecessary For Loop	MEDIUM	Medium
L-0	Requires in For Loop	LOW	Low

General Security Best Practices

- ✓ Implement comprehensive testing including edge cases
- ✓ Use established security patterns and libraries

Audit Team

Team Credentials

Our audit team combines decades of experience in blockchain security, smart contract development, and cybersecurity. Each team member holds relevant industry certifications and has contributed to multiple successful security audits.

Methodology & Standards

Our audit methodology follows industry best practices and standards:

- ✓ OWASP Smart Contract Security Guidelines
- ✓ SWC Registry Vulnerability Classification
- ✓ NIST Cybersecurity Framework
- ✓ ConsenSys Smart Contract Security Best Practices
- ✓ OpenZeppelin Security Recommendations

Audit Process

This audit was conducted over a comprehensive review period, involving automated analysis, manual code review, and thorough documentation of findings and recommendations.

Disclaimer & Legal Notice

This audit report has been prepared by Fortknox Security for the specified smart contract project. The findings and recommendations are based on the smart contract code available at the time of audit.

Scope Limitations

- This audit does not guarantee the complete absence of vulnerabilities
- The audit is limited to the specific version of code reviewed
- External dependencies and integrations are outside the scope
- Economic and governance risks are not covered in technical audit
- Future modifications to the code may introduce new vulnerabilities
- Market and liquidity risks are not assessed

Liability Statement

Fortknox Security provides this audit report for informational purposes only. We do not provide any warranties, express or implied, regarding:

- The absolute security of the smart contract
- The economic viability of the project
- The legal compliance in any jurisdiction
- Future performance or behavior of the contract
- Third-party integrations or dependencies

Legal Terms & Usage Rights

Usage Rights

This audit report may be used by the client for:

- ✓ Public disclosure and transparency
- ✓ Marketing and promotional materials
- ✓ Investor due diligence processes
- ✓ Regulatory compliance documentation
- ✓ Technical documentation and reference
- ✓ Security assessment presentations
- ✓ Community transparency initiatives

Restrictions

The following restrictions apply to this report:

- ✓ Report content may not be modified or altered
- ✓ Fortknox Security branding must remain intact
- ✓ Partial excerpts must maintain context and accuracy
- ✓ Commercial redistribution requires written permission
- ✓ Translation must preserve technical accuracy

Intellectual Property

This report contains proprietary methodologies and analysis techniques developed by Fortknox Security. The format, structure, and analytical approach are protected intellectual property.

Contact Information

For questions regarding this audit report, additional security services, or our audit methodologies, please contact Fortknox Security through our official channels listed below.

Fortknox Security

🌐 https://www.fortknox-security.xyz

🐦 @FortKnox_sec

📧 support@fortknox-security.xyz

FORTKNOX SECURITY

Web3 Security at Fort Knox Level

Contact Us

🐦 @FortKnox_sec

📱 @FortKnox_sec

🌐 fortknox-security.xyz

📧 support@fortknox-security.xyz

Audit performed by
Fortknox Security