



# Smart Contract Audit Report

Baseline-Dedicate

## Audit Performed By

Fortknox Security  
Professional Smart Contract Auditing

June 30, 2024



## Table of Contents

Executive Summary	3
Audit Methodology	5
Audit Scope	8
Vulnerability Analysis	9
Contract Privileges Analysis	11
Detailed Findings	8
Recommendations	9
Audit Team	21
Disclaimer & Legal Notice	22
Legal Terms & Usage Rights	23



## Executive Summary

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

Q

8

TOTAL  
ISSUES  
FOUND

⚠

0

CRITICAL  
+ HIGH

i

LOW

✓

100%

CODE  
COVERAGE

## Security Assessment Overview



### Critical Issues

0

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

IMPACT: SEVERE FINANCIAL LOSS



### High Issues

0

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 Baseline-Dedicate smart contract audit reveals **8 total findings** across various security categories. **No critical or high severity issues were identified.** 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	Baseline-Dedicate
Total Issues Found	8
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 Baseline-Dedicate.

## 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



# M-0 | Operations Using Outdated Circulating Supply

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	MEDIUM	CREDT.v1.sol: 226	Resolved

## Description

When a credit is defaulted, its collateral is sent directly to the `BPOOL` contract through the `_burnDefaultedCollateral` function. This creates a temporary condition where the circulating supply is higher than expected, if the market making operations are executed before defaulting the expired loans.

```
BPOOL  
_burnDefaultedCollateral
```

## Recommendation

Ensure `defaultOutstanding` is executed at the start of market making operations.

```
defaultOutstanding
```

## Resolution

Baseline Team: The issue was resolved.



## M-1 | Arbitrage Attack After Slide Operation

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	MEDIUM	MarketMaking.sol: 255	Acknowledged

### Description

The market making `slide` operation rebalances the liquidity structure when active tick goes below a certain threshold.

slide

### Recommendation

Verify the initial liquidity structure does not allow these arbitrage attacks to take place.

### Resolution

Baseline Team: Acknowledged.



## M-2 | getLeverageFactor DoS

CATEGORY	SEVERITY	LOCATION	STATUS
DoS	MEDIUM	MarketMaking.sol: 339	Resolved

### Description

In the `getLeverageFactor` function the resulting `leverageFactor` is computed by dividing the `totalCollateral` by the `_bAssetsCirculating - totalCollateral`.

```
getLeverageFactor  
leverageFactor  
totalCollateral  
_bAssetsCirculating  
totalCollateral
```

### Recommendation

Add a case to handle the scenario where all circulating assets are being used as collateral in the `getLeverageFactor` function.

```
getLeverageFactor
```

### Resolution

Baseline Team: The issue was resolved.



# L-0 | Slide May Result In Equal Anchor And Discovery Liquidity

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	LOW	MarketMaking.sol: 300	Acknowledged

## Description

During slide, if `liquidityA` is equal to `liquidityThreshold` then both anchor and discovery will be re-deployed with the same liquidity.

```
liquidityA  
liquidityThreshold
```

## Recommendation

Reduce anchor's liquidity by the inverse liquidity premium if `liquidityA >= liquidityThreshold`. Alternatively, consider increasing discovery's liquidity in this scenario.

```
liquidityA >= liquidityThreshold
```

## Resolution

Baseline Team: Acknowledged.



# L-1 | Strict Inequality For Anchor Threshold Comparison

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	LOW	Partially Resolved	Partially Resolved

## Description

In the `sweep` function the `surplusReservesD` are allocated to the anchor and floor positions depending on if it would put the anchor above the desired threshold.

```
sweep  
surplusReservesD
```

## Recommendation

Change the `reservesA + surplusReservesD < thresholdReservesA` comparison to `reservesA + surplusReservesD <= thresholdReservesA`.

```
reservesA + surplusReservesD < thresholdReservesA  
reservesA + surplusReservesD <= thresholdReservesA
```

## Resolution

Baseline Team: The issue was resolved



## L-2 | Reserve Token Should Use safeTransfer

Category	Severity	Location	Status
Logical Error	LOW	CreditFacility.sol: 194	Resolved

### Description

The reserve token can be any arbitrary token in future pairs, in order to maintain compatibility with as many tokens as possible, `safeTransfer` and `safeTransferFrom` ought to be used to validate returned values.

`safeTransfer`  
`safeTransferFrom`

### Recommendation

Use `safeTransfer` and `safeTransferFrom` when transferring the reserve token throughout the codebase.

`safeTransfer`  
`safeTransferFrom`

### Resolution

Baseline Team: The issue was resolved.



## L-3 | Inconsistent Default Pattern Array Lengths

Category	Severity	Location	Status
Logical Error	LOW	MarketMaking.sol: 82	Resolved

### Description

In the `configureDependencies` function the dependencies array is initialized with a length of 3, however only 2 entries are written to.

```
configureDependencies
```

### Recommendation

Reduce the size of the dependencies array in the `configureDependencies` function as well as the requests array in the `requestPermissions` function.

```
configureDependencies  
requestPermissions
```

### Resolution

Baseline Team: The issue was resolved.



## L-4 | Superfluous brs Address

CATEGORY	SEVERITY	LOCATION	STATUS
	LOW		Resolved

### Description

In the `BPOOL` contract there is an address variable named `brs` which is not referenced nor assigned to.

BPOOL  
brs

### Recommendation

Remove the `brs` address state variable.

brs

### Resolution

Baseline Team: The issue was resolved.



# Summary of Recommendations

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

## Priority Matrix

Issue ID	Title	Severity	Priority
M-0	Operations Using Outdated Circulating Supply	MEDIUM	Medium
M-1	Arbitrage Attack After Slide Operation	MEDIUM	Medium
M-2	getLeverageFactor Dos	MEDIUM	Medium
L-0	Slide May Result In Equal Anchor And Discovery Liquidity	LOW	Low
L-1	Strict Inequality For Anchor Threshold Comparison	LOW	Low
L-2	Reserve Token Should Use safeTransfer	LOW	Low
L-3	Inconsistent Default Pattern Array Lengths	LOW	Low
L-4	Superfluous brs Address	LOW	Low

## General Security Best Practices

- ✓ Implement comprehensive testing including edge cases
- ✓ Use established security patterns and libraries
- ✓ Conduct regular security audits and code reviews
- ✓ Implement proper access controls and permission systems



## 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](mailto:support@fortknox-security.xyz)



# FORTKNOX SECURITY

Web3 Security at Fort Knox Level

## Contact Us

 @FortKnox\_sec

 @FortKnox\_sec

 [fortknox-security.xyz](http://fortknox-security.xyz)

 [support@fortknox-security.xyz](mailto:support@fortknox-security.xyz)

Audit performed by  
Fortknox Security