BLOCKHAT
SECURITY

Clixpesa

Smart Contract Security Audit

Prepared by BlockHat
April 9th, 2025 - April 23rd 2025
BlockHat.io
contact@blockhat.io

Document Properties

Client	Clixpesa Solutions Ltd
Version	1.3.0
Classification	Public

Scope

Repository	Commit Hash
https://github.com/clixpesa/mint-contracts/tree/6c02023	6c02023bb8fffd29d751e06611784eaed0977d99

Re-Audit Files

Repository	Commit Hash
https://github.com/clixpesa/mint-contracts/tree/5202ee07	5202ee07dd3e928817ab4985fc5ce008fd2fcb2b

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1. Introduction **3**

1.1 About Clixpesa 3

1.2 Approach and Methodology 4

1.2.1 Risk Methodology 4

1.2.2 Severity Classification 4

2. Findings Overview **5**

2.1 Summary 5

2.2 Key Findings Summary 5

3. Detailed Findings **7**

3.1 Critical Findings 7

[C-01] Price Manipulation Vulnerability 7

3.2 High Severity Findings 8

[H-01] Missing Access Control for Critical Functions 8

[H-02] Reentrancy Risk in External Calls 9

[H-03] Business Logic Flaw in Repayment Handling 10

[H-04] Missing Access Control in repayLoan 12

[H-05] Loan Disbursement Fails to Check Token Return Value 13

3.3 Medium Severity Findings 14

[M-01] Unhandled Token Edge Cases 14

[M-02] Lack of Events for Critical State Changes 15

[M-03] Incomplete Debt Management System 16

[M-04] Fixed-Point Arithmetic Inconsistencies 17

[M-05] Reentrancy Risk in approveLoan 18

[M-06] Missing Events for Critical Functions 19

[M-07] Status Not Reset in requestLoan 19

[M-08] Missing Validation in topUpRoscaLoanPool 20

[M-09] Incorrect Balance Check During Loan Approval 21

3.4 Low Severity Findings 22

[L-01] Missing Input Validation 22
[L-02] Commented Out Code 23
[L-03] Lack of Contract Pause Mechanism 23
[L-04] Hardcoded Fee Values 24
[L-05] Missing NatSpec Documentation 25
[L-06] Boolean Comparison Gas Optimization 26
[L-07] Inefficient Array Usage in _addressExists 26
[L-08] Inconsistent Error Handling 27
[L-09] Missing Input Validation 28
[L-10] Redundant Role Assignments 29
[L-11] Unlimited Array Length Operations 30
3.5 Informational Severity Findings 30
[I-01] Gas Optimization Opportunities 30
[I-02] Code Structure and Separation of Concerns 32
[I-03] Potential Code Duplication 33
[I-04] Testing Coverage 33
[I-05] Documentation Comments Missing 34
[I-06] Gas Optimizations for Storage Access 35
[I-07] Code Structure Improvements 35
4. StaticAnalysis(Slither) 36
5. Conclusion 83

1. Introduction

Clixpesa Solutions Ltd engaged BlockHat to conduct a security assessment on Clixpesa's smart contracts beginning on April 9th, 2025 and ending April 23rd, 2025. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns. This document summarizes the findings of our audit.

1.1 About Clixpesa

Clixpesa Roscas: With Rotating Savings & Credit Associations (RoSCAs) users can come together as a group to help each other stay financially resilient. Users contribute to a pot, and the target amount goes to one of the users in a particular order until everyone has received a pot and the cycle starts over. This utility commonly known in Kenya as Chamas, helps many raise funds for otherwise big financial goals such as business capital or bills. Within the RoSCAs members can also ask for financial support for financial needs outside of the pot allocations. Users can create a RoSCA easily by inviting their friends through their phone numbers. Once the RoSCAs is created they can select their admins and around can be started. Funds disbursement happens automatically once a pot deadline is reached. Signatories to the RoSCA funds are randomised by the platform in order to give all members equal control over their funds.
Clixpesa Overdrafts: Here users are able to overdraw their transactions and spend more than they have. Once they receive any of the supported funds, the funds are deducted and used to repay any outstanding overdrafts.

1.2 Approach and Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

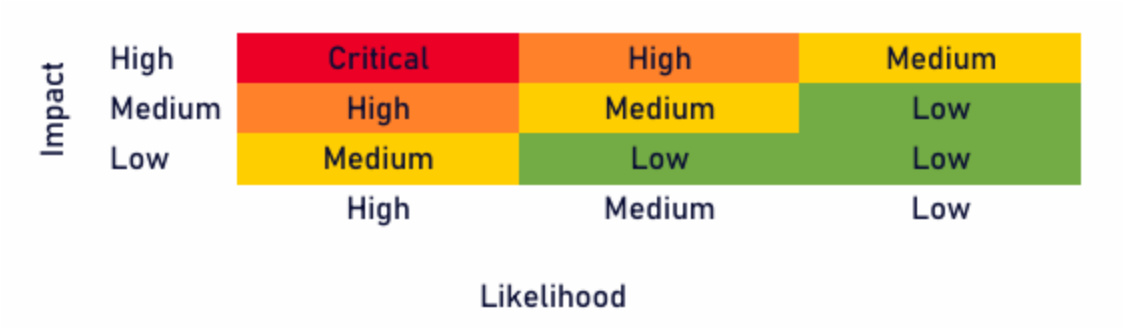
1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.
Its quantitative paradigm enables repeatable and precise measurement, while also revealing the

underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to1, with 5 indicating the greatest possibility or impact. Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed. Impact quantifies the technical and economic costs of successful attack. Severity indicates the risk's overall criticality. Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low,respectively. Severity is determined by probability and impact and is categorized into five levels,namely Critical, High, Medium, Low and Informational.

1.2.2 Severity Classification

Severity Level	**Description**
Critical	Issues that can lead to significant loss of funds, compromise of contract integrity, or complete contract failure
High	Issues that can lead to loss of funds, compromise of contract security, or severe functional problems
Medium	Issues that don't pose an immediate risk but should be addressed to improve security
Low	Issues that don't pose a significant risk but represent best practice improvements
Informational	Code style suggestions, gas optimizations, and general recommendations



2. Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the Clixpesa implementation. During the first part of our audit, we examine the smart contract source code and run the code base via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes,and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings Summary

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include 1 critical-severity, 5 high-severity, 9 medium-severity, 11 low-severity, 7 informational-severity vulnerabilities.

ID	Contract	Vulnerabilities	Severity	Status
C-01	Overdraft	Price Manipulation Vulnerability	CRITICAL	Acknowledged
H-01	Overdraft	Missing Access Control for Critical Functions	HIGH	Fixed
H-02	Overdraft	Reentrancy Risk in External Calls	HIGH	Fixed
H-03	Overdraft	Business Logic Flaw in Repayment Handling	HIGH	Acknowledged
H-04	Roscas	Missing Access Control in repayLoan	HIGH	Fixed
H-05	Roscas	Loan Disbursement Fails to Check Token Return Value	HIGH	Fixed
M-01	Overdraft	Unhandled Token Edge Cases	MEDIUM	Acknowledged
M-02	Overdraft	Lack of Events for Critical State Changes	MEDIUM	Fixed
M-03	Overdraft	Incomplete Debt Management System	MEDIUM	Acknowledged
M-04	Overdraft	Fixed-Point Arithmetic Inconsistencies	MEDIUM	Acknowledged
M-05	Roscas	Reentrancy Risk in approveLoan	MEDIUM	Acknowledged

ID	Contract	Vulnerabilities	Severity	Status
M-06	Roscas	Missing Events for Critical Functions	MEDIUM	Fixed
M-07	Roscas	Status Not Reset in requestLoan	MEDIUM	Fixed
M-08	Roscas	Missing Validation in topUpRoscaLoanPool	MEDIUM	Acknowledged
M-09	Roscas	Incorrect Balance Check During Loan Approval	MEDIUM	Fixed
L-01	Overdraft	Missing Input Validation	LOW	Acknowledged
L-02	Overdraft	Commented Out Code	LOW	Acknowledged
L-03	Overdraft	Lack of Contract Pause Mechanism	LOW	Acknowledged
L-04	Overdraft	Hardcoded Fee Values	LOW	Acknowledged
L-05	Overdraft	Missing NatSpec Documentation	LOW	Acknowledged
L-06	Roscas	Boolean Comparison Gas Optimization	LOW	Acknowledged
L-07	Roscas	Inefficient Array Usage in _addressExists	LOW	Acknowledged
L-08	Roscas	Inconsistent Error Handling	LOW	Acknowledged
L-09	Roscas	Missing Input Validation	LOW	Acknowledged
L-10	Roscas	Redundant Role Assignments	LOW	Acknowledged
L-11	Roscas	Unlimited Array Length Operations	LOW	Acknowledged
I-01	Overdraft	Gas Optimization Opportunities	INFO	Acknowledged
I-02	Overdraft	Code Structure and Separation of Concerns	INFO	Acknowledged
I-03	Overdraft	Potential Code Duplication	INFO	Acknowledged
I-04	Overdraft	Testing Coverage	INFO	Acknowledged
I-05	Roscas	Documentation Comments Missing	INFO	Acknowledged
I-06	Roscas	Gas Optimizations for Storage Access	INFO	Acknowledged

ID	Contract	Vulnerabilities	Severity	Status
I-07	Roscas	Code Structure Improvements	INFO	Acknowledged

3. Detailed Findings

3.1 Critical Findings

[C-01] Price Manipulation Vulnerability

Contract: ClixpesaOverdraft
Severity: **CRITICAL**
Status: Acknowledge

Description:
The contract relies on Uniswap V3 spot prices without any time-weighted average or manipulation resistance. This exposes the contract to flash loan attacks where an attacker could manipulate the price in a UniswapV3 pool before calling contract functions that rely on these price feeds.

Impact:
An attacker could manipulate prices to extract more value from the protocol than entitled. For example, they could artificially depress the price of a token to borrow more value than they should be able to access.

Likelihood – 3 Impact - 5

Recommendation:
Implement price manipulation resistance:

- Use time-weighted average prices (TWAP) with a suitable period
- Consider integrating Chainlink price feeds as a primary or secondary oracle
- Implement circuit breakers for extreme price movements

```
// Example implementation of TWAP
function _getRate(address uniswapPool) internal view returns (uint256 rate) {
    IUniswapV3Pool localUSDPool = IUniswapV3Pool(uniswapPool);
    uint32 twapInterval = 3600; // 1 hour TWAP
```

```
    uint32[] memory secondsAgos = new uint32[](2);
    secondsAgos[0] = twapInterval; // from 1 hour ago
    secondsAgos[1] = 0; // to now
    (int56[] memory tickCumulatives, ) = localUSDPool.observe(secondsAgos);
    int56 tickCumulativesDelta = tickCumulatives[1] - tickCumulatives[0];
    int24 timeWeightedAverageTick = int24(tickCumulativesDelta / twapInterval);
    // Convert tick to price
    uint160 sqrtPriceX96 = TickMath.getSqrtRatioAtTick(timeWeightedAverageTick);
    uint256 price = FullMath.mulDiv(
        uint256(sqrtPriceX96) * S_FACTOR,
        uint256(sqrtPriceX96),
        FixedPoint96.Q96 * S_FACTOR
    );
    return (price * S_FACTOR) / FixedPoint96.Q96;
}
```

3.2 High Severity Findings

[H-01] Missing Access Control for Critical Functions

Contract: ClixpesaOverdraft
Severity: HIGH
Status: Fixed

Description:
The `useOverdraft()` and `repayOverdraft()` functions lack proper access control. Any address can call these functions on behalf of any user. This creates a security vulnerability where malicious actors could take unauthorized actions on behalf of users.

Impact:
Malicious actors could force unwanted overdrafts on users or make unauthorized repayments, potentially leading to loss of funds or manipulation of user debt records.

Recommendation:
Implement proper access control for these functions. Either:

1. Require that the caller is the user themselves (using `msg.sender` instead of a parameter):

```
function useOverdraft(address token, uint256 amount) external {
    User storage user = users[msg.sender];
    // Rest of the function using msg.sender instead of userAddress
}
```

2. Or if delegate functionality is required, implement an approval mechanism:

```
mapping(address => mapping(address => bool)) private approvedDelegates;
function approveDelegate(address delegate, bool approval) external {
    approvedDelegates[msg.sender][delegate] = approval;
}
function useOverdraft(address userAddress, address token, uint256 amount)
external {
    if (msg.sender != userAddress &&
!approvedDelegates[userAddress][msg.sender])
        revert Unauthorized();
    // Rest of the function
}
```

[H-02] Reentrancy Risk in External Calls

Contract: ClixpesaOverdraft
Severity: HIGH
Status: Fixed

Description:
Despite using the ReentrancyGuardUpgradeable, the `useOverdraft()` function performs state changes after external calls to `IERC20.transfer()`. The function modifies user data after the transfer, creating a reentrancy vulnerability.

Impact:
If the token is malicious or compromised, it could reenter the contract during the transfer and exploit the contract state before the user data is updated.

Recommendation:
Follow the checks-effects-interactions pattern by updating the contract state before making external calls:

```
function useOverdraft(address userAddress, address token, uint256 amount)
external nonReentrant {
    // Validation checks...
    bytes6 id = GenerateId.withAddressNCounter(userAddress, ++idCounter);
    uint256 requestedAt = block.timestamp;
    // Create overdraft record
    Overdraft memory overdraft = Overdraft({
        token: IERC20(token),
```

```
        userAddress: payable(userAddress),
        tokenAmount: amount,
        baseAmount: baseAmount,
        takenAt: requestedAt
    });
    // Update state BEFORE external calls
   overdrafts[id] = overdraft;
    user.availableLimit = user.availableLimit - baseAmount;
    user.overdraftIds.push(id);
    // Update debt information
    user.overdraftDebt = OverdraftDebt({...});
    users[userAddress] = user;
    // External call AFTER state changes
   require(IERC20(token).transfer(userAddress, amount), "Transfer failed");
    emit OverdraftUsed(userAddress, baseAmount, token, amount);
}
```

[H-03] Business Logic Flaw in Repayment Handling

Contract: ClixpesaOverdraft
Severity: HIGH
Status: Acknowledged

Description:
The `repayOverdraft()` function has a logic issue where it checks the user's token balance but then performs a `transferFrom`. This is problematic because:

1. It should check allowance instead of balance for `transferFrom`
2. The balance check is insufficient since the user's balance might change between the check and the actual transfer

Impact:
This could lead to failed transactions, user frustration, and potential for manipulation by advanced users.

Recommendation:
Update the validation logic to check allowance instead of balance, and rely on the transfer reversion rather than pre-checking:

```
function repayOverdraft(
    address userAddress,
```

```
    address token,
    uint256 amount
) external {
    if (supportedTokens[0] != token && supportedTokens[1] != token)
        revert OD_InvalidToken();
    if (amount == 0) revert OD_MustMoreBeThanZero();
    // Remove incorrect balance check
    // if (IERC20(token).balanceOf(userAddress) < amount) revert
OD_InsufficientBalance();
    User storage user = users[userAddress];
    if (user.overdraftDebt.amountDue == 0) revert OD_NoOverdarftDebt();
    uint256 baseAmount = _getBaseAmount(amount, token);
    if (baseAmount > user.overdraftDebt.amountDue) {
        uint256 tokenAmount = _getTokenAmount(
            user.overdraftDebt.amountDue,
            token
        );
        // The transferFrom will revert if balance or allowance is
insufficient
        require(
            IERC20(token).transferFrom(userAddress, address(this),
tokenAmount),
            "Repayment Failed"
        );
        // Continue with debt update...
    } else {
        // The transferFrom will revert if balance or allowance is
insufficient
        require(
            IERC20(token).transferFrom(userAddress, address(this), amount),
            "Repayment Failed"
        );
        // Continue with debt update...
    }
    // Rest of the function...
}
```

[H-04] Missing Access Control in repayLoan

Contract: ClixpesaRoscas
Severity: HIGH
Status: Fixed

Description:
The `repayLoan` function allows anyone to repay a loan on behalf of a borrower, but doesn't adequately validate that the repayment is being made by or on behalf of the correct borrower. The function accepts a borrower address parameter but doesn't validate the relationship between `msg.sender` and the borrower.

```
function repayLoan(
    uint256 _requestId,
    uint256 _amount,
    uint256 _roscaId,
    address borrower
) public screening nonReentrant {
    if (_roscaId != 0) {
        if (!hasRole(ROSCA_MEMBER_ROLE, borrower)) revert NotRoscaMember();
    } else {
        if (!hasRole(MEMBER_ROLE, borrower)) revert NotRegistered();
    }
    Loan storage loan = loans[borrower][_requestId];
    if (loan.status != Status.Active) revert NotActive();
    IERC20(loan.token).transferFrom(msg.sender, address(this), _amount);
    // ... remainder of function
}
```

Impact:
An attacker could potentially manipulate the repayment process by repaying loans for others, which could lead to unexpected behavior or conflicts in the loan management system.

Recommendation:
Add proper validation to ensure that only the borrower, an authorized representative, or an admin can repay a loan. Consider adding a check that either `msg.sender` is the borrower or has the appropriate admin role:

```
require(msg.sender == borrower || hasRole(ADMIN_ROLE, msg.sender) ||
        (hasRole(ROSCA_ADMIN_ROLE, msg.sender) && userOnRosca[msg.sender]
== _roscaId),
```

```
        "Not authorized to repay this loan");
```

[H-05] Loan Disbursement Fails to Check Token Return Value

Contract: ClixpesaRoscas
Severity: HIGH
Status: Fixed

Description:
In the `approveLoan` function, the contract uses the `transfer` method of the ERC20 token without checking the return value. Some ERC20 tokens don't revert on failure but return false, which could lead to a situation where the loan is marked as approved but no funds are actually transferred.

```
IERC20(loanRequest.token).transfer(_member, loanRequest.requestedAmount);
```

Impact:
This could result in financial loss where loans are recorded as disbursed, but no actual tokens are transferred to the borrower.

Recommendation:
Use the SafeERC20 library's `safeTransfer` function which handles the return value check properly:

```
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

// ...
using SafeERC20 for IERC20;
// ...
IERC20(loanRequest.token).safeTransfer(_member,
loanRequest.requestedAmount);
```

3.3 Medium Severity Findings

[M-01] Unhandled Token Edge Cases

Contract: ClixpesaOverdraft
Severity: MEDIUM
Status: Acknowledged

Description:
The contract doesn't handle edge cases for ERC20 tokens:

1. It doesn't handle fee-on-transfer tokens
2. It assumes all tokens have 18 decimals
3. It doesn't handle rebasing tokens properly

Impact:
These edge cases could lead to accounting errors, potential fund loss, and incorrect debt calculations.

Recommendation:

1. Add specific token validation in the initialization:

```
function initialize(...) public initializer {
    // Existing code...
    // Validate token decimals
    for (uint i = 0; i < _supportedTokens.length; i++) {
        uint8 decimals = IERC20Metadata(_supportedTokens[i]).decimals();
        require(decimals == 18, "Token must have 18 decimals");
    }
}
```

2. Consider adding a whitelist mechanism where only vetted tokens can be used
3. Track actual amounts transferred rather than assuming the requested amount:

```
function useOverdraft(address userAddress, address token, uint256 amount)
external {
    // Existing validations...
    uint256 balanceBefore = IERC20(token).balanceOf(userAddress);
    require(IERC20(token).transfer(userAddress, amount), "Transfer
failed");
    uint256 balanceAfter = IERC20(token).balanceOf(userAddress);
    uint256 actualTransferred = balanceAfter - balanceBefore;
     // Use actualTransferred for calculations
    // ...
}
```

[M-02] Lack of Events for Critical State Changes

Contract: ClixpesaOverdraft
Severity: MEDIUM
Status: Fixed

Description:
The contract is missing events for critical state changes, particularly when debt states change from Good to Grace or Grace to Defaulted. This makes it difficult to track important contract state changes off-chain.

Impact:
Reduced transparency and difficulty in building reliable monitoring systems around the protocol, potentially leading to missed critical state changes.

Recommendation:
Add events for all important state changes:

```
// Add new events
event OverdraftStateChanged(address indexed user, Status oldState, Status
newState, uint256 amountDue);
event UserLimitUpdated(address indexed user, uint256 oldLimit, uint256
newLimit);
// Update functions to emit events when states change
function updateUserDebt(address userAddress) external {
    User storage user = users[userAddress];
    // Existing code...
    Status oldState = user.overdraftDebt.state;
    // Determine new state based on time and conditions
    Status newState = _determineNewState(user);
    if (oldState != newState) {
        user.overdraftDebt.state = newState;
        emit OverdraftStateChanged(userAddress, oldState, newState,
user.overdraftDebt.amountDue);
    }
}
```

[M-03] Incomplete Debt Management System

Contract: ClixpesaOverdraft
Severity: MEDIUM
Status: Open

Description:
The debt management system is incomplete. The status transitions (Good → Grace → Defaulted) are not fully implemented, and there's no actual handling of overdue debts or penalties.

Impact:
The current implementation doesn't provide a complete lending protocol, and grace periods/defaults are not properly enforced, potentially leading to protocol insolvency.

Recommendation:
Implement a complete debt lifecycle:

```
function updateUserDebt(address userAddress) external {
    User storage user = users[userAddress];
    if (user.overdraftDebt.amountDue == 0) revert OD_MustMoreBeThanZero();
    if (user.overdraftDebt.lastChecked + 1 days - 1 > block.timestamp)
        revert OD_CheckedEarly();
    // Calculate time-based status
    Status oldState = user.overdraftDebt.state;
    Status newState = oldState;
    // Determine status based on due time
    if (block.timestamp > user.overdraftDebt.dueTime) {
        if (block.timestamp > user.overdraftDebt.dueTime + 30 days) {
            // More than 30 days overdue - Defaulted
            newState = Status.Defaulted;
        } else {
            // Overdue but less than 30 days - Grace
            newState = Status.Grace;
        }
    }
    // Apply different fee structures based on status
    uint256 additionalFee = user.overdraftDebt.serviceFee;
    if (newState == Status.Grace) {
        // Apply penalty interest for Grace period
        additionalFee += (user.overdraftDebt.amountDue * 5e16) / S_FACTOR;
// 5% penalty
    } else if (newState == Status.Defaulted) {
        // Apply higher penalty for Defaulted status
        additionalFee += (user.overdraftDebt.amountDue * 1e17) / S_FACTOR;
// 10% penalty
    }
```

```
    user.overdraftDebt.amountDue += additionalFee;
    user.overdraftDebt.lastChecked = block.timestamp;
    if (oldState != newState) {
        user.overdraftDebt.state = newState;
        emit OverdraftStateChanged(
            userAddress,
            oldState,
            newState,
            user.overdraftDebt.amountDue
        );
    }
}
```

[M-04] Fixed-Point Arithmetic Inconsistencies

Contract: ClixpesaOverdraft
Severity: MEDIUM
Status: Open

Description:
The contract uses fixed-point arithmetic inconsistently across different functions. Sometimes it correctly applies the scaling factor `S_FACTOR`, but in other cases, it appears to use different scaling approaches.

Impact:
This inconsistency could lead to calculation errors, rounding issues, and potential accounting discrepancies.

Recommendation:
Standardize the fixed-point arithmetic approach:

1. Create utility functions for consistent fixed-point operations:

```
function _mulDiv(uint256 a, uint256 b, uint256 denominator) internal pure
returns (uint256) {
    return (a * b) / denominator;
}
function _applyPercentage(uint256 amount, uint256 percentage) internal pure
returns (uint256) {
    // percentage is expressed as a fixed-point number with S_FACTOR
precision
    return _mulDiv(amount, percentage, S_FACTOR);
```

2. Use these utility functions consistently throughout the contract for all calculations

[M-05] Reentrancy Risk in approveLoan

Contract: ClixpesaRoscas
Severity: MEDIUM
Status: Open

Description:
Although the `approveLoan` function is protected with the `nonReentrant` modifier, the order of operations could still allow a reentrancy attack. The contract updates the loan pool balance after making the external token transfer:

```
if (_roscaId != 0) {
    roscaLoanPools[loanRequest.roscaId] -= loanRequest.requestedAmount;
}
IERC20(loanRequest.token).transfer(_member, loanRequest.requestedAmount);
```

Impact:
If the token transfer triggers a callback to the contract, the loan pool balance might not yet be updated, potentially allowing multiple loans to be disbursed beyond the pool's intended capacity.

Recommendation:
Follow the checks-effects-interactions pattern by updating the state variables before making external calls:

```
if (_roscaId != 0) {
    roscaLoanPools[loanRequest.roscaId] -= loanRequest.requestedAmount;
}
// State changes should be complete before external interactions
IERC20(loanRequest.token).safeTransfer(_member,
loanRequest.requestedAmount);
```

[M-06] Missing Events for Critical Functions

Contract: ClixpesaRoscas
Severity: MEDIUM
Status: Fixed

Description:
Several critical functions in the contract don't emit events, making it difficult to track important

actions off-chain. For example, the `togglePause` function doesn't emit any event when the contract is paused or unpaused.

Impact:
Without proper events, it becomes difficult for users and external systems to monitor the state of the contract and react to changes in its operation.

Recommendation:
Add events for all state-changing functions, especially those that affect the contract's availability or user funds. For example:

```
event ContractPaused(bool isPaused);

function togglePause(bool _status) public onlyRole(ADMIN_ROLE) {
    if (paused == _status) revert NoChange();
    paused = !paused;
    emit ContractPaused(paused);
}
```

[M-07] Status Not Reset in requestLoan

Contract: ClixpesaRoscas
Severity: MEDIUM
Status: Fixed

Description:
In the `requestLoan` function, the `userLoanStatus` is set to true, but there's no mechanism to reset it if the loan request is later changed to a status other than "Requested" before being approved:

userLoanStatus[borrower] = true;

Impact:
If a loan request is rejected or expires without being approved, the user might be permanently prevented from requesting new loans.

Recommendation:
Ensure that `userLoanStatus` is appropriately reset when a loan request is rejected or changes status. Consider adding a function to reset the status:

```
function resetUserLoanStatus(address _borrower) internal {
    userLoanStatus[_borrower] = false;
}
```

Then call this function in the `rejectLoan` function and any other function that terminates a loan request.

[M-08] Missing Validation in topUpRoscaLoanPool

Contract: ClixpesaRoscas
Severity: MEDIUM
Status: Fixed

Description:
The `topUpRoscaLoanPool` function doesn't validate that the contract has actually received the tokens being added to the pool:

```
function topUpRoscaLoanPool(uint256 _roscaId, uint256 _amount) public
screening onlyRole(ADMIN_ROLE) {
    if (!roscas[_roscaId].isOpen) revert Closed();
    roscaLoanPools[_roscaId] += _amount;
}
```

Impact:
Without verifying that tokens have been transferred to the contract, the loan pool balance could be increased without the corresponding funds, leading to insufficient funds when loans are disbursed.

Recommendation:
Implement a proper transfer mechanism that ensures tokens are actually received by the contract before increasing the pool balance:

```
function topUpRoscaLoanPool(uint256 _roscaId, address _tokenAddress,
uint256 _amount) public screening onlyRole(ADMIN_ROLE) {
    if (!roscas[_roscaId].isOpen) revert Closed();
    IERC20 token = IERC20(_tokenAddress);
    token.safeTransferFrom(msg.sender, address(this), _amount);
    roscaLoanPools[_roscaId] += _amount;
}
```

[M-09] Incorrect Balance Check During Loan Approval

Contract: ClixpesaRoscas
Severity: MEDIUM
Status: Open

Description:
In the `checkFunds` function, the contract checks if there are enough tokens to cover the requested loan amount but doesn't account for tokens that might already be allocated to other purposes:

```
function checkFunds(LoanRequest memory loanRequest) internal view {
    // Confirm the Smart contract has enough funds
    uint256 tokenBalance =
IERC20(loanRequest.token).balanceOf(address(this));
    if (loanRequest.requestedAmount > tokenBalance) {
        revert InsufficientContractFunds();
    }
    // If rosca loan, confirm the rosca has enough funds
    if (loanRequest.roscaId != 0 && loanRequest.requestedAmount >
roscaLoanPools[loanRequest.roscaId]) {
        revert InsufficientRoscaFunds();
    }
}
```

Impact:
This could lead to situations where the contract appears to have enough tokens, but those tokens are already committed to other purposes, resulting in failed transfers or insufficient funds for other operations.

Recommendation:
Implement a proper accounting system to track the available balance of tokens for each specific purpose, separate from the total token balance of the contract.

3.4 Low Severity Findings

[L-01] Missing Input Validation

Contract: ClixpesaOverdraft
Severity: LOW
Status: Open

Description:
Several functions lack complete input validation:

- `subscribeUser()` doesn't validate that `initialLimit` is greater than zero
- Token addresses aren't validated to be actual contracts
- Uniswap pool addresses aren't validated

Impact:
Invalid inputs could cause unexpected behavior or transaction reverts.

Recommendation:
Add comprehensive input validation:

```
function initialize(
    address[] memory _supportedTokens,
    address[] memory _uniswapV3Pools,
    string memory _key
) public initializer {
    __Ownable_init(msg.sender);
   __UUPSUpgradeable_init();
    // Validate inputs
    require(_supportedTokens.length > 0, "Empty supported tokens");
    require(_uniswapV3Pools.length > 0, "Empty Uniswap pools");
    // Validate token contracts
    for (uint i = 0; i < _supportedTokens.length; i++) {
        require(_supportedTokens[i] != address(0), "Zero address");
        require(isContract(_supportedTokens[i]), "Not a contract");
    }
    // Validate Uniswap pools
    for (uint i = 0; i < _uniswapV3Pools.length; i++) {
        require(_uniswapV3Pools[i] != address(0), "Zero address");
        require(isContract(_uniswapV3Pools[i]), "Not a contract");
    }
    supportedTokens = _supportedTokens;
    uniswapPools = _uniswapV3Pools;
    subscriptionKey = keccak256(abi.encodePacked(_key));
}

// Helper function
function isContract(address addr) internal view returns (bool) {
    uint size;
    assembly { size := extcodesize(addr) }
    return size > 0;
}
```

[L-02] Commented Out Code

Contract: ClixpesaOverdraft
Severity: LOW
Status: Open

Description:
There is commented-out code in the contract, which indicates potential uncertainty in implementation and could lead to confusion.

Impact:
Reduced code clarity and potential confusion for developers.

Recommendation:
Remove commented code that's not needed:

```
// Remove this commented code
/*
constructor(address[] memory _supportedTokens, string memory _key) {
    supportedTokens = _supportedTokens;
    subscriptionKey = keccak256(abi.encodePacked(_key));
}*/
```

[L-03] Lack of Contract Pause Mechanism

Contract: ClixpesaOverdraft
Severity: LOW
Status: Open

Description:
The ClixpesaOverdraft contract lacks an emergency pause mechanism, which is a standard safety feature for financial contracts, especially those that are upgradeable. Having a pause mechanism is crucial to prevent further damage if vulnerabilities are discovered.

Impact:
In case of discovered vulnerabilities, the contract cannot be paused to prevent further damage, potentially leading to loss of funds or exploitation of vulnerabilities.

Recommendation:
Implement the OpenZeppelin Pausable contract or same as that done in Roscas

```
event ContractPaused(bool isPaused);

function togglePause(bool _status) public onlyRole(ADMIN_ROLE) {
    if (paused == _status) revert NoChange();
    paused = !paused;
    emit ContractPaused(paused);
}
```

[L-04] Hardcoded Fee Values

Contract: ClixpesaOverdraft
Severity: LOW
Status: Open

Description:
The contract uses hardcoded fee values in the `_getAccessFee()` and `_getServiceFee()` functions, which makes it inflexible if fee adjustments are needed in the future.

Impact:
Any fee change would require a contract upgrade, which is a costly and risky operation. This limits the flexibility of the protocol and could prevent administrators from quickly adapting to market conditions.

Recommendation:
Implement configurable fee parameters:

```
uint256 private accessFeePercentage; // stored as fixed-point, e.g., 0.01e18 for 1%
mapping(uint256 => uint256) private serviceFeesByTier; // amount tiers to fee amounts

function initialize(...) public initializer {
    // Existing code...
    // Set default fees
    accessFeePercentage = 0.01e18; // 1%
    // Set service fees by tier
    serviceFeesByTier[1e18] = 0; // <1 token
    serviceFeesByTier[5e18] = 0.2e17; // 1-5 tokens
    serviceFeesByTier[10e18] = 0.8e17; // 5-10 tokens
    // ...and so on
}
// Admin function to update fees
function updateAccessFee(uint256 newFeePercentage) external onlyOwner {
    accessFeePercentage = newFeePercentage;
    emit AccessFeeUpdated(newFeePercentage);
}
function updateServiceFee(uint256 tier, uint256 fee) external onlyOwner {
    serviceFeesByTier[tier] = fee;
    emit ServiceFeeUpdated(tier, fee);
}
```

[L-05] Missing NatSpec Documentation

Contract: ClixpesaOverdraft
Severity: LOW
Status: Open

Description:
The contract is missing comprehensive NatSpec documentation, which reduces code understandability and makes it harder for other developers to contribute or audit the code.

Impact:
Reduced code maintainability and increased chance of implementation errors during upgrades or when new developers work on the codebase.

Recommendation:
Add comprehensive NatSpec comments to all functions:

```
/**
 * @notice Allows a user to take an overdraft against their available limit
 * @dev Transfers tokens to the user and creates a debt record
 * @param userAddress The address of the user taking the overdraft
 * @param token The token address being withdrawn
 * @param amount The amount of tokens to withdraw
 * @return bytes6 The generated ID for this overdraft
 */
function useOverdraft(address userAddress, address token, uint256 amount)
external returns (bytes6) {
    // Function implementation...
}
```

[L-06] Boolean Comparison Gas Optimization

Contract: ClixpesaRoscas
Severity: LOW
Status: Open

Description:
Throughout the contract, there are comparisons with boolean literals that consume unnecessary gas:

```
if (paused == _status) revert NoChange();
```

Impact:
These comparisons result in slightly higher gas consumption than necessary, which could impact transaction costs, especially for functions that are called frequently.

Recommendation:
Optimize gas usage by directly using the boolean value:

```
if (paused == _status) revert NoChange();
// Change to:
if (paused != !_status) revert NoChange();
```

[L-07] Inefficient Array Usage in _addressExists

Contract: ClixpesaRoscas
Severity: LOW
Status: Open

Description:
The `_addressExists` function iterates through an array to check for an address, which is inefficient for large arrays:

```
function _addressExists(address[] memory addresses, address _address)
internal pure returns (bool) {
    for (uint256 i = 0; i < addresses.length; i++) {
        if (addresses[i] == _address) {
            return true;
        }
    }
    return false;
}
```

Impact:
This implementation has O(n) time complexity, which could lead to high gas costs or even out-of-gas errors when checking for membership in large arrays.

Recommendation:
Consider using a mapping to track addresses instead of arrays where possible, or use the

EnumerableSet library which already provides a contains function:

```
import "@openzeppelin/contracts/utils/structs/EnumerableSet.sol";

contract ClixpesaRoscas {
    using EnumerableSet for EnumerableSet.AddressSet;

    // Instead of address[] members
    EnumerableSet.AddressSet private members;

    // Add a member
    function addMember(address member) internal {
        members.add(member);
    }

    // Check if member exists
    function memberExists(address member) internal view returns (bool) {
        return members.contains(member);
    }
}
```

[L-08] Inconsistent Error Handling

Contract: ClixpesaRoscas
Severity: LOW
Status: Open

Description:
The contract uses both custom errors and require statements inconsistently, with some functions using `revert` with custom errors while others use `require`:

```
// Custom error
if (!roscas[_roscaId].isOpen) revert Closed();

// Require statement
require(!hasActiveRoscaLoanRequest[_roscaId], "Rosca already has an active
loan");
```

Impact:
This inconsistency makes the code less maintainable and readable. Additionally, `require` statements with string messages consume more gas than custom errors.

Recommendation:
Standardize error handling throughout the contract. Consider using custom errors consistently as they are more gas-efficient:

```
// Define custom error error RoscaHasActiveLoan();
// Use custom error consistently
if (hasActiveRoscaLoanRequest[_roscaId]) revert RoscaHasActiveLoan();
```

[L-09] Missing Input Validation

Contract: ClixpesaRoscas
Severity: LOW
Status: Open

Description:
Some functions don't validate their inputs adequately:

```
function sendTokens(address _tokenAddress, address _to, uint256 _amount)
external onlyRole(ADMIN_ROLE) {
    IERC20 token = IERC20(_tokenAddress);
    token.safeTransfer(_to, _amount);
}
```

There's no validation that `_to` is not the zero address or that `_amount` is greater than zero.

Impact:
Missing input validation could lead to unexpected behavior, failed transactions, or in some cases, loss of funds.

Recommendation:
Add proper validation for all function inputs:

```
function sendTokens(address _tokenAddress, address _to, uint256 _amount)
external onlyRole(ADMIN_ROLE) {
    require(_tokenAddress != address(0), "Invalid token address");
    require(_to != address(0), "Cannot send to zero address");
    require(_amount > 0, "Amount must be greater than zero");
    IERC20 token = IERC20(_tokenAddress);
    token.safeTransfer(_to, _amount);
}
```

[L-10] Redundant Role Assignments

Contract: ClixpesaRoscas
Severity: LOW
Status: Open

Description:
In the `initialize` function, roles are assigned redundantly:

```
_grantRole(DEFAULT_ADMIN_ROLE, defaultAdmin);
_grantRole(ADMIN_ROLE, defaultAdmin);
_grantRole(UPGRADER_ROLE, defaultAdmin);
```

The DEFAULT_ADMIN_ROLE already has all permissions, so explicitly granting ADMIN_ROLE and UPGRADER_ROLE might be redundant.

Impact:
Redundant role assignments increase gas costs during initialization and may create confusion about the role hierarchy.

Recommendation:
Review the role hierarchy and remove redundant role assignments where appropriate. If the DEFAULT_ADMIN_ROLE is intended to have all permissions, then additional role assignments are unnecessary:

```
function initialize(address defaultAdmin) public initializer {
    // Grant only the DEFAULT_ADMIN_ROLE
    _grantRole(DEFAULT_ADMIN_ROLE, defaultAdmin);
    // Other initialization code...
}
```

[L-11] Unlimited Array Length Operations

Contract: ClixpesaRoscas
Severity: LOW
Status: Open

Description: Several functions operate on arrays with potentially unbounded lengths:

```
function registerMembers(address[] calldata _members) external screening
onlyCAdminOrRAdmin {  }
```

This could lead to out-of-gas errors if the array is too large.

Impact:
Functions operating on unbounded arrays might fail due to exceeding the block gas limit, leading to partial execution or complete transaction failure.

Recommendation:
Add limits to array lengths or process arrays in batches to avoid exceeding gas limits:

```
function registerMembers(address[] calldata _members) external screening
onlyCAdminOrRAdmin {
    require(_members.length <= 100, "Too many members in a single call");
    // ...
}
```

3.5 Informational Severity Findings

[I-01] Gas Optimization Opportunities

Contract: ClixpesaOverdraft
Severity: Informational
Status: Open

Description:
Several gas optimizations are possible throughout the contract:

1. Use unchecked blocks for arithmetic that cannot overflow:

```
// For Solidity 0.8.x which has built-in overflow checking
unchecked {
    user.overdraftDebt.amountDue = user.overdraftDebt.amountDue +
baseAmount + accessFee;
}
```

2. Cache array length in for loops:

```
function getUserOverdrafts(address user) public view returns (Overdraft[]
memory) {
    User storage userData = users[user];
    bytes6[] storage ids = userData.overdraftIds;
```

```
    uint256 length = ids.length; // Cache length
    Overdraft[] memory results = new Overdraft[](length);
    for (uint256 i = 0; i < length; i++) {
        results[i] = overdrafts[ids[i]];
    }
    return results;
}
```

3. Use custom errors instead of revert strings:

```
// Instead of:
require(IERC20(token).transfer(userAddress, amount), "Transfer failed");
// Use:
error TransferFailed();
...
if (!IERC20(token).transfer(userAddress, amount)) revert TransferFailed();
```

Impact:
Implementing these optimizations could significantly reduce gas costs, especially for functions that are called frequently.

Recommendation:
Apply these gas optimization techniques throughout the contract to reduce transaction costs.

[I-02] Code Structure and Separation of Concerns

Contract: ClixpesaOverdraft
Severity: Informational
Status: Open

Description:
The contract can be improved by better separation of concerns:

1. Consider splitting into multiple contracts:
    - Core overdraft logic
    - Fee management
    - Price oracle integration
    - User management
2. Use inheritance for cleaner code structure

Impact:
The current structure may lead to a complex, difficult-to-maintain codebase as the protocol evolves. Better code organization could improve readability, maintainability, and make auditing easier.

Recommendation:
Refactor the contract to better separate concerns and consider implementing a modular contract architecture.

```
contract PriceOracle {
    function getRate(address pool) public view returns (uint256);
}

contract FeeManager {
    function getAccessFee(uint256 amount) public view returns (uint256);
    function getServiceFee(uint256 amount) public view returns (uint256);
}

contract ClixpesaOverdraft is
    Initializable,
    OwnableUpgradeable,
    ReentrancyGuardUpgradeable,
    UUPSUpgradeable,
    PriceOracle,
    FeeManager {
    // Main contract logic
}
```

[I-03] Potential Code Duplication

Contract: ClixpesaOverdraft
Severity: Informational
Status: Open

Description:
There is duplication in token amount calculation logic that could be refactored:

```
// Consider creating utility functions:
function _convertAmountByToken(uint256 amount, address fromToken, address
toToken) internal view returns (uint256) {
    if (fromToken == toToken) return amount;
```

```
    uint256 baseAmount = _getBaseAmount(amount, fromToken);
    return _getTokenAmount(baseAmount, toToken);
}
```

Impact:
Code duplication increases maintenance overhead and the risk of bugs if changes are made to one instance but not others.

Recommendation:
Create utility functions for common operations to reduce code duplication and improve maintainability.

[I-04] Testing Coverage

Contract: ClixpesaOverdraft
Severity: Informational
Status: Open

Description:
The contract appears to be missing comprehensive test coverage, particularly for edge cases.

Impact:
Insufficient test coverage increases the risk of undiscovered bugs and vulnerabilities.

Recommendation:

1. Implement extensive test coverage:
    - Unit tests for each function
    - Integration tests for full workflows
    - Fuzzing tests for unexpected inputs
    - Invariant tests for state consistency
2. Use formal verification tools where possible to prove critical properties of the contract.

[I-05] Documentation Comments Missing

Contract: ClixpesaRoscas
Severity: Informational
Status: Open

Description:
The contract lacks NatSpec comments for many functions, making it difficult for developers to understand the purpose and usage of each function.

Impact:

Lack of documentation reduces code readability and makes it harder for developers to use, maintain, or audit the contract.

Recommendation:
Add NatSpec comments (/// @notice, /// @param, /// @return) to all public and external functions:

```
/// @notice Requests a loan for a user
/// @param _amount The amount to borrow
/// @param _token The token address to borrow
/// @param _roscaId The ID of the ROSCA (0 for non-ROSCA loans)
/// @param borrower The address of the borrower
/// @return requestId The ID of the loan request
function requestLoan(
    uint256 _amount,
    address _token,
    uint256 _roscaId,
    address borrower
) public screening returns (uint256 requestId) {
    // Function body...
}
```

[I-06] Gas Optimizations for Storage Access

Contract: ClixpesaRoscas
Severity: Informational
Status: Open

Description:
The contract frequently accesses storage variables without caching them in memory, which increases gas costs.

Impact:
Repeated storage access leads to higher gas costs, especially for functions that are called frequently.

Recommendation:
Cache storage variables in memory when they are accessed multiple times within a function:

```
function someFunction(uint256 _roscaId) public {
    // Instead of repeatedly accessing roscas[_roscaId]
    Rosca memory rosca = roscas[_roscaId];
    if (rosca.isOpen) {
        // Do something with rosca
    }
    if (rosca.members.length > 0) {
        // Do something else with rosca
    }
}
```

[I-07] Code Structure Improvements

Contract: ClixpesaRoscas
Severity: Informational
Status: Open

Description:
The contract is quite large and handles many different functionalities, which can make it difficult to maintain and audit.

Impact:
Large, monolithic contracts are more prone to bugs, harder to audit, and costlier to deploy due to their size.

Recommendation:
Consider splitting the contract into multiple smaller contracts with specific responsibilities. This would make the code more modular, easier to maintain, and potentially reduce deployment costs.

4. StaticAnalysis(Slither)

BlockHat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

```
INFO:Detectors:
Overdraft.repayOverdraft(address,uint256,address) (src/mocks/MockOverdraft.sol#31-36) uses
arbitrary from in transferFrom: IERC20(tokenAddress).transferFrom(user,address(this),amount)
(src/mocks/MockOverdraft.sol#33)
ClixpesaOverdraft.repayOverdraft(address,address,uint256) (src/overdraft/Overdraft.sol#158-195)
uses arbitrary from in transferFrom:
require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),tokenAmount),Repaymen
t Failed) (src/overdraft/Overdraft.sol#171)
ClixpesaOverdraft.repayOverdraft(address,address,uint256) (src/overdraft/Overdraft.sol#158-195)
uses arbitrary from in transferFrom:
require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),amount),Repayment
Failed) (src/overdraft/Overdraft.sol#175)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
BaseAccount._payPrefund(uint256)
(lib/account-abstraction/contracts/core/BaseAccount.sol#103-112) sends eth to arbitrary user
 Dangerous calls:
- (success,None) = address(msg.sender).call{gas: type()(uint256).max,value:
missingAccountFunds}() (lib/account-abstraction/contracts/core/BaseAccount.sol#105-108)
SmartAccount._call(address,uint256,bytes) (src/account/SmartAccount.sol#115-122) sends eth to
arbitrary user
  Dangerous calls:
        - (success,result) = target.call{value: value}(data) (src/account/SmartAccount.sol#116)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbi
trary-destinations
INFO:Detectors:
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) has bitwise-xor operator ^
instead of the exponentiation operator **        - inverse = (3 * denominator) ^ 2
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#205)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) has bitwise-xor
operator ^ instead of the exponentiation operator **:
         - inv = (3 * denominator) ^ 2 (src/libraries/FullMath.sol#87)
Reference:
```

```
https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-exponentiation
INFO:Detectors:
Overdraft.getOverdraft(address,uint256,address) (src/mocks/MockOverdraft.sol#23-29) ignores
return value by IERC20(tokenAddress).transfer(user,amount) (src/mocks/MockOverdraft.sol#25)
Overdraft.repayOverdraft(address,uint256,address) (src/mocks/MockOverdraft.sol#31-36) ignores
return value by IERC20(tokenAddress).transferFrom(user,address(this),amount)
(src/mocks/MockOverdraft.sol#33)
ClixpesaRoscas.approveLoan(address,uint256,uint256) (src/roscas/Roscas.sol#281-327) ignores
return value by IERC20(loanRequest.token).transfer(_member,loanRequest.requestedAmount)
(src/roscas/Roscas.sol#306)
ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address) (src/roscas/Roscas.sol#342-375)
ignores return value by IERC20(loan.token).transferFrom(msg.sender,address(this),_amount)
(src/roscas/Roscas.sol#356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) performs a multiplication on
the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#190)
        - inverse = (3 * denominator) ^ 2
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#205)
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) performs a multiplication on
the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#190)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#209)
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) performs a multiplication on
the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#190)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#210)
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) performs a multiplication on
the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#190)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#211)
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) performs a multiplication on
the result of a division:
        - denominator = denominator / twos
```

```
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#190)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#212)
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) performs a multiplication on
the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#190)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#213)
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) performs a multiplication on
the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#190)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#214)
Math.mulDiv(uint256,uint256,uint256)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#144-223) performs a multiplication on
the result of a division:
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#193)
        - result = prod0 * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#220)
Math.invMod(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#243-289)
performs a multiplication on the result of a division:
        - quotient = gcd / remainder
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#265)
        - (gcd,remainder) = (remainder,gcd - remainder * quotient)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#267-274)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) performs a
multiplication on the result of a division:
        - denominator = denominator / twos (src/libraries/FullMath.sol#67)
        - inv = (3 * denominator) ^ 2 (src/libraries/FullMath.sol#87)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) performs a
multiplication on the result of a division:
        - denominator = denominator / twos (src/libraries/FullMath.sol#67)
        - inv *= 2 - denominator * inv (src/libraries/FullMath.sol#91)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) performs a
multiplication on the result of a division:
        - denominator = denominator / twos (src/libraries/FullMath.sol#67)
        - inv *= 2 - denominator * inv (src/libraries/FullMath.sol#92)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) performs a
multiplication on the result of a division:
        - denominator = denominator / twos (src/libraries/FullMath.sol#67)
        - inv *= 2 - denominator * inv (src/libraries/FullMath.sol#93)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) performs a
multiplication on the result of a division:
```

```
        - denominator = denominator / twos (src/libraries/FullMath.sol#67)
        - inv *= 2 - denominator * inv (src/libraries/FullMath.sol#94)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) performs a
multiplication on the result of a division:
        - denominator = denominator / twos (src/libraries/FullMath.sol#67)
        - inv *= 2 - denominator * inv (src/libraries/FullMath.sol#95)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) performs a
multiplication on the result of a division:
        - denominator = denominator / twos (src/libraries/FullMath.sol#67)
        - inv *= 2 - denominator * inv (src/libraries/FullMath.sol#96)
FullMath.mulDiv(uint256,uint256,uint256) (src/libraries/FullMath.sol#14-106) performs a
multiplication on the result of a division:
        - prod0 = prod0 / twos (src/libraries/FullMath.sol#72)
        - result = prod0 * inv (src/libraries/FullMath.sol#104)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0x48a170391f7dc42444e8fa2) >> 128 (src/libraries/TickMath.sol#46)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0x2216e584f5fa1ea926041bedfe98) >> 128
(src/libraries/TickMath.sol#45)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0x5d6af8dedb81196699c329225ee604) >> 128
(src/libraries/TickMath.sol#44)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0x9aa508b5b7a84e1c677de54f3e99bc9) >> 128
(src/libraries/TickMath.sol#43)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0x31be135f97d08fd981231505542fcfa6) >> 128
(src/libraries/TickMath.sol#42)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0x70d869a156d2a1b890bb3df62baf32f7) >> 128
(src/libraries/TickMath.sol#41)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xa9f746462d870fdf8a65dc1f90e061e5) >> 128
(src/libraries/TickMath.sol#40)
```

```
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xd097f3bdfd2022b8845ad8f792aa5825) >> 128
(src/libraries/TickMath.sol#39)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xe7159475a2c29b7443b29c7fa6e889d9) >> 128
(src/libraries/TickMath.sol#38)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xf3392b0822b70005940c7a398e4b70f3) >> 128
(src/libraries/TickMath.sol#37)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xf987a7253ac413176f2b074cf7815e54) >> 128
(src/libraries/TickMath.sol#36)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xfcbe86c7900a88aedcffc83b479aa3a4) >> 128
(src/libraries/TickMath.sol#35)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xfe5dee046a99a2a811c461f1969c3053) >> 128
(src/libraries/TickMath.sol#34)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xff2ea16466c96a3843ec78b326b52861) >> 128
(src/libraries/TickMath.sol#33)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xff973b41fa98c081472e6896dfb254c0) >> 128
(src/libraries/TickMath.sol#32)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xffcb9843d60f6159c9db58835c926644) >> 128
(src/libraries/TickMath.sol#31)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
```

```
on the result of a division:
        - ratio = (ratio * 0xffe5caca7e10e4e61c3624eaa0941cd0) >> 128
(src/libraries/TickMath.sol#30)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xfff2e50f5f656932ef12357cf3c7fdcc) >> 128
(src/libraries/TickMath.sol#29)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) performs a multiplication
on the result of a division:
        - ratio = (ratio * 0xfff97272373d413259a46990580e213a) >> 128
(src/libraries/TickMath.sol#28)
        - ratio = type()(uint256).max / ratio (src/libraries/TickMath.sol#48)
ClixpesaOverdraft._getBaseAmount(uint256,address) (src/overdraft/Overdraft.sol#288-300)
performs a multiplication on the result of a division:
        - (amount * 0.995e18 / rate_scope_0 * S_FACTOR) / S_FACTOR
(src/overdraft/Overdraft.sol#298)
ClixpesaOverdraft._getBaseAmount(uint256,address) (src/overdraft/Overdraft.sol#288-300)
performs a multiplication on the result of a division:
        - (amount * 0.995e18 / rate * S_FACTOR) / S_FACTOR (src/overdraft/Overdraft.sol#294)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)
(src/roscas/Roscas.sol#342-375):
        External calls:
        - IERC20(loan.token).transferFrom(msg.sender,address(this),_amount)
(src/roscas/Roscas.sol#356)
        State variables written after the call(s):
        - loan.repaidAmount += _amount (src/roscas/Roscas.sol#358)
        ClixpesaRoscas.loans (src/roscas/Roscas.sol#25) can be used in cross function
reentrancies:
        - ClixpesaRoscas.getLoan(address,uint256) (src/roscas/Roscas.sol#639-641)
        - ClixpesaRoscas.loans (src/roscas/Roscas.sol#25)
        - ClixpesaRoscas.updateLoanStatus(address,uint256,Status)
(src/roscas/Roscas.sol#568-570)
        - loan.lastRepaymentDate = block.timestamp (src/roscas/Roscas.sol#359)
        ClixpesaRoscas.loans (src/roscas/Roscas.sol#25) can be used in cross function
reentrancies:
        - ClixpesaRoscas.getLoan(address,uint256) (src/roscas/Roscas.sol#639-641)
        - ClixpesaRoscas.loans (src/roscas/Roscas.sol#25)
        - ClixpesaRoscas.updateLoanStatus(address,uint256,Status)
(src/roscas/Roscas.sol#568-570)
        - loan.status = Status.Repaid (src/roscas/Roscas.sol#368)
        ClixpesaRoscas.loans (src/roscas/Roscas.sol#25) can be used in cross function
reentrancies:
        - ClixpesaRoscas.getLoan(address,uint256) (src/roscas/Roscas.sol#639-641)
```

```
        - ClixpesaRoscas.loans (src/roscas/Roscas.sol#25)
        - ClixpesaRoscas.updateLoanStatus(address,uint256,Status)
(src/roscas/Roscas.sol#568-570)
        - loan.status = Status.PaidLate (src/roscas/Roscas.sol#370)
        ClixpesaRoscas.loans (src/roscas/Roscas.sol#25) can be used in cross function
reentrancies:
        - ClixpesaRoscas.getLoan(address,uint256) (src/roscas/Roscas.sol#639-641)
        - ClixpesaRoscas.loans (src/roscas/Roscas.sol#25)
        - ClixpesaRoscas.updateLoanStatus(address,uint256,Status)
(src/roscas/Roscas.sol#568-570)
Reentrancy in ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195):
        External calls:
        -
require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),tokenAmount),Repaymen
t Failed) (src/overdraft/Overdraft.sol#171)
        State variables written after the call(s):
        - user.overdraftDebt.amountDue = 0 (src/overdraft/Overdraft.sol#172)
        ClixpesaOverdraft.users (src/overdraft/Overdraft.sol#75) can be used in cross function
reentrancies:
        - ClixpesaOverdraft.getUser(address) (src/overdraft/Overdraft.sol#270-272)
        - ClixpesaOverdraft.getUserOverdrafts(address) (src/overdraft/Overdraft.sol#254-262)
        - ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195)
        - ClixpesaOverdraft.subscribeUser(address,uint256,string)
(src/overdraft/Overdraft.sol#197-224)
        - ClixpesaOverdraft.unsubscribeUser(address) (src/overdraft/Overdraft.sol#230-233)
        - ClixpesaOverdraft.updateUserDebt(address) (src/overdraft/Overdraft.sol#239-246)
        - ClixpesaOverdraft.useOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#115-156)
Reentrancy in ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195):
        External calls:
        -
require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),amount),Repayment
Failed) (src/overdraft/Overdraft.sol#175)
        State variables written after the call(s):
        - user.overdraftDebt.amountDue = user.overdraftDebt.amountDue - baseAmount
(src/overdraft/Overdraft.sol#176)
        ClixpesaOverdraft.users (src/overdraft/Overdraft.sol#75) can be used in cross function
reentrancies:
        - ClixpesaOverdraft.getUser(address) (src/overdraft/Overdraft.sol#270-272)
        - ClixpesaOverdraft.getUserOverdrafts(address) (src/overdraft/Overdraft.sol#254-262)
        - ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195)
        - ClixpesaOverdraft.subscribeUser(address,uint256,string)
(src/overdraft/Overdraft.sol#197-224)
```

```
        - ClixpesaOverdraft.unsubscribeUser(address) (src/overdraft/Overdraft.sol#230-233)
        - ClixpesaOverdraft.updateUserDebt(address) (src/overdraft/Overdraft.sol#239-246)
        - ClixpesaOverdraft.useOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#115-156)
Reentrancy in ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195):
        External calls:
        -
require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),tokenAmount),Repaymen
t Failed) (src/overdraft/Overdraft.sol#171)
        -
require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),amount),Repayment
Failed) (src/overdraft/Overdraft.sol#175)
        State variables written after the call(s):
        - user.overdraftDebt =
OverdraftDebt({amountDue:0,serviceFee:0,effectTime:0,dueTime:0,principal:0,lastChecked:block.ti
mestamp,state:Status.Good}) (src/overdraft/Overdraft.sol#182-190)
        ClixpesaOverdraft.users (src/overdraft/Overdraft.sol#75) can be used in cross function
reentrancies:
        - ClixpesaOverdraft.getUser(address) (src/overdraft/Overdraft.sol#270-272)
        - ClixpesaOverdraft.getUserOverdrafts(address) (src/overdraft/Overdraft.sol#254-262)
        - ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195)
        - ClixpesaOverdraft.subscribeUser(address,uint256,string)
(src/overdraft/Overdraft.sol#197-224)
        - ClixpesaOverdraft.unsubscribeUser(address) (src/overdraft/Overdraft.sol#230-233)
        - ClixpesaOverdraft.updateUserDebt(address) (src/overdraft/Overdraft.sol#239-246)
        - ClixpesaOverdraft.useOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#115-156)
        - users[userAddress] = user (src/overdraft/Overdraft.sol#191)
        ClixpesaOverdraft.users (src/overdraft/Overdraft.sol#75) can be used in cross function
reentrancies:
        - ClixpesaOverdraft.getUser(address) (src/overdraft/Overdraft.sol#270-272)
        - ClixpesaOverdraft.getUserOverdrafts(address) (src/overdraft/Overdraft.sol#254-262)
        - ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195)
        - ClixpesaOverdraft.subscribeUser(address,uint256,string)
(src/overdraft/Overdraft.sol#197-224)
        - ClixpesaOverdraft.unsubscribeUser(address) (src/overdraft/Overdraft.sol#230-233)
        - ClixpesaOverdraft.updateUserDebt(address) (src/overdraft/Overdraft.sol#239-246)
        - ClixpesaOverdraft.useOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#115-156)
        - users[userAddress] = user (src/overdraft/Overdraft.sol#193)
        ClixpesaOverdraft.users (src/overdraft/Overdraft.sol#75) can be used in cross function
reentrancies:
        - ClixpesaOverdraft.getUser(address) (src/overdraft/Overdraft.sol#270-272)
        - ClixpesaOverdraft.getUserOverdrafts(address) (src/overdraft/Overdraft.sol#254-262)
```

```
        - ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195)
        - ClixpesaOverdraft.subscribeUser(address,uint256,string)
(src/overdraft/Overdraft.sol#197-224)
        - ClixpesaOverdraft.unsubscribeUser(address) (src/overdraft/Overdraft.sol#230-233)
        - ClixpesaOverdraft.updateUserDebt(address) (src/overdraft/Overdraft.sol#239-246)
        - ClixpesaOverdraft.useOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#115-156)
Reentrancy in ClixpesaOverdraft.useOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#115-156):
        External calls:
        - require(bool,string)(IERC20(token).transfer(userAddress,amount),Tranfer failed)
(src/overdraft/Overdraft.sol#152)
        State variables written after the call(s):
        - users[userAddress] = user (src/overdraft/Overdraft.sol#153)
        ClixpesaOverdraft.users (src/overdraft/Overdraft.sol#75) can be used in cross function
reentrancies:
        - ClixpesaOverdraft.getUser(address) (src/overdraft/Overdraft.sol#270-272)
        - ClixpesaOverdraft.getUserOverdrafts(address) (src/overdraft/Overdraft.sol#254-262)
        - ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195)
        - ClixpesaOverdraft.subscribeUser(address,uint256,string)
(src/overdraft/Overdraft.sol#197-224)
        - ClixpesaOverdraft.unsubscribeUser(address) (src/overdraft/Overdraft.sol#230-233)
        - ClixpesaOverdraft.updateUserDebt(address) (src/overdraft/Overdraft.sol#239-246)
        - ClixpesaOverdraft.useOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#115-156)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ERC1967Utils.upgradeToAndCall(address,bytes)
(lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Utils.sol#67-76) ignores return
value by Address.functionDelegateCall(newImplementation,data)
(lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Utils.sol#72)
ERC1967Utils.upgradeBeaconToAndCall(address,bytes)
(lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Utils.sol#157-166) ignores return
value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data)
(lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Utils.sol#162)
ClixpesaOverdraft._getRate(address) (src/overdraft/Overdraft.sol#316-322) ignores return value
by (sqrtPriceX96,None,None,None,None,None,None) = localUSDPool.slot0()
(src/overdraft/Overdraft.sol#318)
ClixpesaRoscas.createRosca(address,address,bool) (src/roscas/Roscas.sol#131-149) ignores return
value by rosca.members.add(_admin) (src/roscas/Roscas.sol#141)
ClixpesaRoscas.joinRosca(address[],uint256) (src/roscas/Roscas.sol#151-174) ignores return
value by rosca.members.add(_members[i]) (src/roscas/Roscas.sol#166)
ClixpesaRoscas.leaveRosca() (src/roscas/Roscas.sol#176-188) ignores return value by
rosca.members.remove(msg.sender) (src/roscas/Roscas.sol#181)
```

```
ClixpesaRoscas.addMembers(address[],uint256) (src/roscas/Roscas.sol#410-433) ignores return
value by rosca.members.add(_members[i]) (src/roscas/Roscas.sol#425)
ClixpesaRoscas.removeMembers(address[]) (src/roscas/Roscas.sol#435-454) ignores return value by
rosca.members.remove(_members[i]) (src/roscas/Roscas.sol#443)
ClixpesaRoscas.changeAdmin(uint256,address) (src/roscas/Roscas.sol#456-472) ignores return
value by roscas[_roscaId].members.add(_newAdmin) (src/roscas/Roscas.sol#461)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Paymaster.constructor(IEntryPoint,address)._verifyingSigner (src/account/Paymaster.sol#29)
lacks a zero-check on :
                - verifyingSigner = _verifyingSigner (src/account/Paymaster.sol#30)
Overdraft.constructor(address)._reserveToken (src/mocks/MockOverdraft.sol#19) lacks a
zero-check on :
                - reserveToken = _reserveToken (src/mocks/MockOverdraft.sol#20)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
SmartAccount._call(address,uint256,bytes) (src/account/SmartAccount.sol#115-122) has external
calls inside a loop: (success,result) = target.call{value: value}(data)
(src/account/SmartAccount.sol#116)
        Calls stack containing the loop:
                SmartAccount.executeBatch(address[],uint256[],bytes[])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in ClixpesaRoscas.approveLoan(address,uint256,uint256)
(src/roscas/Roscas.sol#281-327):
        External calls:
        - IERC20(loanRequest.token).transfer(_member,loanRequest.requestedAmount)
(src/roscas/Roscas.sol#306)
        State variables written after the call(s):
        - loan.borrower = _member (src/roscas/Roscas.sol#309)
        - loan.id = _requestId (src/roscas/Roscas.sol#310)
        - loan.roscaId = loanRequest.roscaId (src/roscas/Roscas.sol#311)
        - loan.principalAmount = loanRequest.requestedAmount (src/roscas/Roscas.sol#312)
        - loan.interestAmount = loanRequest.interestAmount (src/roscas/Roscas.sol#313)
        - loan.repaidAmount = 0 (src/roscas/Roscas.sol#314)
        - loan.lastRepaymentDate = 0 (src/roscas/Roscas.sol#315)
        - loan.disbursedDate = block.timestamp (src/roscas/Roscas.sol#316)
        - loan.maturityDate = block.timestamp + loanRequest.tenor (src/roscas/Roscas.sol#317)
        - loan.tenor = loanRequest.tenor (src/roscas/Roscas.sol#318)
        - loan.status = Status.Active (src/roscas/Roscas.sol#319)
        - loan.frequency = loanRequest.frequency (src/roscas/Roscas.sol#320)
        - loan.token = loanRequest.token (src/roscas/Roscas.sol#321)
        - loan.numberOfInstallments = loanRequest.numberOfInstallments
(src/roscas/Roscas.sol#322)
        - loan.installmentAmount = loanRequest.installmentAmount (src/roscas/Roscas.sol#323)
        - loan.dueDate = loan.disbursedDate + loanRequest.tenor * 86400
```

```
(src/roscas/Roscas.sol#324)
Reentrancy in Overdraft.getOverdraft(address,uint256,address)
(src/mocks/MockOverdraft.sol#23-29):
        External calls:
        - IERC20(tokenAddress).transfer(user,amount) (src/mocks/MockOverdraft.sol#25)
        State variables written after the call(s):
        - overdraftDebt[user].amount += amount (src/mocks/MockOverdraft.sol#26)
        - overdraftDebt[user].takenAt = block.timestamp (src/mocks/MockOverdraft.sol#27)
Reentrancy in ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)
(src/roscas/Roscas.sol#342-375):
        External calls:
        - IERC20(loan.token).transferFrom(msg.sender,address(this),_amount)
(src/roscas/Roscas.sol#356)
        State variables written after the call(s):
        - userLoanStatus[borrower] = false (src/roscas/Roscas.sol#373)
Reentrancy in Overdraft.repayOverdraft(address,uint256,address)
(src/mocks/MockOverdraft.sol#31-36):
        External calls:
        - IERC20(tokenAddress).transferFrom(user,address(this),amount)
(src/mocks/MockOverdraft.sol#33)
        State variables written after the call(s):
        - overdraftDebt[user].amount -= amount (src/mocks/MockOverdraft.sol#34)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Overdraft.getOverdraft(address,uint256,address)
(src/mocks/MockOverdraft.sol#23-29):
        External calls:
        - IERC20(tokenAddress).transfer(user,amount) (src/mocks/MockOverdraft.sol#25)
        Event emitted after the call(s):
        - OverdraftUsed(user,amount,tokenAddress) (src/mocks/MockOverdraft.sol#28)
Reentrancy in Overdraft.repayOverdraft(address,uint256,address)
(src/mocks/MockOverdraft.sol#31-36):
        External calls:
        - IERC20(tokenAddress).transferFrom(user,address(this),amount)
(src/mocks/MockOverdraft.sol#33)
        Event emitted after the call(s):
        - OverdraftPaid(user,amount,tokenAddress) (src/mocks/MockOverdraft.sol#35)
Reentrancy in ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195):
        External calls:
        -
require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),tokenAmount),Repaymen
t Failed) (src/overdraft/Overdraft.sol#171)
        Event emitted after the call(s):
        - OverdraftPaid(userAddress,user.overdraftDebt.amountDue,token,tokenAmount)
(src/overdraft/Overdraft.sol#173)
```

```
Reentrancy in ClixpesaOverdraft.repayOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#158-195):
        External calls:
        -
require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),amount),Repayment
Failed) (src/overdraft/Overdraft.sol#175)
        Event emitted after the call(s):
        - OverdraftPaid(userAddress,baseAmount,token,amount) (src/overdraft/Overdraft.sol#177)
Reentrancy in ClixpesaOverdraft.useOverdraft(address,address,uint256)
(src/overdraft/Overdraft.sol#115-156):
        External calls:
        - require(bool,string)(IERC20(token).transfer(userAddress,amount),Tranfer failed)
(src/overdraft/Overdraft.sol#152)
        Event emitted after the call(s):
        - OverdraftUsed(userAddress,baseAmount,token,amount) (src/overdraft/Overdraft.sol#155)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
BaseAccount._payPrefund(uint256)
(lib/account-abstraction/contracts/core/BaseAccount.sol#103-112) tries to limit the gas of an
external call that controls implicit decoding
        (success,None) = address(msg.sender).call{gas: type()(uint256).max,value:
missingAccountFunds}() (lib/account-abstraction/contracts/core/BaseAccount.sol#105-108)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#return-bomb
INFO:Detectors:
ClixpesaOverdraft.updateUserDebt(address) (src/overdraft/Overdraft.sol#239-246) uses timestamp
for comparisons
        Dangerous comparisons:
        - user.overdraftDebt.lastChecked + 86400 - 1 > block.timestamp
(src/overdraft/Overdraft.sol#244)
ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address) (src/roscas/Roscas.sol#342-375) uses
timestamp for comparisons
        Dangerous comparisons:
        - loan.repaidAmount < (loan.principalAmount + loan.interestAmount)
(src/roscas/Roscas.sol#361)
        - loan.repaidAmount >= (loan.principalAmount + loan.interestAmount)
(src/roscas/Roscas.sol#365)
        - loan.dueDate + 2592000 >= loan.lastRepaymentDate (src/roscas/Roscas.sol#366)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) has a high cyclomatic
complexity (24).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
_packValidationData(ValidationData) (lib/account-abstraction/contracts/core/Helpers.sol#59-66)
is never used and should be removed
_parseValidationData(uint256) (lib/account-abstraction/contracts/core/Helpers.sol#43-53) is
```

```
never used and should be removed
calldataKeccak(bytes) (lib/account-abstraction/contracts/core/Helpers.sol#89-96) is never used
and should be removed
min(uint256,uint256) (lib/account-abstraction/contracts/core/Helpers.sol#104-106) is never used
and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in BaseAccount._payPrefund(uint256)
(lib/account-abstraction/contracts/core/BaseAccount.sol#103-112):
        - (success,None) = address(msg.sender).call{gas: type()(uint256).max,value:
missingAccountFunds}() (lib/account-abstraction/contracts/core/BaseAccount.sol#105-108)
Low level call in Address.sendValue(address,uint256)
(lib/openzeppelin-contracts/contracts/utils/Address.sol#33-42):
        - (success,returndata) = recipient.call{value: amount}()
(lib/openzeppelin-contracts/contracts/utils/Address.sol#38)
Low level call in Address.functionCallWithValue(address,bytes,uint256)
(lib/openzeppelin-contracts/contracts/utils/Address.sol#75-81):
        - (success,returndata) = target.call{value: value}(data)
(lib/openzeppelin-contracts/contracts/utils/Address.sol#79)
Low level call in Address.functionStaticCall(address,bytes)
(lib/openzeppelin-contracts/contracts/utils/Address.sol#87-90):
        - (success,returndata) = target.staticcall(data)
(lib/openzeppelin-contracts/contracts/utils/Address.sol#88)
Low level call in Address.functionDelegateCall(address,bytes)
(lib/openzeppelin-contracts/contracts/utils/Address.sol#96-99):
        - (success,returndata) = target.delegatecall(data)
(lib/openzeppelin-contracts/contracts/utils/Address.sol#97)
Low level call in SmartAccount._call(address,uint256,bytes)
(src/account/SmartAccount.sol#115-122):
        - (success,result) = target.call{value: value}(data) (src/account/SmartAccount.sol#116)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
FixedPoint96.slitherConstructorConstantVariables() (src/libraries/FixedPoint96.sol#7-10) uses
literals with too many digits:
        - Q96 = 0x1000000000000000000000000 (src/libraries/FixedPoint96.sol#9)
TickMath.getSqrtRatioAtTick(int24) (src/libraries/TickMath.sol#23-54) uses literals with too
many digits:
        - ratio = 0x100000000000000000000000000000000 (src/libraries/TickMath.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
BasePaymaster.PAYMASTER_VALIDATION_GAS_OFFSET
(lib/account-abstraction/contracts/core/BasePaymaster.sol#19) is never used in Paymaster
(src/account/Paymaster.sol#20-107)
BasePaymaster.PAYMASTER_POSTOP_GAS_OFFSET
(lib/account-abstraction/contracts/core/BasePaymaster.sol#20) is never used in Paymaster
(src/account/Paymaster.sol#20-107)
Paymaster.SIGNATURE_OFFSET (src/account/Paymaster.sol#27) is never used in Paymaster
```

```
(src/account/Paymaster.sol#20-107)
ClixpesaRoscas.__gap (src/roscas/Roscas.sol#47) is never used in ClixpesaRoscas
(src/roscas/Roscas.sol#13-705)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Overdraft.reserveToken (src/mocks/MockOverdraft.sol#14) should be immutable
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
**THIS CHECKLIST IS NOT COMPLETE**. Use `--show-ignored-findings` to show all the results.
Summary
 - [arbitrary-send-erc20](#arbitrary-send-erc20) (3 results) (High)
 - [arbitrary-send-eth](#arbitrary-send-eth) (2 results) (High)
 - [incorrect-exp](#incorrect-exp) (2 results) (High)
 - [unchecked-transfer](#unchecked-transfer) (4 results) (High)
 - [divide-before-multiply](#divide-before-multiply) (38 results) (Medium)
 - [reentrancy-no-eth](#reentrancy-no-eth) (5 results) (Medium)
 - [unused-return](#unused-return) (9 results) (Medium)
 - [missing-zero-check](#missing-zero-check) (2 results) (Low)
 - [calls-loop](#calls-loop) (1 results) (Low)
 - [reentrancy-benign](#reentrancy-benign) (4 results) (Low)
 - [reentrancy-events](#reentrancy-events) (5 results) (Low)
 - [return-bomb](#return-bomb) (1 results) (Low)
 - [timestamp](#timestamp) (2 results) (Low)
 - [assembly](#assembly) (38 results) (Informational)
 - [pragma](#pragma) (1 results) (Informational)
 - [cyclomatic-complexity](#cyclomatic-complexity) (1 results) (Informational)
 - [dead-code](#dead-code) (4 results) (Informational)
 - [solc-version](#solc-version) (5 results) (Informational)
 - [low-level-calls](#low-level-calls) (6 results) (Informational)
 - [naming-convention](#naming-convention) (89 results) (Informational)
 - [too-many-digits](#too-many-digits) (2 results) (Informational)
 - [unused-state](#unused-state) (4 results) (Informational)
 - [immutable-states](#immutable-states) (1 results) (Optimization)
## arbitrary-send-erc20
Impact: High
Confidence: High
 - [ ] ID-0
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L195) uses arbitrary from in transferFrom:
[require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),amount),RepaymentFailed)](src/overdraft/Overdraft.sol#L175)

src/overdraft/Overdraft.sol#L158-L195
 - [ ] ID-1
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L195) uses arbitrary from in transferFrom:
```

[require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),tokenAmount),Repayment Failed)](src/overdraft/Overdraft.sol#L171)
src/overdraft/Overdraft.sol#L158-L195
 - [ ] ID-2
[Overdraft.repayOverdraft(address,uint256,address)](src/mocks/MockOverdraft.sol#L31-L36) *uses arbitrary from in transferFrom:*
[IERC20(tokenAddress).transferFrom(user,address(this),amount)](src/mocks/MockOverdraft.sol#L33)
src/mocks/MockOverdraft.sol#L31-L36
arbitrary-send-eth
Impact: High
Confidence: Medium
 - [ ] ID-3
[SmartAccount._call(address,uint256,bytes)](src/account/SmartAccount.sol#L115-L122) *sends eth to arbitrary user*
        Dangerous calls:
        - [(success,result) = target.call{value:
value}(data)](src/account/SmartAccount.sol#L116)

src/account/SmartAccount.sol#L115-L122
 - [ ] ID-4
[BaseAccount._payPrefund(uint256)](lib/account-abstraction/contracts/core/BaseAccount.sol#L103-L112) *sends eth to arbitrary user*
        Dangerous calls:
        - [(success,None) = address(msg.sender).call{gas: type()(uint256).max,value:
missingAccountFunds}()](lib/account-abstraction/contracts/core/BaseAccount.sol#L105-L108)

lib/account-abstraction/contracts/core/BaseAccount.sol#L103-L112
incorrect-exp
Impact: High
Confidence: Medium
 - [ ] ID-5
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) *has bitwise-xor operator ^ instead of the exponentiation operator **:*
         - [inv = (3 * denominator) ^ 2](src/libraries/FullMath.sol#L87)

src/libraries/FullMath.sol#L14-L106
 - [ ] ID-6
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223) *has bitwise-xor operator ^ instead of the exponentiation operator **:*
         - [inverse = (3 * denominator) ^
2](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L205)
lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223
unchecked-transfer
Impact: High
Confidence: Medium
 - [ ] ID-7
[Overdraft.getOverdraft(address,uint256,address)](src/mocks/MockOverdraft.sol#L23-L29) *ignores*

```
return value by [IERC20(tokenAddress).transfer(user,amount)](src/mocks/MockOverdraft.sol#L25)
src/mocks/MockOverdraft.sol#L23-L29
 - [ ] ID-8
[Overdraft.repayOverdraft(address,uint256,address)](src/mocks/MockOverdraft.sol#L31-L36)
ignores return value by
[IERC20(tokenAddress).transferFrom(user,address(this),amount)](src/mocks/MockOverdraft.sol#L33)
src/mocks/MockOverdraft.sol#L31-L36
 - [ ] ID-9
[ClixpesaRoscas.approveLoan(address,uint256,uint256)](src/roscas/Roscas.sol#L281-L327) ignores
return value by
[IERC20(loanRequest.token).transfer(_member,loanRequest.requestedAmount)](src/roscas/Roscas.sol
#L306)
src/roscas/Roscas.sol#L281-L327
 - [ ] ID-10
[ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)](src/roscas/Roscas.sol#L342-L375)
ignores return value by
[IERC20(loan.token).transferFrom(msg.sender,address(this),_amount)](src/roscas/Roscas.sol#L356)
src/roscas/Roscas.sol#L342-L375
## divide-before-multiply
Impact: Medium
Confidence: Medium
 - [ ] ID-11
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol
#L144-L223) performs a multiplication on the result of a division:
        - [denominator = denominator /
twos](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L190)
        - [inverse *= 2 - denominator *
inverse](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L212)

lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223
 - [ ] ID-12
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xff973b41fa98c081472e6896dfb254c0) >>
128](src/libraries/TickMath.sol#L32)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-13
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) performs a
multiplication on the result of a division:
        - [denominator = denominator / twos](src/libraries/FullMath.sol#L67)
        - [inv = (3 * denominator) ^ 2](src/libraries/FullMath.sol#L87)

src/libraries/FullMath.sol#L14-L106
 - [ ] ID-14
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
```

```
multiplication on the result of a division:
        - [ratio = (ratio * 0x5d6af8dedb81196699c329225ee604) >>
128](src/libraries/TickMath.sol#L44)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-15
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0x48a170391f7dc42444e8fa2) >> 128](src/libraries/TickMath.sol#L46)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)
src/libraries/TickMath.sol#L23-L54
 - [ ] ID-16
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xe7159475a2c29b7443b29c7fa6e889d9) >>
128](src/libraries/TickMath.sol#L38)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-17
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0x31be135f97d08fd981231505542fcfa6) >>
128](src/libraries/TickMath.sol#L42)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-18
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol
#L144-L223) performs a multiplication on the result of a division:
        - [denominator = denominator /
twos](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L190)
        - [inverse = (3 * denominator) ^
2](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L205)
lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223

 - [ ] ID-19
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol
#L144-L223) performs a multiplication on the result of a division:
        - [prod0 = prod0 / twos](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L193)
        - [result = prod0 *
inverse](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L220)
lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223
 - [ ] ID-20
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
```

```
        - [ratio = (ratio * 0x2216e584f5fa1ea926041bedfe98) >>
128](src/libraries/TickMath.sol#L45)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)
src/libraries/TickMath.sol#L23-L54
 - [ ] ID-21
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) performs a
multiplication on the result of a division:
        - [denominator = denominator / twos](src/libraries/FullMath.sol#L67)
        - [inv *= 2 - denominator * inv](src/libraries/FullMath.sol#L92)
src/libraries/FullMath.sol#L14-L106
 - [ ] ID-22
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xfe5dee046a99a2a811c461f1969c3053) >>
128](src/libraries/TickMath.sol#L34)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-23
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xffe5caca7e10e4e61c3624eaa0941cd0) >>
128](src/libraries/TickMath.sol#L30)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-24
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol
#L144-L223) performs a multiplication on the result of a division:
        - [denominator = denominator /
twos](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L190)
        - [inverse *= 2 - denominator *
inverse](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L209)

lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223
 - [ ] ID-25
[ClixpesaOverdraft._getBaseAmount(uint256,address)](src/overdraft/Overdraft.sol#L288-L300)
performs a multiplication on the result of a division:
        - [(amount * 0.995e18 / rate * S_FACTOR) / S_FACTOR](src/overdraft/Overdraft.sol#L294)

src/overdraft/Overdraft.sol#L288-L300
 - [ ] ID-26
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) performs a
multiplication on the result of a division:
        - [denominator = denominator / twos](src/libraries/FullMath.sol#L67)
        - [inv *= 2 - denominator * inv](src/libraries/FullMath.sol#L94)
```

```
src/libraries/FullMath.sol#L14-L106
 - [ ] ID-27
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xfff2e50f5f656932ef12357cf3c7fdcc) >>
128](src/libraries/TickMath.sol#L29)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)
src/libraries/TickMath.sol#L23-L54
 - [ ] ID-28
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) performs a
multiplication on the result of a division:
        - [prod0 = prod0 / twos](src/libraries/FullMath.sol#L72)
        - [result = prod0 * inv](src/libraries/FullMath.sol#L104)
src/libraries/FullMath.sol#L14-L106
 - [ ] ID-29
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xd097f3bdfd2022b8845ad8f792aa5825) >>
128](src/libraries/TickMath.sol#L39)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-30
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol
#L144-L223) performs a multiplication on the result of a division:
        - [denominator = denominator /
twos](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L190)
        - [inverse *= 2 - denominator *
inverse](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L210)

lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223

 - [ ] ID-31
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) performs a
multiplication on the result of a division:
        - [denominator = denominator / twos](src/libraries/FullMath.sol#L67)
        - [inv *= 2 - denominator * inv](src/libraries/FullMath.sol#L96)

src/libraries/FullMath.sol#L14-L106
 - [ ] ID-32
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol
#L144-L223) performs a multiplication on the result of a division:
        - [denominator = denominator /
twos](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L190)
        - [inverse *= 2 - denominator *
inverse](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L214)
lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223
```

```
 - [ ] ID-33
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xfcbe86c7900a88aedcffc83b479aa3a4) >>
128](src/libraries/TickMath.sol#L35)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-34
[ClixpesaOverdraft._getBaseAmount(uint256,address)](src/overdraft/Overdraft.sol#L288-L300)
performs a multiplication on the result of a division:
        - [(amount * 0.995e18 / rate_scope_0 * S_FACTOR) /
S_FACTOR](src/overdraft/Overdraft.sol#L298)

src/overdraft/Overdraft.sol#L288-L300
 - [ ] ID-35
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xfff97272373d413259a46990580e213a) >>
128](src/libraries/TickMath.sol#L28)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-36
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xf987a7253ac413176f2b074cf7815e54) >>
128](src/libraries/TickMath.sol#L36)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)
src/libraries/TickMath.sol#L23-L54
 - [ ] ID-37
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xa9f746462d870fdf8a65dc1f90e061e5) >>
128](src/libraries/TickMath.sol#L40)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54


 - [ ] ID-38
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0x9aa508b5b7a84e1c677de54f3e99bc9) >>
128](src/libraries/TickMath.sol#L43)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)
```

```
src/libraries/TickMath.sol#L23-L54

 - [ ] ID-39
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol
#L144-L223) performs a multiplication on the result of a division:
        - [denominator = denominator /
twos](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L190)
        - [inverse *= 2 - denominator *
inverse](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L211)

lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223
 - [ ] ID-40
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xffcb9843d60f6159c9db58835c926644) >>
128](src/libraries/TickMath.sol#L31)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)
src/libraries/TickMath.sol#L23-L54

 - [ ] ID-41
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xff2ea16466c96a3843ec78b326b52861) >>
128](src/libraries/TickMath.sol#L33)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-42
[Math.invMod(uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L243-L2
89) performs a multiplication on the result of a division:
        - [quotient = gcd /
remainder](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L265)
        - [(gcd,remainder) = (remainder,gcd - remainder *
quotient)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L267-L274)

lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L243-L289
 - [ ] ID-43
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) performs a
multiplication on the result of a division:
        - [denominator = denominator / twos](src/libraries/FullMath.sol#L67)
        - [inv *= 2 - denominator * inv](src/libraries/FullMath.sol#L95)

src/libraries/FullMath.sol#L14-L106
 - [ ] ID-44
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0x70d869a156d2a1b890bb3df62baf32f7) >>
```

```
128](src/libraries/TickMath.sol#L41)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54

 - [ ] ID-45
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) performs a
multiplication on the result of a division:
        - [denominator = denominator / twos](src/libraries/FullMath.sol#L67)
        - [inv *= 2 - denominator * inv](src/libraries/FullMath.sol#L91)

src/libraries/FullMath.sol#L14-L106
 - [ ] ID-46
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) performs a
multiplication on the result of a division:
        - [denominator = denominator / twos](src/libraries/FullMath.sol#L67)
        - [inv *= 2 - denominator * inv](src/libraries/FullMath.sol#L93)

src/libraries/FullMath.sol#L14-L106
 - [ ] ID-47
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) performs a
multiplication on the result of a division:
        - [ratio = (ratio * 0xf3392b0822b70005940c7a398e4b70f3) >>
128](src/libraries/TickMath.sol#L37)
        - [ratio = type()(uint256).max / ratio](src/libraries/TickMath.sol#L48)

src/libraries/TickMath.sol#L23-L54
 - [ ] ID-48
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol
#L144-L223) performs a multiplication on the result of a division:
        - [denominator = denominator /
twos](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L190)
        - [inverse *= 2 - denominator *
inverse](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L213)

lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223

## reentrancy-no-eth
Impact: Medium
Confidence: Medium
 - [ ] ID-49
Reentrancy in
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95):
        External calls:        -
[require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),tokenAmount),Repayme
nt Failed)](src/overdraft/Overdraft.sol#L171)
```

```
        -
[require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),amount),Repayment
Failed)](src/overdraft/Overdraft.sol#L175)
        State variables written after the call(s):
        - [user.overdraftDebt =
OverdraftDebt({amountDue:0,serviceFee:0,effectTime:0,dueTime:0,principal:0,lastChecked:block.ti
mestamp,state:Status.Good})](src/overdraft/Overdraft.sol#L182-L190)
        [ClixpesaOverdraft.users](src/overdraft/Overdraft.sol#L75) can be used in cross
function reentrancies:
        - [ClixpesaOverdraft.getUser(address)](src/overdraft/Overdraft.sol#L270-L272)
        - [ClixpesaOverdraft.getUserOverdrafts(address)](src/overdraft/Overdraft.sol#L254-L262)
        -
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95)
        -
[ClixpesaOverdraft.subscribeUser(address,uint256,string)](src/overdraft/Overdraft.sol#L197-L224
)
        - [ClixpesaOverdraft.unsubscribeUser(address)](src/overdraft/Overdraft.sol#L230-L233)
        - [ClixpesaOverdraft.updateUserDebt(address)](src/overdraft/Overdraft.sol#L239-L246)
        -
[ClixpesaOverdraft.useOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L115-L156
)
        - [users[userAddress] = user](src/overdraft/Overdraft.sol#L191)
        [ClixpesaOverdraft.users](src/overdraft/Overdraft.sol#L75) can be used in cross
function reentrancies:
        - [ClixpesaOverdraft.getUser(address)](src/overdraft/Overdraft.sol#L270-L272)
        - [ClixpesaOverdraft.getUserOverdrafts(address)](src/overdraft/Overdraft.sol#L254-L262)
        -
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95)
        -
[ClixpesaOverdraft.subscribeUser(address,uint256,string)](src/overdraft/Overdraft.sol#L197-L224
)
        - [ClixpesaOverdraft.unsubscribeUser(address)](src/overdraft/Overdraft.sol#L230-L233)
        - [ClixpesaOverdraft.updateUserDebt(address)](src/overdraft/Overdraft.sol#L239-L246)
        -
[ClixpesaOverdraft.useOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L115-L156
)
        - [users[userAddress] = user](src/overdraft/Overdraft.sol#L193)
        [ClixpesaOverdraft.users](src/overdraft/Overdraft.sol#L75) can be used in cross
function reentrancies:
        - [ClixpesaOverdraft.getUser(address)](src/overdraft/Overdraft.sol#L270-L272)
        - [ClixpesaOverdraft.getUserOverdrafts(address)](src/overdraft/Overdraft.sol#L254-L262)
        -
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95)
        -
```

[ClixpesaOverdraft.subscribeUser(address,uint256,string)](src/overdraft/Overdraft.sol#L197-L224)
        - [ClixpesaOverdraft.unsubscribeUser(address)](src/overdraft/Overdraft.sol#L230-L233)
        - [ClixpesaOverdraft.updateUserDebt(address)](src/overdraft/Overdraft.sol#L239-L246)
        - [ClixpesaOverdraft.useOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L115-L156)
src/overdraft/Overdraft.sol#L158-L195
 - [ ] ID-50
Reentrancy in [ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L195):
        External calls: - [require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),tokenAmount),Repayment Failed)](src/overdraft/Overdraft.sol#L171)
        State variables written after the call(s):
        - [user.overdraftDebt.amountDue = 0](src/overdraft/Overdraft.sol#L172)
        [ClixpesaOverdraft.users](src/overdraft/Overdraft.sol#L75) can be used in cross function reentrancies:
        - [ClixpesaOverdraft.getUser(address)](src/overdraft/Overdraft.sol#L270-L272)
        - [ClixpesaOverdraft.getUserOverdrafts(address)](src/overdraft/Overdraft.sol#L254-L262)
        - [ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L195)
        - [ClixpesaOverdraft.subscribeUser(address,uint256,string)](src/overdraft/Overdraft.sol#L197-L224)
        - [ClixpesaOverdraft.unsubscribeUser(address)](src/overdraft/Overdraft.sol#L230-L233)
        - [ClixpesaOverdraft.updateUserDebt(address)](src/overdraft/Overdraft.sol#L239-L246)
        - [ClixpesaOverdraft.useOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L115-L156)
src/overdraft/Overdraft.sol#L158-L195
 - [ ] ID-51
Reentrancy in [ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)](src/roscas/Roscas.sol#L342-L375):
        External calls:
        - [IERC20(loan.token).transferFrom(msg.sender,address(this),_amount)](src/roscas/Roscas.sol#L356)
        State variables written after the call(s):
        - [loan.repaidAmount += _amount](src/roscas/Roscas.sol#L358)
        [ClixpesaRoscas.loans](src/roscas/Roscas.sol#L25) can be used in cross function reentrancies:
        - [ClixpesaRoscas.getLoan(address,uint256)](src/roscas/Roscas.sol#L639-L641)
        - [ClixpesaRoscas.loans](src/roscas/Roscas.sol#L25)
        - [ClixpesaRoscas.updateLoanStatus(address,uint256,Status)](src/roscas/Roscas.sol#L568-L570)

```
        - [loan.lastRepaymentDate = block.timestamp](src/roscas/Roscas.sol#L359)
        [ClixpesaRoscas.loans](src/roscas/Roscas.sol#L25) can be used in cross function
reentrancies:
        - [ClixpesaRoscas.getLoan(address,uint256)](src/roscas/Roscas.sol#L639-L641)
        - [ClixpesaRoscas.loans](src/roscas/Roscas.sol#L25)
        -
[ClixpesaRoscas.updateLoanStatus(address,uint256,Status)](src/roscas/Roscas.sol#L568-L570)
        - [loan.status = Status.Repaid](src/roscas/Roscas.sol#L368)
        [ClixpesaRoscas.loans](src/roscas/Roscas.sol#L25) can be used in cross function
reentrancies:
        - [ClixpesaRoscas.getLoan(address,uint256)](src/roscas/Roscas.sol#L639-L641)
        - [ClixpesaRoscas.loans](src/roscas/Roscas.sol#L25)
        -
[ClixpesaRoscas.updateLoanStatus(address,uint256,Status)](src/roscas/Roscas.sol#L568-L570)
        - [loan.status = Status.PaidLate](src/roscas/Roscas.sol#L370)
        [ClixpesaRoscas.loans](src/roscas/Roscas.sol#L25) can be used in cross function
reentrancies:
        - [ClixpesaRoscas.getLoan(address,uint256)](src/roscas/Roscas.sol#L639-L641)
        - [ClixpesaRoscas.loans](src/roscas/Roscas.sol#L25)
        -
[ClixpesaRoscas.updateLoanStatus(address,uint256,Status)](src/roscas/Roscas.sol#L568-L570)
src/roscas/Roscas.sol#L342-L375
 - [ ] ID-52
Reentrancy in
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95):
        External calls:
        -
[require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),amount),Repayment
Failed)](src/overdraft/Overdraft.sol#L175)
        State variables written after the call(s):
        - [user.overdraftDebt.amountDue = user.overdraftDebt.amountDue -
baseAmount](src/overdraft/Overdraft.sol#L176)
        [ClixpesaOverdraft.users](src/overdraft/Overdraft.sol#L75) can be used in cross
function reentrancies:
        - [ClixpesaOverdraft.getUser(address)](src/overdraft/Overdraft.sol#L270-L272)
        - [ClixpesaOverdraft.getUserOverdrafts(address)](src/overdraft/Overdraft.sol#L254-L262)
        -
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95)
        -
[ClixpesaOverdraft.subscribeUser(address,uint256,string)](src/overdraft/Overdraft.sol#L197-L224
)
        - [ClixpesaOverdraft.unsubscribeUser(address)](src/overdraft/Overdraft.sol#L230-L233)
        - [ClixpesaOverdraft.updateUserDebt(address)](src/overdraft/Overdraft.sol#L239-L246)
        -
[ClixpesaOverdraft.useOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L115-L156
```

```
)
src/overdraft/Overdraft.sol#L158-L195
 - [ ] ID-53
Reentrancy in
[ClixpesaOverdraft.useOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L115-L156
):
        External calls:
        - [require(bool,string)(IERC20(token).transfer(userAddress,amount),Tranfer
failed)](src/overdraft/Overdraft.sol#L152)
        State variables written after the call(s):
        - [users[userAddress] = user](src/overdraft/Overdraft.sol#L153)
        [ClixpesaOverdraft.users](src/overdraft/Overdraft.sol#L75) can be used in cross
function reentrancies:
        - [ClixpesaOverdraft.getUser(address)](src/overdraft/Overdraft.sol#L270-L272)
        - [ClixpesaOverdraft.getUserOverdrafts(address)](src/overdraft/Overdraft.sol#L254-L262)
        -
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95)
        -
[ClixpesaOverdraft.subscribeUser(address,uint256,string)](src/overdraft/Overdraft.sol#L197-L224
)
        - [ClixpesaOverdraft.unsubscribeUser(address)](src/overdraft/Overdraft.sol#L230-L233)
        - [ClixpesaOverdraft.updateUserDebt(address)](src/overdraft/Overdraft.sol#L239-L246)
        -
[ClixpesaOverdraft.useOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L115-L156
)
src/overdraft/Overdraft.sol#L115-L156
## unused-return
Impact: Medium
Confidence: Medium
 - [ ] ID-54
[ERC1967Utils.upgradeBeaconToAndCall(address,bytes)](lib/openzeppelin-contracts/contracts/proxy
/ERC1967/ERC1967Utils.sol#L157-L166) ignores return value by
[Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data)](lib/openzeppelin-contr
acts/contracts/proxy/ERC1967/ERC1967Utils.sol#L162)
lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Utils.sol#L157-L166
 - [ ] ID-55
[ClixpesaRoscas.addMembers(address[],uint256)](src/roscas/Roscas.sol#L410-L433) ignores return
value by [rosca.members.add(_members[i])](src/roscas/Roscas.sol#L425)
src/roscas/Roscas.sol#L410-L433
 - [ ] ID-56
[ClixpesaRoscas.leaveRosca()](src/roscas/Roscas.sol#L176-L188) ignores return value by
[rosca.members.remove(msg.sender)](src/roscas/Roscas.sol#L181)
src/roscas/Roscas.sol#L176-L188
 - [ ] ID-57
[ERC1967Utils.upgradeToAndCall(address,bytes)](lib/openzeppelin-contracts/contracts/proxy/ERC19
67/ERC1967Utils.sol#L67-L76) ignores return value by
```

[Address.functionDelegateCall(newImplementation,data)](lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Utils.sol#L72)
lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Utils.sol#L67-L76
 - [ ] ID-58
[ClixpesaRoscas.changeAdmin(uint256,address)](src/roscas/Roscas.sol#L456-L472) *ignores return value by [roscas[_roscaId].members.add(_newAdmin)](src/roscas/Roscas.sol#L461)*
src/roscas/Roscas.sol#L456-L472
 - [ ] ID-59
[ClixpesaOverdraft._getRate(address)](src/overdraft/Overdraft.sol#L316-L322) *ignores return value by [(sqrtPriceX96,None,None,None,None,None,None) = localUSDPool.slot0()](src/overdraft/Overdraft.sol#L318)*
src/overdraft/Overdraft.sol#L316-L322
 - [ ] ID-60
[ClixpesaRoscas.joinRosca(address[],uint256)](src/roscas/Roscas.sol#L151-L174) *ignores return value by [rosca.members.add(_members[i])](src/roscas/Roscas.sol#L166)*
src/roscas/Roscas.sol#L151-L174
 - [ ] ID-61
[ClixpesaRoscas.removeMembers(address[])](src/roscas/Roscas.sol#L435-L454) *ignores return value by [rosca.members.remove(_members[i])](src/roscas/Roscas.sol#L443)*

src/roscas/Roscas.sol#L435-L454
 - [ ] ID-62
[ClixpesaRoscas.createRosca(address,address,bool)](src/roscas/Roscas.sol#L131-L149) *ignores return value by [rosca.members.add(_admin)](src/roscas/Roscas.sol#L141)*
src/roscas/Roscas.sol#L131-L149
missing-zero-check
Impact: Low
Confidence: Medium
 - [ ] ID-63
[Overdraft.constructor(address)._reserveToken](src/mocks/MockOverdraft.sol#L19) *lacks a zero-check on :*
                - [reserveToken = _reserveToken](src/mocks/MockOverdraft.sol#L20)

src/mocks/MockOverdraft.sol#L19
 - [ ] ID-64
[Paymaster.constructor(IEntryPoint,address)._verifyingSigner](src/account/Paymaster.sol#L29) *lacks a zero-check on :*
                - [verifyingSigner = _verifyingSigner](src/account/Paymaster.sol#L30)

src/account/Paymaster.sol#L29
calls-loop
Impact: Low
Confidence: Medium
 - [ ] ID-65
[SmartAccount._call(address,uint256,bytes)](src/account/SmartAccount.sol#L115-L122) *has external calls inside a loop: [(success,result) = target.call{value: value}(data)](src/account/SmartAccount.sol#L116)*

```
        Calls stack containing the loop:
                SmartAccount.executeBatch(address[],uint256[],bytes[])
src/account/SmartAccount.sol#L115-L122
## reentrancy-benign
Impact: Low
Confidence: Medium
 - [ ] ID-66
Reentrancy in
[ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)](src/roscas/Roscas.sol#L342-L375):
        External calls:        -
[IERC20(loan.token).transferFrom(msg.sender,address(this),_amount)](src/roscas/Roscas.sol#L356)
        State variables written after the call(s):
        - [userLoanStatus[borrower] = false](src/roscas/Roscas.sol#L373)
src/roscas/Roscas.sol#L342-L375
 - [ ] ID-67
Reentrancy in
[Overdraft.repayOverdraft(address,uint256,address)](src/mocks/MockOverdraft.sol#L31-L36):
        External calls:        -
[IERC20(tokenAddress).transferFrom(user,address(this),amount)](src/mocks/MockOverdraft.sol#L33)
        State variables written after the call(s):
        - [overdraftDebt[user].amount -= amount](src/mocks/MockOverdraft.sol#L34)
src/mocks/MockOverdraft.sol#L31-L36
 - [ ] ID-68
Reentrancy in
[Overdraft.getOverdraft(address,uint256,address)](src/mocks/MockOverdraft.sol#L23-L29):
        External calls:
        - [IERC20(tokenAddress).transfer(user,amount)](src/mocks/MockOverdraft.sol#L25)
        State variables written after the call(s):
        - [overdraftDebt[user].amount += amount](src/mocks/MockOverdraft.sol#L26)
        - [overdraftDebt[user].takenAt = block.timestamp](src/mocks/MockOverdraft.sol#L27)
src/mocks/MockOverdraft.sol#L23-L29
 - [ ] ID-69
Reentrancy in
[ClixpesaRoscas.approveLoan(address,uint256,uint256)](src/roscas/Roscas.sol#L281-L327):
        External calls:        -
[IERC20(loanRequest.token).transfer(_member,loanRequest.requestedAmount)](src/roscas/Roscas.sol
#L306)
        State variables written after the call(s):
        - [loan.borrower = _member](src/roscas/Roscas.sol#L309)
        - [loan.id = _requestId](src/roscas/Roscas.sol#L310)
        - [loan.roscaId = loanRequest.roscaId](src/roscas/Roscas.sol#L311)
        - [loan.principalAmount = loanRequest.requestedAmount](src/roscas/Roscas.sol#L312)
        - [loan.interestAmount = loanRequest.interestAmount](src/roscas/Roscas.sol#L313)
        - [loan.repaidAmount = 0](src/roscas/Roscas.sol#L314)
        - [loan.lastRepaymentDate = 0](src/roscas/Roscas.sol#L315)
        - [loan.disbursedDate = block.timestamp](src/roscas/Roscas.sol#L316)
        - [loan.maturityDate = block.timestamp + loanRequest.tenor](src/roscas/Roscas.sol#L317)
```

```
        - [loan.tenor = loanRequest.tenor](src/roscas/Roscas.sol#L318)
        - [loan.status = Status.Active](src/roscas/Roscas.sol#L319)
        - [loan.frequency = loanRequest.frequency](src/roscas/Roscas.sol#L320)
        - [loan.token = loanRequest.token](src/roscas/Roscas.sol#L321)
        - [loan.numberOfInstallments =
loanRequest.numberOfInstallments](src/roscas/Roscas.sol#L322)
        - [loan.installmentAmount = loanRequest.installmentAmount](src/roscas/Roscas.sol#L323)
        - [loan.dueDate = loan.disbursedDate + loanRequest.tenor *
86400](src/roscas/Roscas.sol#L324)
src/roscas/Roscas.sol#L281-L327
## reentrancy-events
Impact: Low
Confidence: Medium
 - [ ] ID-70
Reentrancy in
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95):
        External calls:        -
[require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),amount),Repayment
Failed)](src/overdraft/Overdraft.sol#L175)
        Event emitted after the call(s):        -
[OverdraftPaid(userAddress,baseAmount,token,amount)](src/overdraft/Overdraft.sol#L177)
src/overdraft/Overdraft.sol#L158-L195
 - [ ] ID-71
Reentrancy in
[ClixpesaOverdraft.repayOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L158-L1
95):
        External calls:        -
[require(bool,string)(IERC20(token).transferFrom(userAddress,address(this),tokenAmount),Repayme
nt Failed)](src/overdraft/Overdraft.sol#L171)
        Event emitted after the call(s):        -
[OverdraftPaid(userAddress,user.overdraftDebt.amountDue,token,tokenAmount)](src/overdraft/Overd
raft.sol#L173)
src/overdraft/Overdraft.sol#L158-L195

 - [ ] ID-72
Reentrancy in
[ClixpesaOverdraft.useOverdraft(address,address,uint256)](src/overdraft/Overdraft.sol#L115-L156
):
        External calls:
        - [require(bool,string)(IERC20(token).transfer(userAddress,amount),Tranfer
failed)](src/overdraft/Overdraft.sol#L152)
        Event emitted after the call(s):
        -
[OverdraftUsed(userAddress,baseAmount,token,amount)](src/overdraft/Overdraft.sol#L155)

src/overdraft/Overdraft.sol#L115-L156
```

```
 - [ ] ID-73
Reentrancy in
[Overdraft.repayOverdraft(address,uint256,address)](src/mocks/MockOverdraft.sol#L31-L36):
External calls:        -
[IERC20(tokenAddress).transferFrom(user,address(this),amount)](src/mocks/MockOverdraft.sol#L33)
        Event emitted after the call(s):
        - [OverdraftPaid(user,amount,tokenAddress)](src/mocks/MockOverdraft.sol#L35)

src/mocks/MockOverdraft.sol#L31-L36
 - [ ] ID-74
Reentrancy in
[Overdraft.getOverdraft(address,uint256,address)](src/mocks/MockOverdraft.sol#L23-L29):
        External calls:
        - [IERC20(tokenAddress).transfer(user,amount)](src/mocks/MockOverdraft.sol#L25)
        Event emitted after the call(s):
        - [OverdraftUsed(user,amount,tokenAddress)](src/mocks/MockOverdraft.sol#L28)
src/mocks/MockOverdraft.sol#L23-L29
## return-bomb
Impact: Low
Confidence: Medium
 - [ ] ID-75
[BaseAccount._payPrefund(uint256)](lib/account-abstraction/contracts/core/BaseAccount.sol#L103-
L112) tries to limit the gas of an external call that controls implicit decoding
        [(success,None) = address(msg.sender).call{gas: type()(uint256).max,value:
missingAccountFunds}()](lib/account-abstraction/contracts/core/BaseAccount.sol#L105-L108)

lib/account-abstraction/contracts/core/BaseAccount.sol#L103-L112
## timestamp
Impact: Low
Confidence: Medium
 - [ ] ID-76
[ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)](src/roscas/Roscas.sol#L342-L375)
uses timestamp for comparisons
        Dangerous comparisons:
        - [loan.repaidAmount < (loan.principalAmount +
loan.interestAmount)](src/roscas/Roscas.sol#L361)
        - [loan.repaidAmount >= (loan.principalAmount +
loan.interestAmount)](src/roscas/Roscas.sol#L365)
        - [loan.dueDate + 2592000 >= loan.lastRepaymentDate](src/roscas/Roscas.sol#L366)

src/roscas/Roscas.sol#L342-L375
 - [ ] ID-77
[ClixpesaOverdraft.updateUserDebt(address)](src/overdraft/Overdraft.sol#L239-L246) uses
timestamp for comparisons
        Dangerous comparisons:
        - [user.overdraftDebt.lastChecked + 86400 - 1 >
block.timestamp](src/overdraft/Overdraft.sol#L244)
```

src/overdraft/Overdraft.sol#L239-L246

assembly
Impact: Informational
Confidence: High
 - [ ] ID-78
[SafeERC20._callOptionalReturn(IERC20,bytes)](lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#L159-L177) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#L162-L172)
lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#L159-L177

 - [ ] ID-79
[StorageSlot.getAddressSlot(bytes32)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L66-L70) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L67-L69)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L66-L70

 - [ ] ID-80
[Math.tryModExp(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L337-L361) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L339-L360)
lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L337-L361

 - [ ] ID-81
[ReentrancyGuardUpgradeable._getReentrancyGuardStorage()](lib/openzeppelin-contracts-upgradeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L49-L53) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts-upgradeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L50-L52)
lib/openzeppelin-contracts-upgradeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L49-L53
 - [ ] ID-82
[Create2.deploy(uint256,bytes32,bytes)](lib/openzeppelin-contracts/contracts/utils/Create2.sol#L37-L56) uses assembly
       - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/Create2.sol#L44-L52)
lib/openzeppelin-contracts/contracts/utils/Create2.sol#L37-L56

 - [ ] ID-83
[Initializable._getInitializableStorage()](lib/openzeppelin-contracts-upgradeable/contracts/proxy/utils/Initializable.sol#L232-L237) uses assembly   - [INLINE ASM](lib/openzeppelin-contracts-upgradeable/contracts/proxy/utils/Initializable.sol#L234-L236)
lib/openzeppelin-contracts-upgradeable/contracts/proxy/utils/Initializable.sol#L232-L237

 - [ ] ID-84
[Proxy._delegate(address)](lib/openzeppelin-contracts/contracts/proxy/Proxy.sol#L22-L45) uses assembly

```
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/proxy/Proxy.sol#L23-L44)
lib/openzeppelin-contracts/contracts/proxy/Proxy.sol#L22-L45

 - [ ] ID-85
[Create2.computeAddress(bytes32,bytes32,address)](lib/openzeppelin-contracts/contracts/utils/Cr
eate2.sol#L70-L91) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/Create2.sol#L71-L90)
lib/openzeppelin-contracts/contracts/utils/Create2.sol#L70-L91
 - [ ] ID-86
[SafeCast.toUint(bool)](lib/openzeppelin-contracts/contracts/utils/math/SafeCast.sol#L1157-L116
1) uses assembly
        - [INLINE
ASM](lib/openzeppelin-contracts/contracts/utils/math/SafeCast.sol#L1158-L1160)
lib/openzeppelin-contracts/contracts/utils/math/SafeCast.sol#L1157-L1161
 - [ ] ID-87
[StorageSlot.getInt256Slot(bytes32)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol
#L102-L106) uses assembly
       - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L103-L105)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L102-L106
 - [ ] ID-88
[Strings.toChecksumHexString(address)](lib/openzeppelin-contracts/contracts/utils/Strings.sol#L
103-L121) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/Strings.sol#L108-L110)
lib/openzeppelin-contracts/contracts/utils/Strings.sol#L103-L121

 - [ ] ID-89
[ECDSA.tryRecover(bytes32,bytes)](lib/openzeppelin-contracts/contracts/utils/cryptography/ECDSA
.sol#L56-L75) uses assembly        - [INLINE
ASM](lib/openzeppelin-contracts/contracts/utils/cryptography/ECDSA.sol#L66-L70)
lib/openzeppelin-contracts/contracts/utils/cryptography/ECDSA.sol#L56-L75


 - [ ] ID-90
[OwnableUpgradeable._getOwnableStorage()](lib/openzeppelin-contracts-upgradeable/contracts/acce
ss/OwnableUpgradeable.sol#L30-L34) uses assembly
        -
[INLINEASM](lib/openzeppelin-contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#L31-
L33)
lib/openzeppelin-contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#L30-L34
 - [ ] ID-91
[Panic.panic(uint256)](lib/openzeppelin-contracts/contracts/utils/Panic.sol#L50-L56) uses
assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/Panic.sol#L51-L55)
lib/openzeppelin-contracts/contracts/utils/Panic.sol#L50-L56

 - [ ] ID-92
[StorageSlot.getBytesSlot(bytes32)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#
```

```
L129-L133) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L130-L132)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L129-L133
 - [ ] ID-93
[Math.tryModExp(bytes,bytes,bytes)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L3
77-L399) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L389-L398)
lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L377-L399

 - [ ] ID-94
[SmartAccount._call(address,uint256,bytes)](src/account/SmartAccount.sol#L115-L122) uses
assembly
        - [INLINE ASM](src/account/SmartAccount.sol#L118-L120)
src/account/SmartAccount.sol#L115-L122

 - [ ] ID-95
[FullMath.mulDiv(uint256,uint256,uint256)](src/libraries/FullMath.sol#L14-L106) uses assembly

src/libraries/FullMath.sol#L14-L106
- [ ] ID-96
[EnumerableSet.values(EnumerableSet.UintSet)](lib/openzeppelin-contracts/contracts/utils/struct
s/EnumerableSet.sol#L365-L374) uses assembly
        - [INLINE
ASM](lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#L369-L371)

lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#L365-L374
 - [ ] ID-97
[StorageSlot.getStringSlot(string)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#
L120-L124) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L121-L123)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L120-L124

 - [ ] ID-98
[MessageHashUtils.toTypedDataHash(bytes32,bytes32)](lib/openzeppelin-contracts/contracts/utils/
cryptography/MessageHashUtils.sol#L75-L83) uses assembly
        - [INLINE
ASM](lib/openzeppelin-contracts/contracts/utils/cryptography/MessageHashUtils.sol#L76-L82)

lib/openzeppelin-contracts/contracts/utils/cryptography/MessageHashUtils.sol#L75-L83

 - [ ] ID-99
[StorageSlot.getBytes32Slot(bytes32)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.so
l#L84-L88) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L85-L87)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L84-L88

 - [ ] ID-100
```

[StorageSlot.getBytesSlot(bytes)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L138-L142) uses assembly
	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L139-L141)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L138-L142

- [ ] ID-101
[Strings._unsafeReadBytesOffset(bytes,uint256)](lib/openzeppelin-contracts/contracts/utils/Strings.sol#L435-L440) uses assembly
	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/Strings.sol#L437-L439)
lib/openzeppelin-contracts/contracts/utils/Strings.sol#L435-L440
- [ ] ID-102
[Address._revert(bytes)](lib/openzeppelin-contracts/contracts/utils/Address.sol#L138-L149) uses assembly
	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/Address.sol#L142-L145)
lib/openzeppelin-contracts/contracts/utils/Address.sol#L138-L149
- [ ] ID-103
[StorageSlot.getBooleanSlot(bytes32)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L75-L79) uses assembly
	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L76-L78)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L75-L79
- [ ] ID-104
[EnumerableSet.values(EnumerableSet.Bytes32Set)](lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#L219-L228) uses assembly	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#L223-L225)
lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#L219-L228
- [ ] ID-105
[StorageSlot.getStringSlot(bytes32)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L111-L115) uses assembly
	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L112-L114)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L111-L115

- [ ] ID-106
[SafeERC20._callOptionalReturnBool(IERC20,bytes)](lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#L187-L197) uses assembly	- [INLINE ASM](lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#L191-L195)
lib/openzeppelin-contracts/contracts/token/ERC20/utils/SafeERC20.sol#L187-L197
- [ ] ID-107
[Initializable._getInitializableStorage()](lib/openzeppelin-contracts/contracts/proxy/utils/Initializable.sol#L223-L227) uses assembly	- [INLINE ASM](lib/openzeppelin-contracts/contracts/proxy/utils/Initializable.sol#L224-L226)
lib/openzeppelin-contracts/contracts/proxy/utils/Initializable.sol#L223-L227
- [ ] ID-108
[Math.mulDiv(uint256,uint256,uint256)](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223) uses assembly
	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L151-L154)
	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L175-L182)
	- [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L188-L197)

lib/openzeppelin-contracts/contracts/utils/math/Math.sol#L144-L223
 - [ ] ID-109
[Strings.toString(uint256)](lib/openzeppelin-contracts/contracts/utils/Strings.sol#L37-L55) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/Strings.sol#L42-L44)
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/Strings.sol#L47-L49)
lib/openzeppelin-contracts/contracts/utils/Strings.sol#L37-L55
 - [ ] ID-110
[TickMath.getTickAtSqrtRatio(uint160)](src/libraries/TickMath.sol#L61-L204) uses assembly

src/libraries/TickMath.sol#L61-L204
 - [ ] ID-111
[StorageSlot.getUint256Slot(bytes32)](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L93-L97) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L94-L96)
lib/openzeppelin-contracts/contracts/utils/StorageSlot.sol#L93-L97

 - [ ] ID-112
[EnumerableSet.values(EnumerableSet.AddressSet)](lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#L292-L301) uses assembly
       - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#L296-L298)
lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#L292-L301
 - [ ] ID-113
[MessageHashUtils.toEthSignedMessageHash(bytes32)](lib/openzeppelin-contracts/contracts/utils/cryptography/MessageHashUtils.sol#L30-L36) uses assembly
        - [INLINE ASM](lib/openzeppelin-contracts/contracts/utils/cryptography/MessageHashUtils.sol#L31-L35)
lib/openzeppelin-contracts/contracts/utils/cryptography/MessageHashUtils.sol#L30-L36

 - [ ] ID-114
[UserOperationLib.getSender(PackedUserOperation)](lib/account-abstraction/contracts/core/UserOperationLib.sol#L21-L30) uses assembly
        - [INLINE ASM](lib/account-abstraction/contracts/core/UserOperationLib.sol#L26-L28)
lib/account-abstraction/contracts/core/UserOperationLib.sol#L21-L30
 - [ ] ID-115
[AccessControlUpgradeable._getAccessControlStorage()](lib/openzeppelin-contracts-upgradeable/contracts/access/AccessControlUpgradeable.sol#L67-L71) uses assembly   - [INLINE ASM](lib/openzeppelin-contracts-upgradeable/contracts/access/AccessControlUpgradeable.sol#L68-L70)

lib/openzeppelin-contracts-upgradeable/contracts/access/AccessControlUpgradeable.sol#L67-L71
pragma
Impact: Informational
Confidence: High

```
 - [ ] ID-116
8 different versions of Solidity are used:
        - Version constraint ^0.8.23 is used by:
                -[^0.8.23](lib/account-abstraction/contracts/core/BaseAccount.sol#L2)
lib/account-abstraction/contracts/core/BaseAccount.sol#L2
## cyclomatic-complexity
Impact: Informational
Confidence: High
 - [ ] ID-117
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) has a high cyclomatic
complexity (24).

src/libraries/TickMath.sol#L23-L54
## dead-code
Impact: Informational
Confidence: Medium
 - [ ] ID-118
[_packValidationData(ValidationData)](lib/account-abstraction/contracts/core/Helpers.sol#L59-L6
6) is never used and should be removed

lib/account-abstraction/contracts/core/Helpers.sol#L59-L66
 - [ ] ID-119
[_parseValidationData(uint256)](lib/account-abstraction/contracts/core/Helpers.sol#L43-L53) is
never used and should be removed
lib/account-abstraction/contracts/core/Helpers.sol#L43-L53
 - [ ] ID-120
[calldataKeccak(bytes)](lib/account-abstraction/contracts/core/Helpers.sol#L89-L96) is never
used and should be removed
lib/account-abstraction/contracts/core/Helpers.sol#L89-L96

 - [ ] ID-121
[min(uint256,uint256)](lib/account-abstraction/contracts/core/Helpers.sol#L104-L106) is never
used and should be removed
lib/account-abstraction/contracts/core/Helpers.sol#L104-L106

## low-level-calls
Impact: Informational
Confidence: High
 - [ ] ID-127
Low level call in
[SmartAccount._call(address,uint256,bytes)](src/account/SmartAccount.sol#L115-L122):
        - [(success,result) = target.call{value:
value}(data)](src/account/SmartAccount.sol#L116)
src/account/SmartAccount.sol#L115-L122
 - [ ] ID-128
Low level call in
[BaseAccount._payPrefund(uint256)](lib/account-abstraction/contracts/core/BaseAccount.sol#L103-
```

```
L112):
        - [(success,None) = address(msg.sender).call{gas: type()(uint256).max,value:
missingAccountFunds}()](lib/account-abstraction/contracts/core/BaseAccount.sol#L105-L108)

lib/account-abstraction/contracts/core/BaseAccount.sol#L103-L112

 - [ ] ID-129
Low level call in
[Address.functionDelegateCall(address,bytes)](lib/openzeppelin-contracts/contracts/utils/Addres
s.sol#L96-L99):
        - [(success,returndata) =
target.delegatecall(data)](lib/openzeppelin-contracts/contracts/utils/Address.sol#L97)
lib/openzeppelin-contracts/contracts/utils/Address.sol#L96-L99
 - [ ] ID-130
Low level call in
[Address.functionCallWithValue(address,bytes,uint256)](lib/openzeppelin-contracts/contracts/uti
ls/Address.sol#L75-L81):        - [(success,returndata) = target.call{value:
value}(data)](lib/openzeppelin-contracts/contracts/utils/Address.sol#L79)
lib/openzeppelin-contracts/contracts/utils/Address.sol#L75-L81
 - [ ] ID-131
Low level call in
[Address.sendValue(address,uint256)](lib/openzeppelin-contracts/contracts/utils/Address.sol#L33
-L42):
        - [(success,returndata) = recipient.call{value:
amount}()](lib/openzeppelin-contracts/contracts/utils/Address.sol#L38)

lib/openzeppelin-contracts/contracts/utils/Address.sol#L33-L42
 - [ ] ID-132
Low level call in
[Address.functionStaticCall(address,bytes)](lib/openzeppelin-contracts/contracts/utils/Address.
sol#L87-L90):
        - [(success,returndata) =
target.staticcall(data)](lib/openzeppelin-contracts/contracts/utils/Address.sol#L88)
lib/openzeppelin-contracts/contracts/utils/Address.sol#L87-L90
## naming-convention
Impact: Informational
Confidence: High
 - [ ] ID-133
Parameter [ClixpesaRoscas.addMembers(address[],uint256)._members](src/roscas/Roscas.sol#L410)
is not in mixedCase
src/roscas/Roscas.sol#L410

 - [ ] ID-134
Parameter
[ClixpesaRoscas.requestLoan(uint256,uint256,uint256,Frequency,uint8,address,uint256,address)._b
orrower](src/roscas/Roscas.sol#L205) is not in mixedCase
src/roscas/Roscas.sol#L205
```

- [ ] ID-135
Parameter [ClixpesaRoscas.requestRoscaLoan(uint256,uint256,uint256)._tenor](src/roscas/Roscas.sol#L495) is not in mixedCase
src/roscas/Roscas.sol#L495
- [ ] ID-136
Constant [AccessControlUpgradeable.AccessControlStorageLocation](lib/openzeppelin-contracts-upgradeable/contracts/access/AccessControlUpgradeable.sol#L65) is not in UPPER_CASE_WITH_UNDERSCORES
lib/openzeppelin-contracts-upgradeable/contracts/access/AccessControlUpgradeable.sol#L65
- [ ] ID-137
Parameter [ClixpesaRoscas.registerMembers(address[])._members](src/roscas/Roscas.sol#L386) is not in mixedCase
src/roscas/Roscas.sol#L386
- [ ] ID-138
Parameter [ClixpesaRoscas.createRosca(address,address,bool)._tokenAddress](src/roscas/Roscas.sol#L131) is not in mixedCase
src/roscas/Roscas.sol#L131
- [ ] ID-139
Variable [ClixpesaRoscas.__gap](src/roscas/Roscas.sol#L47) is not in mixedCase
src/roscas/Roscas.sol#L47
- [ ] ID-140
Parameter [ClixpesaRoscas.togglePause(bool)._status](src/roscas/Roscas.sol#L701) is not in mixedCase
src/roscas/Roscas.sol#L701
- [ ] ID-141
Parameter [ClixpesaRoscas.rejectRoscaLoanRequest(uint256,uint256)._roscaLoanId](src/roscas/Roscas.sol#L559) is not in mixedCase
src/roscas/Roscas.sol#L559
- [ ] ID-142
Parameter [ClixpesaRoscas.changeAdmin(uint256,address)._roscaId](src/roscas/Roscas.sol#L456) is not in mixedCase
src/roscas/Roscas.sol#L456
- [ ] ID-143
Parameter [ClixpesaRoscas.joinRosca(address[],uint256)._roscaId](src/roscas/Roscas.sol#L151) is not in mixedCase
src/roscas/Roscas.sol#L151
- [ ] ID-144
Parameter [ClixpesaRoscas.rejectLoan(address,uint256,uint256)._roscaId](src/roscas/Roscas.sol#L329) is not in mixedCase
src/roscas/Roscas.sol#L329
- [ ] ID-145
Parameter [ClixpesaRoscas.requestLoan(uint256,uint256,uint256,Frequency,uint8,address,uint256,address)._t

```
oken](src/roscas/Roscas.sol#L203) is not in mixedCase
src/roscas/Roscas.sol#L203
 - [ ] ID-146
Function
[AccessControlUpgradeable.__AccessControl_init_unchained()](lib/openzeppelin-contracts-upgradea
ble/contracts/access/AccessControlUpgradeable.sol#L85-L86) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/access/AccessControlUpgradeable.sol#L85-L86
 - [ ] ID-147
Constant
[OwnableUpgradeable.OwnableStorageLocation](lib/openzeppelin-contracts-upgradeable/contracts/ac
cess/OwnableUpgradeable.sol#L28) is not in UPPER_CASE_WITH_UNDERSCORES
lib/openzeppelin-contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#L28
 - [ ] ID-148
Parameter
[ClixpesaRoscas.requestRoscaLoan(uint256,uint256,uint256)._requestedAmount](src/roscas/Roscas.s
ol#L495) is not in mixedCase
src/roscas/Roscas.sol#L495
 - [ ] ID-149
Parameter [ClixpesaRoscas.getRoscaLoan(uint256,uint256)._roscaId](src/roscas/Roscas.sol#L631)
is not in mixedCase
src/roscas/Roscas.sol#L631
 - [ ] ID-150
Parameter
[MockUniswapV3Pool.setSlot0(uint160,int24,uint16,uint16,uint16,uint8,bool)._feeProtocol](src/mo
cks/MockUniswapV3Pool.sol#L39) is not in mixedCase
src/mocks/MockUniswapV3Pool.sol#L39
 - [ ] ID-151
Parameter
[ClixpesaRoscas.approveRoscaLoanRequest(uint256,uint256,uint256)._roscaLoanId](src/roscas/Rosca
s.sol#L518) is not in mixedCase
src/roscas/Roscas.sol#L518

 - [ ] ID-152
Function
[ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained()](lib/openzeppelin-contracts-upgr
adeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L64-L67) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L64-L67
 - [ ] ID-153
Parameter [ClixpesaRoscas.sendTokens(address,address,uint256)._to](src/roscas/Roscas.sol#L488)
is not in mixedCase
src/roscas/Roscas.sol#L488
 - [ ] ID-154
Parameter [ClixpesaRoscas.addMembers(address[],uint256)._roscaId](src/roscas/Roscas.sol#L410)
is not in mixedCase
src/roscas/Roscas.sol#L410
 - [ ] ID-155
Parameter
```

[ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)._amount](src/roscas/Roscas.sol#L342) is not in mixedCase
src/roscas/Roscas.sol#L342
 - [ ] ID-156
Parameter
[ClixpesaRoscas.signOffLoanRequest(address,uint256)._member](src/roscas/Roscas.sol#L262) is not in mixedCase
src/roscas/Roscas.sol#L262
 - [ ] ID-157
Parameter
[MockUniswapV3Pool.setSlot0(uint160,int24,uint16,uint16,uint16,uint8,bool)._unlocked](src/mocks/MockUniswapV3Pool.sol#L40) is not in mixedCase
src/mocks/MockUniswapV3Pool.sol#L40
 - [ ] ID-158
Parameter
[ClixpesaRoscas.approveRoscaLoanRequest(uint256,uint256,uint256)._roscaId](src/roscas/Roscas.sol#L518) is not in mixedCase
src/roscas/Roscas.sol#L518
 - [ ] ID-159
Function
[OwnableUpgradeable.__Ownable_init(address)](lib/openzeppelin-contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#L51-L53) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#L51-L53
 - [ ] ID-160
Parameter
[ClixpesaRoscas.approveLoan(address,uint256,uint256)._requestId](src/roscas/Roscas.sol#L281) is not in mixedCasesrc/roscas/Roscas.sol#L281
 - [ ] ID-161
Parameter
[ClixpesaOverdraft.initialize(address[],address[],string)._key](src/overdraft/Overdraft.sol#L105) is not in mixedCase
src/overdraft/Overdraft.sol#L105
 - [ ] ID-162
Parameter
[ClixpesaRoscas.requestLoan(uint256,uint256,uint256,Frequency,uint8,address,uint256,address)._frequency](src/roscas/Roscas.sol#L201) is not in mixedCase
src/roscas/Roscas.sol#L201
 - [ ] ID-163
Parameter
[MockUniswapV3Pool.setSlot0(uint160,int24,uint16,uint16,uint16,uint8,bool)._observationCardinalityNext](src/mocks/MockUniswapV3Pool.sol#L38) is not in mixedCase
src/mocks/MockUniswapV3Pool.sol#L38
 - [ ] ID-164
Parameter [ClixpesaRoscas.setRoscaStatus(uint256,bool)._roscaId](src/roscas/Roscas.sol#L474) is not in mixedCase
src/roscas/Roscas.sol#L474

- [ ] ID-165
Parameter [ClixpesaRoscas.removeMembers(address[])._members](src/roscas/Roscas.sol#L435) is not in mixedCase
src/roscas/Roscas.sol#L435
- [ ] ID-166
Parameter [ClixpesaRoscas.setRoscaStatus(uint256,bool)._isOpen](src/roscas/Roscas.sol#L474) is not in mixedCasesrc/roscas/Roscas.sol#L474
- [ ] ID-167
Parameter [ClixpesaRoscas.requestLoan(uint256,uint256,uint256,Frequency,uint8,address,uint256,address)._tenor](src/roscas/Roscas.sol#L200) is not in mixedCase
src/roscas/Roscas.sol#L200
- [ ] ID-168
Parameter [ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)._requestId](src/roscas/Roscas.sol#L342) is not in mixedCase
sc/roscas/Roscas.sol#L342
- [ ] ID-169
Parameter [ClixpesaRoscas.signOffLoanRequest(address,uint256)._requestId](src/roscas/Roscas.sol#L262) is not in mixedCase
src/roscas/Roscas.sol#L262
- [ ] ID-170
Function [OwnableUpgradeable.__Ownable_init_unchained(address)](lib/openzeppelin-contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#L55-L60) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#L55-L60
- [ ] ID-171
Parameter [ClixpesaRoscas.approveLoan(address,uint256,uint256)._member](src/roscas/Roscas.sol#L281) is not in mixedCase
src/roscas/Roscas.sol#L281
- [ ] ID-172
Function [ReentrancyGuardUpgradeable.__ReentrancyGuard_init()](lib/openzeppelin-contracts-upgradeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L60-L62) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L60-L62
- [ ] ID-173
Parameter [ClixpesaRoscas.getLoan(address,uint256)._member](src/roscas/Roscas.sol#L639) is not in mixedCase
src/roscas/Roscas.sol#L639
- [ ] ID-174
Function [ERC165Upgradeable.__ERC165_init_unchained()](lib/openzeppelin-contracts-upgradeable/contracts/utils/introspection/ERC165Upgradeable.sol#L25-L26) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/utils/introspection/ERC165Upgradeable.sol#L25-L26

```
 - [ ] ID-175
Function
[ContextUpgradeable.__Context_init_unchained()](lib/openzeppelin-contracts-upgradeable/contract
s/utils/ContextUpgradeable.sol#L21-L22) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/utils/ContextUpgradeable.sol#L21-L22
 - [ ] ID-176
Parameter [ClixpesaRoscas.joinRosca(address[],uint256)._members](src/roscas/Roscas.sol#L151) is
not in mixedCase
src/roscas/Roscas.sol#L151
 - [ ] ID-177
Parameter
[ClixpesaRoscas.requestLoan(uint256,uint256,uint256,Frequency,uint8,address,uint256,address)._r
equestedAmount](src/roscas/Roscas.sol#L198) is not in mixedCase
src/roscas/Roscas.sol#L198
 - [ ] ID-178
Function
[UUPSUpgradeable.__UUPSUpgradeable_init_unchained()](lib/openzeppelin-contracts-upgradeable/con
tracts/proxy/utils/UUPSUpgradeable.sol#L68-L69) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/proxy/utils/UUPSUpgradeable.sol#L68-L69
 - [ ] ID-179
Parameter [ClixpesaRoscas.blockAddress(address,bool)._blocked](src/roscas/Roscas.sol#L484) is
not in mixedCase

src/roscas/Roscas.sol#L484
 - [ ] ID-180
Function
[AccessControlUpgradeable.__AccessControl_init()](lib/openzeppelin-contracts-upgradeable/contra
cts/access/AccessControlUpgradeable.sol#L82-L83) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/access/AccessControlUpgradeable.sol#L82-L83
 - [ ] ID-181
Struct [Overdraft.overdraft](src/mocks/MockOverdraft.sol#L8-L11) is not in CapWords
src/mocks/MockOverdraft.sol#L8-L11
 - [ ] ID-182
Parameter
[ClixpesaOverdraft.initialize(address[],address[],string)._supportedTokens](src/overdraft/Overd
raft.sol#L103) is not in mixedCase
src/overdraft/Overdraft.sol#L10
 - [ ] ID-183
Parameter [ClixpesaRoscas.grantSignatoryRole(address)._member](src/roscas/Roscas.sol#L664) is
not in mixedCase
src/roscas/Roscas.sol#L664
 - [ ] ID-184
Parameter [ClixpesaRoscas.getLoanRequest(address,uint256)._member](src/roscas/Roscas.sol#L635)
is not in mixedCase
sc/roscas/Roscas.sol#L635
 - [ ] ID-185
Parameter
```

[ClixpesaRoscas.updateLoanStatus(address,uint256,Status)._requestId](src/roscas/Roscas.sol#L568) is not in mixedCase
src/roscas/Roscas.sol#L56
 - [ ] ID-186
Parameter [ClixpesaRoscas.updateLoanStatus(address,uint256,Status)._member](src/roscas/Roscas.sol#L568) is not in mixedCase
src/roscas/Roscas.sol#L568
 - [ ] ID-187
Parameter [ClixpesaRoscas.requestLoan(uint256,uint256,uint256,Frequency,uint8,address,uint256,address)._numberOfInstallments](src/roscas/Roscas.sol#L202) is not in mixedCase
src/roscas/Roscas.sol#L202
 - [ ] ID-188
Parameter [ClixpesaRoscas.getRosca(uint256)._roscaId](src/roscas/Roscas.sol#L618) is not in mixedCase
src/roscas/Roscas.sol#L618
 - [ ] ID-189
Parameter [ClixpesaRoscas.getRoscaLoan(uint256,uint256)._roscaLoanId](src/roscas/Roscas.sol#L631) is not in mixedCase
src/roscas/Roscas.sol#L631
 - [ ] ID-190
Parameter [ClixpesaRoscas.unregisterMember(address)._member](src/roscas/Roscas.sol#L404) is not in mixedCase
src/roscas/Roscas.sol#L404
 - [ ] ID-191
Variable [UUPSUpgradeable.__self](lib/openzeppelin-contracts/contracts/proxy/utils/UUPSUpgradeable.sol#L21) is not in mixedCase
lib/openzeppelin-contracts/contracts/proxy/utils/UUPSUpgradeable.sol#L21
 - [ ] ID-192
Parameter [MockUniswapV3Pool.setSlot0(uint160,int24,uint16,uint16,uint16,uint8,bool)._observationCardinality](src/mocks/MockUniswapV3Pool.sol#L37) is not in mixedCase
sc/mocks/MockUniswapV3Pool.sol#L37
 - [ ] ID-193
Parameter [ClixpesaRoscas.approveLoan(address,uint256,uint256)._roscaId](src/roscas/Roscas.sol#L281) is not in mixedCase
src/roscas/Roscas.sol#L281
 - [ ] ID-194
Parameter [MockUniswapV3Pool.setSlot0(uint160,int24,uint16,uint16,uint16,uint8,bool)._tick](src/mocks/MockUniswapV3Pool.sol#L35) is not in mixedCase
src/mocks/MockUniswapV3Pool.sol#L35
 - [ ] ID-195

Parameter [ClixpesaRoscas.rejectRoscaLoanRequest(uint256,uint256)._roscaId](src/roscas/Roscas.sol#L559) is not in mixedCase

src/roscas/Roscas.sol#L559
 - [ ] ID-196
Function [UUPSUpgradeable.__UUPSUpgradeable_init()](lib/openzeppelin-contracts-upgradeable/contracts/proxy/utils/UUPSUpgradeable.sol#L65-L66) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/proxy/utils/UUPSUpgradeable.sol#L65-L66
 - [ ] ID-197
Parameter [MockUniswapV3Pool.setSlot0(uint160,int24,uint16,uint16,uint16,uint8,bool)._observationIndex](src/mocks/MockUniswapV3Pool.sol#L36) is not in mixedCase
src/mocks/MockUniswapV3Pool.sol#L36
 - [ ] ID-198
Parameter [ClixpesaRoscas.changeAdmin(uint256,address)._newAdmin](src/roscas/Roscas.sol#L456) is not in mixedCase
src/roscas/Roscas.sol#L456
 - [ ] ID-199
Parameter [ClixpesaRoscas.requestRoscaLoan(uint256,uint256,uint256)._roscaId](src/roscas/Roscas.sol#L495) is not in mixedCase
src/roscas/Roscas.sol#L495
 - [ ] ID-200
Parameter [ClixpesaRoscas.createRosca(address,address,bool)._admin](src/roscas/Roscas.sol#L131) is not in mixedCase
src/roscas/Roscas.sol#L131
 - [ ] ID-201
Parameter [ClixpesaRoscas.getLoanRequest(address,uint256)._requestId](src/roscas/Roscas.sol#L635) is not in mixedCase
src/roscas/Roscas.sol#L635
 - [ ] ID-202
Constant [ReentrancyGuardUpgradeable.ReentrancyGuardStorageLocation](lib/openzeppelin-contracts-upgradeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L47) is not in UPPER_CASE_WITH_UNDERSCORES
lib/openzeppelin-contracts-upgradeable/contracts/utils/ReentrancyGuardUpgradeable.sol#L47
 - [ ] ID-203
Parameter [ClixpesaRoscas.sendTokens(address,address,uint256)._amount](src/roscas/Roscas.sol#L488) is not in mixedCase
src/roscas/Roscas.sol#L488
 - [ ] ID-204
Parameter [ClixpesaRoscas.requestLoan(uint256,uint256,uint256,Frequency,uint8,address,uint256,address)._roscaId](src/roscas/Roscas.sol#L204) is not in mixedCase

src/roscas/Roscas.sol#L204
 - [ ] ID-205
Parameter
[ClixpesaRoscas.requestLoan(uint256,uint256,uint256,Frequency,uint8,address,uint256,address)._interestAmount](src/roscas/Roscas.sol#L199) is not in mixedCase
src/roscas/Roscas.sol#L199
 - [ ] ID-206
Parameter
[ClixpesaRoscas.repayLoan(uint256,uint256,uint256,address)._roscaId](src/roscas/Roscas.sol#L342) is not in mixedCase

src/roscas/Roscas.sol#L342
 - [ ] ID-207
Function
[ContextUpgradeable.__Context_init()](lib/openzeppelin-contracts-upgradeable/contracts/utils/ContextUpgradeable.sol#L18-L19) is not in mixedCase
lib/openzeppelin-contracts-upgradeable/contracts/utils/ContextUpgradeable.sol#L18-L19
 - [ ] ID-208
Parameter
[ClixpesaRoscas.approveRoscaLoanRequest(uint256,uint256,uint256)._interestAmount](src/roscas/Roscas.sol#L518) is not in mixedCase
src/roscas/Roscas.sol#L518
 - [ ] ID-209
Function
[ERC165Upgradeable.__ERC165_init()](lib/openzeppelin-contracts-upgradeable/contracts/utils/introspection/ERC165Upgradeable.sol#L22-L23) is not in mixedCase
lb/openzeppelin-contracts-upgradeable/contracts/utils/introspection/ERC165Upgradeable.sol#L22-L23
 - [ ] ID-210
Parameter
[ClixpesaRoscas.updateLoanStatus(address,uint256,Status)._status](src/roscas/Roscas.sol#L568) is not in mixedCase
src/roscas/Roscas.sol#L568
 - [ ] ID-211
Parameter
[ClixpesaRoscas.getRoscaLoanRequest(uint256,uint256)._roscaId](src/roscas/Roscas.sol#L623) is not in mixedCase
src/roscas/Roscas.sol#L623
 - [ ] ID-212
Parameter
[ClixpesaRoscas.topUpRoscaLoanPool(uint256,uint256)._amount](src/roscas/Roscas.sol#L572) is not in mixedCase
src/roscas/Roscas.sol#L572
 - [ ] ID-213
Parameter
[ClixpesaRoscas.rejectLoan(address,uint256,uint256)._member](src/roscas/Roscas.sol#L329) is not in mixedCase

src/roscas/Roscas.sol#L329
 - [ ] ID-214
Parameter
[ClixpesaOverdraft.initialize(address[],address[],string)._uniswapV3Pools](src/overdraft/Overdraft.sol#L104) is not in mixedCase
src/overdraft/Overdraft.sol#L104
 - [ ] ID-215
Parameter
[ClixpesaRoscas.rejectLoan(address,uint256,uint256)._requestId](src/roscas/Roscas.sol#L329) is not in mixedCase
src/roscas/Roscas.sol#L329
 - [ ] ID-216
Parameter [ClixpesaRoscas.emptyRoscaLoanPool(uint256)._roscaId](src/roscas/Roscas.sol#L577) is not in mixedCase
src/roscas/Roscas.sol#L577
 - [ ] ID-217
Parameter
[ClixpesaRoscas.sendTokens(address,address,uint256)._tokenAddress](src/roscas/Roscas.sol#L488) is not in mixedCase
src/roscas/Roscas.sol#L488
 - [ ] ID-218
Parameter
[ClixpesaRoscas.topUpRoscaLoanPool(uint256,uint256)._roscaId](src/roscas/Roscas.sol#L572) is not in mixedCase
src/roscas/Roscas.sol#L572
 - [ ] ID-219
Parameter
[MockUniswapV3Pool.setSlot0(uint160,int24,uint16,uint16,uint16,uint8,bool)._sqrtPriceX96](src/mocks/MockUniswapV3Pool.sol#L34) is not in mixedCase
src/mocks/MockUniswapV3Pool.sol#L34
 - [ ] ID-220
Parameter [ClixpesaRoscas.getLoan(address,uint256)._roscaLoanId](src/roscas/Roscas.sol#L639) is not in mixedCase
src/roscas/Roscas.sol#L639
 - [ ] ID-221
Parameter [ClixpesaRoscas.blockAddress(address,bool)._address](src/roscas/Roscas.sol#L484) is not in mixedCase
src/roscas/Roscas.sol#L484
too-many-digits
Impact: Informational
Confidence: Medium
 - [ ] ID-222
[FixedPoint96.slitherConstructorConstantVariables()](src/libraries/FixedPoint96.sol#L7-L10) uses literals with too many digits:
        - [Q96 = 0x1000000000000000000000000](src/libraries/FixedPoint96.sol#L9)
src/libraries/FixedPoint96.sol#L7-L10
 - [ ] ID-223

```
[TickMath.getSqrtRatioAtTick(int24)](src/libraries/TickMath.sol#L23-L54) uses literals with too
many digits:
        - [ratio = 0x100000000000000000000000000000000](src/libraries/TickMath.sol#L27)
src/libraries/TickMath.sol#L23-L54
## unused-state
Impact: Informational
Confidence: High
 - [ ] ID-224
[Paymaster.SIGNATURE_OFFSET](src/account/Paymaster.sol#L27) is never used in
[Paymaster](src/account/Paymaster.sol#L20-L107)

src/account/Paymaster.sol#L27
 - [ ] ID-225
[BasePaymaster.PAYMASTER_POSTOP_GAS_OFFSET](lib/account-abstraction/contracts/core/BasePaymaste
r.sol#L20) is never used in [Paymaster](src/account/Paymaster.sol#L20-L107)
lib/account-abstraction/contracts/core/BasePaymaster.sol#L20
 - [ ] ID-226
[ClixpesaRoscas.__gap](src/roscas/Roscas.sol#L47) is never used in
[ClixpesaRoscas](src/roscas/Roscas.sol#L13-L705)
src/roscas/Roscas.sol#L47
 - [ ] ID-227
[BasePaymaster.PAYMASTER_VALIDATION_GAS_OFFSET](lib/account-abstraction/contracts/core/BasePaym
aster.sol#L19) is never used in [Paymaster](src/account/Paymaster.sol#L20-L107)
lib/account-abstraction/contracts/core/BasePaymaster.sol#L19

## immutable-states
Impact: Optimization
Confidence: High
 - [ ] ID-228
[Overdraft.reserveToken](src/mocks/MockOverdraft.sol#L14) should be immutable
src/mocks/MockOverdraft.sol#L14
INFO:Slither:. analyzed (65 contracts with 100 detectors), 229 result(s) found
```

5. Conclusion

Our comprehensive security audit of the ClixpesaOverdraft and ClixpesaRoscas contracts has identified several security vulnerabilities and improvement opportunities across both contracts. The most critical issues include price manipulation vulnerabilities, missing access controls, reentrancy risks, and business logic flaws in both contracts.

The ClixpesaOverdraft contract handles overdraft facilities for users but contains several vulnerabilities, notably the critical price manipulation vulnerability in its price feed mechanism. This could allow attackers to manipulate prices and extract more value than intended.

Additionally, issues with access control, reentrancy protection, and debt management need to be addressed.

The ClixpesaRoscas contract, which manages community-based savings and loan groups, has high-severity issues with missing access control in the repayLoan function and inadequate token transfer validation. There are also medium-severity issues around reentrancy risks, missing events, and insufficient validation.

Both contracts would benefit from improved input validation, better error handling consistency, more comprehensive event emissions, and adding emergency pause mechanisms. Additionally, implementing gas optimizations, better code structuring, and comprehensive documentation would significantly improve both contracts' security, efficiency, and maintainability.

Clixpesa Solutions Ltd team addressed issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. Blockhat auditors advised Clixpesa Solutions Ltd Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications

BLOCKHAT
SECURITY

For a Smart Contract Audit, contact us at contact@blockhat.io