Smart Contract Audit Report

MIMSwap

Audit Performed By

Fortknox Security

Professional Smart Contract Auditing

March 15, 2024

Table of Contents

Executive Summary

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

28	6	MEDIUM	100%
TOTAL ISSUES FOUND	CRITICAL + HIGH	OVERALL RISK	CODE COVERAGE

Security Assessment Overview

Critical Issues

1

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

IMPACT: SEVERE FINANCIAL LOSS

High Issues

5

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 MIMSwap smart contract audit reveals **28 total findings** across various security categories. Immediate attention is required for 6 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	MIMSwap
Total Issues Found	28
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 MIMSwap.

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	WARNING

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 | Claiming Yield DoS

CATEGORY	SEVERITY	LOCATION	STATUS
DoS	CRITICAL	Global	Resolved

Description

The `BlastMagicLP` and `BlastOnboarding` contracts do not expect to hold native Ether and therefore will not accrue claimable native ether yield, however they still attempt to claim native yield with the `BlastYields.claimAllNativeYields` function.

```
BlastMagicLP
BlastOnboarding
BlastYields.claimAllNativeYields
```

Recommendation

Either do not invoke the `claimAllNativeYields` function if there is no native yield to be claimed, or separate the yield claiming logic into independent functions for each type of yield.

```
claimAllNativeYields
```

Resolution

Pending resolution.

H-0 | Weth Transferred From The Wrong Address

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	HIGH	Router.sol: 78	Resolved

Description

In the `Router.createPoolETH` function, a deposit to the weth contract is made but then a subsequent weth `safeTransferFrom` is made from the `msg.sender` to the newly created pool.

```
Router.createPoolETH
safeTransferFrom
msg.sender
```

Recommendation

Use the weth `safeTransferFrom` to transfer from the `Router` contract to the newly created pool rather than from the `msg.sender`.

```
safeTransferFrom
Router
msg.sender
```

Resolution

Pending resolution.

H-1 | twapUpdate Overflow DoS

CATEGORY	SEVERITY	LOCATION	STATUS
DoS	HIGH	MagicLP.sol	Resolved

Description

In MagicLP.sol, `_twapUpdate` is called every time `setReserve` or `sync` is called. The update adds to `_BASE_PRICE_CUMULATIVE_LAST_` which is an ever increasing value.

```
_twapUpdate
setReserve
sync
_BASE_PRICE_CUMULATIVE_LAST_
```

Recommendation

Wrap with an `unchecked` block to allow for overflow.

```
unchecked
```

Resolution

Pending resolution.

H-2 | previewAddLiquidity Incorrect quoteBalance

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	HIGH	Router.sol: 90	Resolved

Description

In the `previewAddLiquidity` function the `quoteBalance` is intended to account for the current balance

```
previewAddLiquidity
quoteBalance
```

Recommendation

Correct the `quoteBalance` to be:

```
quoteBalance
```

Resolution

Pending resolution.

H-3 | Lacking Gas Yields Claiming Logic

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	HIGH	Global	Resolved

Description

There are several contracts in the Abracadabra and Mimswap systems that cannot claim gas yields

Recommendation

Consider implementing gas yield claiming logic for these contracts.

Resolution

Pending resolution.

H-4 | Native Yield Token Yields Cannot Be Configured After Deployment

CATEGORY	SEVERITY	LOCATION	STATUS
DoS	HIGH	IBlast.sol: 82	Resolved

Description

In the IERC20Rebasing interface the configure function is defined to return a YieldMode enum value.

Recommendation

Correct the `IERC20Rebasing` interface to return a `uint256` value from the configure function rather

```
IERC20Rebasing
uint256
```

Resolution

Pending resolution.

M-0 | Risk Of Function Selector And Storage Collision

CATEGORY	SEVERITY	LOCATION	STATUS
Best Practices	MEDIUM	BlastOnboarding.sol	Resolved

Description

The `BlastOnboarding` contract is itself a proxy, however it has several declared storage variables and functions. As a result the contract is prone to storage and function selector collision.

```
BlastOnboarding
```

Recommendation

Any `bootstrapper` implementation that is used in the future should be rigorously verified to have no collisions with the existing function selectors or storage slots in the `BlastOnboarding` contract.

```
bootstrapper
BlastOnboarding
```

Resolution

Pending resolution.

M-1 | Predictable MagicLP Salt

CATEGORY	SEVERITY	LOCATION	STATUS
Gaming	MEDIUM	Factory.sol	Resolved

Description

The salt used to deploy the new MagicLP contract in the factory is based upon replicable values that any malicious user may pass in.

Recommendation

Consider including the `msg.sender` in the salt for pool creation.

```
msg.sender
```

Resolution

Pending resolution.

M-2 | previewAddLiquidity Disagrees With _adjustLiquidity

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	MEDIUM	Router.sol: 492, 493	Resolved

Description

In the `_adjustAddLiquidity` function the `baseAdjustedInAmount` and `quoteAdjustedInAmount` are adjusted without considering tokens that may be sitting in the `MagicLP` contract and unaccounted for in the reserves.

```
_adjustAddLiquidity
baseAdjustedInAmount
quoteAdjustedInAmount
MagicLP
```

Recommendation

Consider accounting for any additional tokens in the lp contract to match the behavior of the `previewAddLiquidity` function.

```
previewAddLiquidity
```

Resolution

Pending resolution.

M-3 | Blast Point Remunerations May Be Gamed

CATEGORY	SEVERITY	LOCATION	STATUS
Gaming	MEDIUM	Global	Resolved

Description

In the `BlastOnboarding` contract users may deposit their token balances and will be remunerated with the points they would have otherwise received if they didn't deposit into the contract.

```
BlastOnboarding
```

Recommendation

In the case of the `BlastOnboarding` contract, consider requiring that users have locked amounts to reward them with point distributions. Otherwise ensure there is no way for users to predict when the off-chain point remuneration will occur.

```
BlastOnboarding
```

Resolution

Pending resolution.

M-4 | Disabled Native Yield Tokens Cause Loss Of Funds

CATEGORY	SEVERITY	LOCATION	STATUS
Unexpected Behavior	MEDIUM	BlastBox.sol	Acknowledged

Description

If the owner does not want to enable native yield for a token with function `setTokenEnabled`, that token will have the default mode which is AUTOMATIC for WETH and USDB. When a token is in AUTOMATIC mode, the balance of the token in the contract increases as yield is gained. However, the DegenBox contract is unable to support rebasing tokens:

```
setTokenEnabled
```

Recommendation

Consider adding feature that allows the owner to set token yield mode to VOID to ensure compatibility when the owner does not want token yield enabled.

Resolution

Pending resolution.

M-5 | Lacking LP Validations Allows For Malicious Intent

CATEGORY	SEVERITY	LOCATION	STATUS
Validation	MEDIUM	Router.sol	Resolved

Description

Anyone can create a pool with the same base/quote tokens as well as the same params (i, k) as an 'official' `MIMSwap` pool. An attacker could create such pools to steal liquidity away, or even worse implement such pools with rug pull functions.

```
MIMSwap
```

Recommendation

In the `Router` contract, validate the lp address against the `pools` mapping in the `Factory` before allowing the function to continue executing.

```
Router
pools
Factory
```

Resolution

Pending resolution.

M-6 | Governor Contract Does Not Configure Blast Points

CATEGORY	SEVERITY	LOCATION	STATUS
Unexpected Behavior	MEDIUM	BlastGovernor.sol	Resolved

Description

The blast governor contract does not configure blast points, this will cause the contract to miss out on points that can later be used to receive an airdrop of tokens from blast.

Recommendation

Add a call to blast points configure function to allow the governor contract's points to be harvested. Add this to constructor `BlastPoints.configure();`

```
BlastPoints.configure();
```

Resolution

Pending resolution.

L-0 | Unable To Maximize Gas Yields

CATEGORY	SEVERITY	LOCATION	STATUS
Optimization	LOW	Global	Resolved

Description

In the BlastBox and BlastMagicLP contracts the single claimYields function claims both gas yields

Recommendation

Consider separating the `claimYields` function into two functions where one can be used to claim

```
claimYields
```

Resolution

Pending resolution.

L-1 | Typo

CATEGORY	SEVERITY	LOCATION	STATUS
Typo	LOW	MagicLP.sol: 369	Resolved

Description

In the comment on line 369, the word occurring is misspelled as “occuring”.

Recommendation

Replace “occuring” with occurring.

Resolution

Pending resolution.

L-2 | Lacking Zero Address Validation

CATEGORY	SEVERITY	LOCATION	STATUS
Best Practices	LOW	BlastGovernor.sol: 38	Resolved

Description

In the `setFeeTo` function there is no validation that the `_feeTo` address is nonzero.

```
setFeeTo
_feeTo
```

Recommendation

Add validation that the `feeTo` address cannot be assigned to 0.

```
feeTo
```

Resolution

Pending resolution.

L-3 | Potentially Unexpected Pool Creator Recorded

CATEGORY	SEVERITY	LOCATION	STATUS
Unexpected Behavior	LOW	Factory.sol: 82	Resolved

Description

When deploying a new `MagicLP` pool through the `Router.createPool` or `Router.createPoolETH` functions, the creator of the pool will be recorded as the Router contract, with the pool being added to the `userPools` mapping entry for the router contract.

```
MagicLP
Router.createPool
Router.createPoolETH
userPools
```

Recommendation

Consider if this is expected behavior or not. If the original creator of the pool ought to be recorded, then consider creating a dedicated function for the `Router` to be able to pass along the address of the user who created the pool through the `Router` contract.

```
Router
Router
```

Resolution

Pending resolution.

L-4 | Lacking SafeCast

CATEGORY	SEVERITY	LOCATION	STATUS
Validation	LOW	MagicLP.sol: 356, 405	Resolved

Description

In the `buyShares` and `sellShares` functions token amounts are casted to `uint112` variables for the

```
buyShares
sellShares
uint112
```

Recommendation

Use the `SafeCastLib` when casting token amounts to `uint112` variables in the buyShares and

```
SafeCastLib
uint112
```

Resolution

Pending resolution.

L-5 | Quote Target Rounded To 0

CATEGORY	SEVERITY	LOCATION	STATUS
DoS	LOW	MagicLP.sol: 379	Resolved

Description

In the `buyShares` function the `_QUOTE_TARGET_` may be rounded to 0 when the quote token has less decimals than the base token. As a result swapping via `sellBase` will initially be DoS'ed as it relies upon the `_SolveQuadraticFunctionForTrade` while using the quote target as V0, since R is initially assigned to 1.

```
buyShares
_QUOTE_TARGET_
sellBase
_SolveQuadraticFunctionForTrade
```

Recommendation

Consider removing this edge case entirely by reverting when the `_QUOTE_TARGET_` rounds to 0 when adding initial liquidity with the `buyShares` function.

```
_QUOTE_TARGET_
buyShares
```

Resolution

Pending resolution.

L-6 | previewAddLiquidity Can Give Inaccurate Results

CATEGORY	SEVERITY	LOCATION	STATUS
Unexpected Behavior	LOW	Router.sol: 117-128	Resolved

Description

In the `previewAddLiquidity` function, it is possible for the function to return a `baseAdjustedInAmount`, `quoteAdjustedInAmount`, and shares combination that cannot be achieved as a result of rounding when assigning the adjusted amounts. For example:

```
previewAddLiquidity
baseAdjustedInAmount
quoteAdjustedInAmount
```

Recommendation

Resolution

Pending resolution.

L-7 | System Incompatible With Esoteric Token Pairs

CATEGORY	SEVERITY	LOCATION	STATUS
Documentation	LOW	Global	Resolved

Description

The system represents the target price using `1e18 * quoteToken / baseToken`, however when the

```
1e18 * quoteToken /
          baseToken
```

Recommendation

Be sure to document that the system is not designed to be compatible with `quoteTokens` and

```
quoteTokens
```

Resolution

Pending resolution.

L-8 | Self-Governed Contracts Cannot Change Configuration

CATEGORY	SEVERITY	LOCATION	STATUS
Documentation	LOW	Global	Resolved

Description

Upon `BlastBox` and `BlastMagicLP` deployment/initialization, the yield is configured to be in claimable mode, as well as the governor for the contract is set to be the contract itself as part of the `configureDefaultClaimables` function call: `governorMap[msg.sender] = governor;`

```
BlastBox
BlastMagicLP
configureDefaultClaimables
governorMap[msg.sender] =
          governor;
```

Recommendation

Document and be aware that since `address(this)` is the address of the governor, no further Blast configuration can be performed.

```
address(this)
```

Resolution

Pending resolution.

L-9 | Potential Read-Only Reentrancy

CATEGORY	SEVERITY	LOCATION	STATUS
Reentrancy	LOW	MagicLP.sol: 224, 228	Resolved

Description

The `getQuoteInput` and `getBaseInput` functions rely on the balance of the `MagicLP` contract, which can be manipulated with the use of the `flashLoan` function.

```
getQuoteInput
getBaseInput
MagicLP
flashLoan
```

Recommendation

Be sure to carefully document these risks for integrating parties, otherwise explicitly implement the `nonReadReentrant` modifier from the Solady library to remove this attack surface.

```
nonReadReentrant
```

Resolution

Pending resolution.

L-10 | BlastCauldron Master Contract Can Be Initialized

CATEGORY	SEVERITY	LOCATION	STATUS
Unexpected Behavior	LOW	Router.sol: 62, 79	Resolved

Description

Currently there is no mechanism to prevent the initialization of the `BlastCauldron` master contract.

```
BlastCauldron
```

Recommendation

Consider implementing validation that does not allow the `BlastCauldron` master contract to be initialized.

```
BlastCauldron
```

Resolution

Pending resolution.

L-11 | BlastOnboarding DoS For Max Transfer Tokens

CATEGORY	SEVERITY	LOCATION	STATUS
DoS	LOW	BlastOnBoarding.sol	Acknowledged

Description

Some tokens allow users to transfer their entire balance by specifying `type(uint256).max` as the transfer amount. In such a case where the first user to deposit provides `type(uint256).max` as an `amount` parameter to `deposit`, the totals mapping entry for that token will become the maximum for the `uint256` type and therefore no other users will be allowed to onboard new tokens.

```
type(uint256).max
type(uint256).max
amount
deposit
uint256
```

Recommendation

There are no known tokens with this behavior on the Blast network currently, however this should be carefully considered when supporting new tokens for the `BlastOnboarding` contract.

```
BlastOnboarding
```

Resolution

Pending resolution.

L-12 | Blast Points Address Configured For Testnet

CATEGORY	SEVERITY	LOCATION	STATUS
Warning	LOW	BlastPoints.sol	Resolved

Description

In the library `BlastPoints`, the `BLAST_POINTS` address is hard coded to the testnet implementation. `IBlastPoints public constant BLAST_POINTS = IBlastPoints(0x2fc95838c71e76ec69ff817983BFf17c710F34E0);`

```
BlastPoints
BLAST_POINTS
IBlastPoints public constant BLAST_POINTS =
          IBlastPoints(0x2fc95838c71e76ec69ff817983BFf17c710F34E0);
```

Recommendation

Be sure to update this address for mainnet deployment.

Resolution

Pending resolution.

L-13 | Potential Reentrancy Risk

CATEGORY	SEVERITY	LOCATION	STATUS
Reentrancy	LOW	Router.sol	Acknowledged

Description

In the `Router` contract, `addLiquidityETH` function, a refund of unused ETH is done before transferring token and `addLiquidity` which opens up the possibility of reentrancy attacks.

```
Router
addLiquidityETH
addLiquidity
```

Recommendation

While no attack path was found, it is best to follow CEI pattern and move the refund of ETH to the last action in the function.

Resolution

Pending resolution.

L-14 | BlastCauldronV4 Deployment Bricked

CATEGORY	SEVERITY	LOCATION	STATUS
DoS	LOW	BlastWrappers.sol: 59	Acknowledged

Description

When a new `BlastCauldronV4` is deployed, a malicious actor may front-run the init function and use the cook function to trigger an `ACTION_CALL` to the Blast yields contract that will configure a malicious governor for the `Cauldron`. As a result the cauldron deployment will then be unusable as the call to init will revert upon attempting to call `BlastYields.configureDefaultClaimables`.

```
BlastCauldronV4
ACTION_CALL
Cauldron
BlastYields.configureDefaultClaimables
```

Recommendation

Be aware of this risk during deployments and consider always initializing a cauldron in the same transaction in which it is deployed.

Resolution

Pending resolution.

Summary of Recommendations

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

Priority Matrix

ISSUE ID	TITLE	SEVERITY	PRIORITY
C-0	Claiming Yield DoS	CRITICAL	Immediate
H-0	Weth Transferred From The Wrong Address	HIGH	High
H-1	twapUpdate Overflow DoS	HIGH	High
H-2	previewAddLiquidity Incorrect quoteBalance	HIGH	High
H-3	Lacking Gas Yields Claiming Logic	HIGH	High
H-4	Native Yield Token Yields Cannot Be Configured After Deployment	HIGH	High
M-0	Risk Of Function Selector And Storage Collision	MEDIUM	Medium
M-1	Predictable MagicLP Salt	MEDIUM	Medium
M-2	previewAddLiquidity Disagrees With _adjustLiquidity	MEDIUM	Medium
M-3	Blast Point Remunerations May Be Gamed	MEDIUM	Medium

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