Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.03.12, the SlowMist security team received the EureXa team's security audit application for EureXa, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

EureXa is a blockchain-based AI model asset management and trading ecosystem.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Missing events access control	Malicious Event Log Audit	Suggestion	Acknowledged
N2	Non-standard ERC-20 implementation in	Others	Medium	Acknowledged

NO	Title	Category	Level	Status
	Eurexa_coin contract			
N3	Missing zero address validation	Design Logic Audit	Suggestion	Acknowledged
N4	Improper admin removal in ModelAssetContract	Gas Optimization Audit	Suggestion	Acknowledged
N5	Excessive admin privileges across protocol contracts	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit scope:

https://testnet.bscscan.com/address/0x68424a6f4f0C7Fae3b0cb8C106e11305256321d2#code

https://testnet.bscscan.com/address/0x3C78334D111466CC040FB754b66C68d544D183cC#code

https://testnet.bscscan.com/address/0x73120021464bE4D6427B88D35FF49B31E81F61f5#code

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

Eurexa_coin			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
tokenizeModel	External	Can Modify State	validModelId
purchaseTokens	External	Payable	nonReentrant validModelId

Eurexa_coin			
withdrawEth	External	Can Modify State	onlyAdmin
updateTotalSupply	External	Can Modify State	onlyAdmin
updateModelOwnerPercentage	External	Can Modify State	onlyAdmin
updateEthToTokenRatio	External	Can Modify State	onlyAdmin
transferAdmin	External	Can Modify State	onlyAdmin
transfer	External	Can Modify State	validModelId
approve	External	Can Modify State	validModelId
transferFrom	External	Can Modify State	validModelId
balanceOf	External	-	validModelId
allowance	External	-	validModelId
getAssetDetails	External	-	validModelId
isModelTokenized	External	-	validModelId
getModelTokenizationDetails	External	-	validModelId
getContractConfig	External	-	-
_transfer	Internal	Can Modify State	-
<Receive Ether>	External	Payable	-

ModelAssetContract			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
addAdmin	Public	Can Modify State	onlyAdmin
removeAdmin	Public	Can Modify State	onlyAdmin
createModel	Public	Can Modify State	onlyAdmin

ModelAssetContract			
setModelType	Public	Can Modify State	onlyAdmin modelExists
setGrFile	Public	Can Modify State	onlyAdmin modelExists
setGitHubUsername	Public	Can Modify State	onlyAdmin modelExists
setGPU	Public	Can Modify State	onlyAdmin modelExists
setSimulation	Public	Can Modify State	onlyAdmin modelExists
setAlgorithm	Public	Can Modify State	onlyAdmin modelExists
setDataset	Public	Can Modify State	onlyAdmin modelExists
setParameters	Public	Can Modify State	onlyAdmin modelExists
setSensorData	Public	Can Modify State	onlyAdmin modelExists
setParentModel	Public	Can Modify State	onlyAdmin modelExists
setScore	Public	Can Modify State	onlyAdmin modelExists
upgradeModelVersion	Public	Can Modify State	onlyAdmin modelExists
setRST	Public	Can Modify State	onlyAdmin modelExists
getBasicInfo	Public	-	modelExists
getTechnicalInfo	Public	-	modelExists
getModelData	Public	-	modelExists
getCompleteModel	Public	-	modelExists
getAllModelIds	Public	-	-
getOwnerModelIds	Public	-	-
getMultipleBasicInfo	Public	-	-
getModelOwner	Public	-	modelExists
getGrFile	Public	-	modelExists

ModelAssetContract			
getName	Public	-	modelExists

EurexaMarketplace			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
addAdmin	External	Can Modify State	onlyOwner
removeAdmin	External	Can Modify State	onlyOwner
updateFeePercentage	External	Can Modify State	onlyAdmin
withdrawFees	External	Can Modify State	onlyAdmin
createListing	External	Can Modify State	nonReentrant
cancelListing	External	Can Modify State	validListing nonReentrant
purchaseListing	External	Payable	validListing nonReentrant
updateListing	External	Can Modify State	validListing nonReentrant
getActiveModelListings	External	-	-
getUserListings	External	-	-
getListingDetails	External	-	validListing
calculatePurchaseCost	External	-	validListing
<Receive Ether>	External	Payable	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing events access control

Category: Malicious Event Log Audit

Content

The `transferAdmin` function in the Eurexa_coin contract fails to emit an event when admin privileges are transferred to a new address. This lack of transparency violates best practices for access control changes and prevents off-chain monitoring systems from tracking critical administrative changes.

- Eurexa_coin.sol

```
function transferAdmin(address newAdmin)
    external
    onlyAdmin
{
    require(newAdmin != address(0), "New admin cannot be zero address");
    admin = newAdmin;
}
```

Solution

Implement event emission for admin transfers by adding a dedicated event and emitting it when the admin address changes.

Status

Acknowledged

[N2] [Medium] Non-standard ERC-20 implementation in Eurexa_coin contract

Category: Others

Content

The Eurexa_coin contract implements a token system that significantly deviates from the ERC-20 standard, while still including elements that suggest ERC-20 compatibility. This non-standard implementation creates compatibility issues with existing DeFi protocols, wallets, and other infrastructure designed to work with ERC-20 tokens.

Solution

Implement a standard ERC-20 contract.

Status

Acknowledged; This is a new asset protocol developed by the project party. Exchanges or other users interacting with the protocol need to actively ensure compatibility with the token protocol.

[N3] [Suggestion] Missing zero address validation

Category: Design Logic Audit

Content

The addAdmin function in the ModelAssetContract has no validation against zero address input, which could lead to inadvertently granting admin privileges to the zero address (0x0).

- ModelAssetContract.sol

```
function addAdmin(address admin) public onlyAdmin {
    _admins[admin] = true;
}
```

Solution

Implement a zero address check.

Status

Acknowledged

[N4] [Suggestion] Improper admin removal in ModelAssetContract

Category: Gas Optimization Audit

Content

The removeAdmin function in ModelAssetContract incorrectly sets the admin status to false rather than completely removing the admin entry from the mapping. This approach is inefficient for gas usage and could potentially lead to confusion or security issues if the contract logic is modified in the future.

- ModelAssetContract.sol

```
function removeAdmin(address admin) public onlyAdmin {
    _admins[admin] = false;
}
```

Solution

Using delete frees up storage space and provides a gas refund.

```
delete _admins[admin];
```

Status

Acknowledged

[N5] [Medium] Excessive admin privileges across protocol contracts

Category: Authority Control Vulnerability Audit

Content

Analysis of the EureXa protocol contracts reveals a concerning pattern of excessive administrative privileges with insufficient access controls and privilege separation. Admin accounts have near-absolute control over critical contract functions, with inadequate checks and balances to prevent misuse or potential centralization issues.

1.ModelAssetContract.sol

- Admin accounts can add/remove other admins without restrictions
- Admins can create models and assign ownership arbitrarily
- Admins can unilaterally modify all model data
- No multi-signature requirements for sensitive operations

```
function addAdmin(address admin) public onlyAdmin {
    _admins[admin] = true;
}
```

2.Eurexa_coin.sol

- Single admin account holds complete control over tokenomics parameters
- Admin can withdraw all ETH from the contract without limits or delays

```
function withdrawEth(address payable to) external onlyAdmin returns (uint256) {
    // No withdrawal limits, time-locks, or multi-sig requirements
    uint256 amount = address(this).balance;
    (bool success, ) = to.call{value: amount}("");
    require(success, "ETH transfer failed");
    emit EthWithdrawn(to, amount);
    return amount;
}
```

3.Market.sol

- Owner/admin can modify fee structures affecting all marketplace transactions
- Admins can withdraw all marketplace fees without limits
- Owner can add/remove admins without restrictions
- Admins can cancel any user's listing arbitrarily

```
function withdrawFees(address payable to) external onlyAdmin {
    // No withdrawal limits or cooldowns
    uint256 balance = address(this).balance;
    (bool success, ) = to.call{value: balance}("");
    require(success, "Transfer failed");
    emit FeesWithdrawn(to, balance);
}
```

Solution

In the short term, transferring admin ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. And the authority involving user funds should be managed by the community, and the EOA address can manage the authority involving emergency contract suspension. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged; The project party will use multi-signature technology in the mainnet to reduce risks.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002503140001	SlowMist Security Team	2025.03.12 - 2025.03.14	Medium Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 2 low risk, 2 suggestion vulnerabilities.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist