Pashov Audit Group

RWf(x) Security Review

August 20th 2025 - August 22nd 2025

Contents

1. About Pashov Audit Group

Pashov Audit Group consists of 40+ freelance security researchers, who are well proven in the space - most have earned over $100k in public contest rewards, are multi-time champions or have truly excelled in audits with us. We only work with proven and motivated talent.

With over 300 security audits completed — uncovering and helping patch thousands of vulnerabilities — the group strives to create the absolute very best audit journey possible. While 100% security is never possible to guarantee, we do guarantee you our team's best efforts for your project.

Check out our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users
- **Medium** - leads to a moderate material loss of assets in the protocol or moderately harms a group of users
- **Low** - leads to a minor material loss of assets in the protocol or harms a small group of users

Likelihood

- **High** - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost
- **Medium** - only a conditionally incentivized attack vector, but still relatively likely
- **Low** - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive

4. About RWf(x)

RWf(x) is a protocol that uses RWA-backed tokens like fractionalized gold (fGOLD) as collateral to mint stablecoins (fToken) and leveraged tokens (xToken). It enables splitting yield-bearing assets into a stable, yield-backed coin (goldUSD) and a leveraged asset (xGOLD), balancing stability and exposure to volatility.

5. Executive Summary

A time-boxed security review of the **RegnumAurumAcquisitionCorp/fx-contracts** repository was done by Pashov Audit Group, during which **rvierdiiev**, **merlinboii**, **Shaka** engaged to review **RWf(x)**. A total of **2** issues were uncovered.

Protocol Summary	
Project Name	RWf(x)
Protocol Type	RWA Tokenization
Timeline	August 20th 2025 - August 22nd 2025

Review commit hash:
- f6e865df2dd46d67a49391d94e54b26e6a8af43c (RegnumAurumAcquisitionCorp/fx-contracts)

Fixes review commit hash:
- 80c514e163f5f8effaa3ba0c4b25cf658a939434 (RegnumAurumAcquisitionCorp/fx-contracts)

Scope

`FxLowVolatilityMath.sol` `HarvestableTreasury.sol` `Market.sol` `Treasury.sol` `IFxMarket.sol` `IFxTreasury.sol` `IRWAVaultPriceOracle.sol`

6. Findings

Findings count

Severity	Amount
High	1
Medium	1
Total findings	**2**

Summary of findings

ID	Title	Severity	Status
[H-01]	Critical functions revert if system is undercollateralized	High	Resolved
[M-01]	Minting of `fToken` and `xToken` allowed during stability mode	Medium	Resolved

High findings

[H-01] Critical functions revert if system is undercollateralized

Severity

Impact: High

Likelihood: Medium

Description

The internal function `Treasury._loadSwapState()` calculates the `xNav` as follows:

```
_state.xNav =
_state.baseSupply.mul(_state.baseNav).sub(_state.fSupply.mul(_state.fNav)).div(_state.xSupply);
```

This is equivalent to:

```
(baseSupplyNav - fSupplyNav) / xSupply
```

In the case of the system being undercollateralized (`baseSupplyNav < fSupplyNav`), the calculation of `xNav` will revert due to subtraction underflow. All transactions involving the execution of `_loadSwapState()` will fail, including all operations that aim to raise the collateral ratio so that it can return to a healthy state.

As a result, the protocol will lack any mechanism to recover the collateralization ratio once it falls below 100%.

Proof of concept

Add the following code to the `Market.spec.ts` test file.

```
context.only("audit", async () => {
  beforeEach(async () => {
    await oracle.setPrice(100000000000); // $1000 with 8 decimals
    await treasury.initializePrice();
    await weth.deposit({ value: ethers.parseEther("10") });
    await weth.approve(market.getAddress(), MaxUint256);
    await market.mint(ethers.parseEther("1"), deployer.address, 0, 0);
  });

  it("reverts when collateral ratio is below 100%", async () => {
    // The system becomes undercollateralized
    await oracle.setPrice(65000000000); // $650 with 8 decimals

    // Manager tries to raise collateral ratio, but transactions revert
    await expect(market.mintXToken(ethers.parseEther("1"), signer.address, 0))
      .to.revertedWith("SafeMath: subtraction overflow");
    await expect(market.addBaseToken(ethers.parseEther("1"), signer.address, 0))
```

```
        .to.revertedWith("SafeMath: subtraction overflow");
    });
  });
```

Recommendations

```
  function redeem(
(...)
-   _baseOut = _state.redeem(_fTokenIn, _xTokenIn);
+   if (_state.xNav == 0) {
+     require (_xTokenIn == 0, "Undercollateralalized");
+     // only redeem fToken proportionally when under collateral.
+     _baseOut = _fTokenIn.mul( _state.baseSupply).div(_state.fSupply);
+   } else {
+     _baseOut = _state.redeem(_fTokenIn, _xTokenIn);
+   }

(...)

    if (_state.xSupply == 0) {
        // no xToken, treat the nav of xToken as 1.0
        _state.xNav = PRECISION;
    } else {
-       _state.xNav =
_state.baseSupply.mul(_state.baseNav).sub(_state.fSupply.mul(_state.fNav)).div(_state.xSupply);
+       uint256 baseSupplyNav = _state.baseSupply.mul(_state.baseNav);
+       uint256 fSupplyNav = _state.fSupply.mul(_state.fNav);
+       if (baseSupplyNav <= fSupplyNav) {
+           _state.xNav = 0;
+       } else {
+           _state.xNav = baseSupplyNav.sub(fSupplyNav).div(_state.xSupply);
+       }
    }
```

Medium findings

[M-01] Minting of `fToken` and `xToken` allowed during stability mode

Severity

Impact: Medium

Likelihood: Medium

Description

The `Market.mint()` function mints both fToken and xToken based on the current collateral ratio.
In the original Aladdin implementation, this function could be called only once. However, RegnumFx removed this restriction, allowing it to be called multiple times.

When the system enters stability mode, the collateral ratio has fallen below the defined safe threshold. This indicates that additional base tokens need to be deposited to restore the ratio.

Allowing `mint()` during stability mode worsens the problem: each new mint increases the number of fTokens in circulation, which in turn raises the amount of base tokens required to bring the system back to a healthy state. As a result, recovery becomes more difficult, and the system may remain undercollateralized for longer.

The severity chosen for this issue is medium, because only whitelisted managers can use the function, and they are trusted entities that are not interested in making stablecoin depeg.

Recommendations

Restrict `mint()` from being called when the system is in stability mode to prevent further dilution of collateralization and to simplify recovery.