SALUS SECURITY

NOV 2024

CODE SECURITY ASSESSMENT

TADLE

Overview

Project Summary

- Name: Tadle - Market Evm
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/tadle-com/market-evm
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Tadle - Market Evm
Version	v4
Type	Solidity
Dates	Nov 01 2024
Logs	Sep 12 2024; Sep 18 2024; Sep 24 2024; Nov 01 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	1
Total informational issues	2
Total	4

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk	Medium	Centralization	Mitigated
2	ReferralCode mechanism is imperfect	Low	Business Logic	Resolved
3	Error message error	Informational	Code Quality	Acknowledged
4	Missing two-step transfer ownership pattern	Informational	Business Logic	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk	
Severity: Medium	Category: Centralization
Target: - src/core/SystemConfigs.sol	

Description

The `SystemConfig` contract exists for the owner's account. When the status of the market is Online, the owner can change the configuration in the market, including the market token address, the market score, and the market cutoff time. Changes to these configurations may affect pending orders that already exist in the market.

If the private key of the owner's address is compromised, an attacker could profit from the buyer by modifying the market token address and score.

If the privileged account is a regular EOA account, this could be a concern and pose a risk to other users.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. Modifying key configurations should be prohibited when the market is not empty.

Status

The team has transferred owner to multi-sig account 0x8aa2...a007.

2. ReferralCode mechanism is imperfect

Severity: Low	Category: Business Logic
Target: - src/core/SystemConfig.sol	

Description

The ReferralCode mechanism allows for the creation and removal of referral codes. However, relying solely on the _referralCode parameter as a hash can leave it vulnerable to exploitation by malicious actors.

Attach Scenario 1:
Block the creation of `_referralCode`**.** Alice wants to create a `_referralCode` like `123`. The attacker can monitor the mempool and front-running the transaction by using MEV with the same `_referralCode`. Then Alice's transaction will revert.

Attach Scenario 2:
Earn tax from the removed `_referralCode` that has already been used and Not the first creator**.** Alice created a `_referralCode` and recommended many users. Then Alice calls `removeReferralCode()` to remove this `_referralCode`. The attacker can monitor the mempool and call `createReferralCode()` immediately to inherit the `_referralCode`. After those operate, the attacker can continue to receive benefits from the `_referralCode`.

src/core/SystemConfig.sol:L42 - L98

```
function createReferralCode(
    string calldata _referralCode,
    uint256 _referrerRate,
    uint256 _refereeRate
) external whenNotPaused {
    if (_referrerRate < baseReferralRate) {
        revert InvalidReferrerRate(_referrerRate);
    }

    uint256 referralExtraRate = referralExtraRateMap[msg.sender];
    uint256 totalRate = baseReferralRate + referralExtraRate;

    if (totalRate > Constants.REFERRAL_RATE_DECIMAL_SCALER) {
        revert InvalidTotalRate(totalRate);
    }

    if (_referrerRate + _refereeRate != totalRate) {
        revert InvalidRate(_referrerRate, _refereeRate, totalRate);
    }

    bytes32 referralCodeId = keccak256(abi.encode(_referralCode));
    if (referralCodeMap[referralCodeId].referrer != address(0x0)) {
        revert ReferralCodeExist(_referralCode);
    }
```

```
    referralCodeMap[referralCodeId] = ReferralInfo(
        msg.sender,
        _referrerRate,
        _refereeRate
    );

    emit CreateReferralCode(
        msg.sender,
        _referralCode,
        _referrerRate,
        _refereeRate
    );
}

function removeReferralCode(
    string calldata _referralCode
) external whenNotPaused {
    bytes32 referralCodeId = keccak256(abi.encode(_referralCode));

    if (referralCodeMap[referralCodeId].referrer != msg.sender) {
        revert Errors.Unauthorized();
    }

    delete referralCodeMap[referralCodeId];

    emit RemoveReferralCode(msg.sender, _referralCode);
}
```

Recommendation

Let the `_referralCode` and `msg.sender` be strongly bound.

Status

This issue has been resolved by the team with commit 4aa94f9.

2.3 Informational Findings

3. Error message error

Severity: Informational	Category: Code Quality
Target: - src/core/TokenManager.sol	

Description

Errors are used incorrectly, the error named `NotEnoughMsgValue` should not be used here.

```
function deposit(
    address _accountAddress,
    address _tokenAddress,
    uint256 _amount,
    bool _isPointToken
)
    external
    payable
    nonReentrant
    onlyRelatedContracts(tadleFactory, msg.sender)
    onlyInTokenWhitelist(_isPointToken, _tokenAddress)
{
    ...
    } else {
        if (msg.value != 0) {
            revert Errors.NotEnoughMsgValue(msg.value, 0);
        }
        /// @notice token is ERC20 token
        _transfer(
            _tokenAddress,
            _accountAddress,
            capitalPoolAddr,
            _amount,
            capitalPoolAddr
        );
    }

    emit Deposit(_accountAddress, _tokenAddress, _amount, _isPointToken);
}
```

Recommendation

Suggest adding a proper error like MsgValueCanNotAccept().

Status

This issue has been acknowledged by the team.

4. Missing two-step transfer ownership pattern

Severity: Informational	Category: Business logic
Target: - src/utils/Rescuable.sol	

Description

The `Rescuable` contract inherits from the Ownable contract. This contract does not implement a two-step process for transferring ownership. Thus, ownership of the contract can easily be lost when making a mistake in transferring ownership.

Recommendation

Consider using the Ownable2Step contract from OpenZeppelin instead.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 4444c21:

File	SHA-1 hash
./src/core/CapitalPool.sol	7a1541593ef12d55d8c6ab6e099cd351bc9aec4f
./src/core/DeliveryPlace.sol	470a47e2199d92a52904c2027e6771c1e7deb542
./src/core/PreMarkets.sol	acc5290b01efd2c55524d48e06e53323a490b517
./src/core/SystemConfig.sol	7cfdba7d6617e713a833e53dcef5eec416ce8d9f
./src/core/TokenManager.sol	a7d4bb3f337aa8a0e3a141053aa00e00fea6dcdf
./src/factory/ITadleFactory.sol	6bc51ff294d281cc747bcfafb6632b650bfc8e0a
./src/factory/TadleFactory.sol	bd6e92dc81b7a6c91cc00ec93bdd089bc0b56c4e
./src/proxy/UpgradeableProxy.sol	19b084df42dc209888ecef6ceaa91f2159d69fa3
./src/interfaces/ISystemConfig.sol	2761a421a17b7920b85abff83ddc24d01ba0a841
./src/interfaces/IPreMarkets.sol	7f4394f6c59783957a9c728a14f8983d34c86ed3
./src/interfaces/ITokenManager.sol	53533e4860f0e76bfa006596212610bfbe521387
./src/interfaces/IWrappedNativeToken.sol	2dae510ae8adf9620334ec5112bfc40ba6adb1f8
./src/interfaces/IDeliveryPlace.sol	6b739996c74f747714d7b892eae608f74e0f2894
./src/interfaces/ICapitalPool.sol	82570b88725d9c18b560fd84d5dc2fc611bc40a7
./src/libraries/OfferLibraries.sol	f442d8526cb954d320e2c3198692a050dfd18224
./src/libraries/Address.sol	dda59a51306dae8c854ae53351c2a612a66f5061
./src/libraries/MarketPlaceLibraries.sol	683dd66701fdf5f9a41ef36bad58dac5e7c5cd85
./src/libraries/Constants.sol	81617149d279954b3dd146f543748d0904242cee
./src/libraries/GenerateAddress.sol	4514cc561d8fee63c8b595e8144b8d1fe5c3fe81
./src/libraries/RelatedContractLibraries.sol	28b9b0ca4e77690608f95b66494aa587e132f24a
./src/utils/Errors.sol	883e7f39eba9e49b5c8347b2af94a004ae90ccec
./src/utils/Related.sol	7b6c460f97ba517be7b54ae4571848a104b84719
./src/utils/Rescuable.sol	db3d18ab3bb7b9c7146e429d42c716a196800f7f

./src/storage/SystemConfigStorage.sol	9c2d5a89dbffb072f754c66c6d54ae803565c722
./src/storage/UpgradeableStorage.sol	5b6df8cbf899a35b13d80f19ec3994d0466d44ca
./src/storage/TokenManagerStorage.sol	3eb1da19cf43e64c3b4e93f907ba707dabea4ef9
./src/storage/PreMarketsStorage.sol	4b3a48becae078c482bdb310697cd3997cf66826
./src/storage/OfferStatus.sol	8a2f5295b46a4804ad7baa79954790fbb1ec1a29
./src/storage/CapitalPoolStorage.sol	a946e721ecbff537bb2b32b9a14a87f0cfaacee9
./src/storage/DeliveryPlaceStorage.sol	8e6a8ea60b7ea73384bc70dc79b68dd92bab17af