Security Assessment

Sweep Protocol

Audit Summary Report

Jul 13, 2023

Table of contents

Introduction to Sublime Group:

Sublime Group is a leading organization revolutionizing the financial landscape through its expertise in Decentralized Finance (DeFi), quantitative trading, market making, and technical security assessment. With a commitment to innovation and trust, Sublime Group offers cutting-edge solutions and advisory services in the rapidly evolving digital asset ecosystem.

Technical Security Assessment and Advisory Services:

Sublime Group excels in providing cutting-edge technical security assessment and advisory services. Leveraging advanced tools and methodologies, we conduct comprehensive security assessments, penetration testing, and vulnerability analysis to fortify systems and infrastructure. Our team of highly skilled cybersecurity experts possesses extensive expertise in identifying and mitigating potential risks and ensuring compliance with regulatory frameworks. By partnering with Sublime Group, clients benefit from our industry-leading security solutions, enabling them to safeguard their digital assets and maintain a robust security posture that surpasses competitors.

Smart Contract Auditing:

At Sublime Group, we pride ourselves on delivering meticulous smart contract audits that go beyond industry standards. Our experienced auditors combine their deep understanding of blockchain technology with an arsenal of cutting-edge tools and advanced methodologies. In addition to manual code analysis, our auditors utilize static code analysis tools to perform automated checks, machine learning techniques to identify complex vulnerabilities, and comprehensive fuzz testing to detect potential security loopholes. By employing these advanced methods, we thoroughly review the reliability, functionality, and security aspects of smart contract code. Our auditors provide actionable recommendations to address identified vulnerabilities, ensuring the robustness and integrity of our clients' blockchain-based applications. Sublime Group's smart contract auditing process, powered by state-of-the-art LLM and fuzzing tools, puts Sublime ahead of the industry standard. By choosing Sublime Group, clients benefit from our unrivaled expertise and innovative approach, guaranteeing the utmost security for their smart contracts and staying ahead of the curve in the ever-evolving landscape of software security.

Decentralized Finance (DeFi):

Sublime Group leverages decentralized protocols and smart contracts to provide seamless access to decentralized lending, borrowing, yield farming, and decentralized exchanges. Our solutions bridge traditional finance with the blockchain ecosystem, empowering users in the DeFi space.

Quantitative Trading:

Sublime Group's skilled quantitative traders optimize trading strategies using advanced algorithms and data analysis techniques. Our expertise in market dynamics, liquidity, and risk management enables efficient execution and enhanced trading performance across digital asset markets.

Market Making:

Sublime Group ensures market liquidity, reduces spreads, and minimizes price volatility through our market-making services. Our proprietary algorithms and risk management systems contribute to fair and efficient price discovery, benefiting institutional and retail investors.

Audit Test and Reporting Disclaimer:

Sublime Group conducted activities for this project according to the statement of work. However, it is important to note that security assessments have time constraints and rely on client-provided information. Thus, the findings in this report may not encompass all security issues or flaws in the system or codebase.

Sublime Group employs automated testing techniques and manual security reviews to assess software controls. However, automated tools have limitations, such as not capturing all edge cases or incomplete analysis within time limits. These limitations are subject to project time and resource constraints.

Clients should understand that while Sublime Group's test coverage is comprehensive within the project's scope, it may not uncover all potential vulnerabilities or flaws. The audit scope does not cover code provided by third party libraries or protocols that Sweep Protocol integrates with. However, we have made due diligence when conducting our test to check for any known vulnerabilities and correct usage of said libraries. Ongoing security assessments and proactive measures are advised to maintain system integrity. Sublime Group remains dedicated to assisting clients in enhancing their security and providing expert guidance throughout the development process.

Audit summary

Sweep Protocol engaged Sublime Group to review the security of its contracts. Our dedicated team has invested 2.5 person-weeks of effort into conducting a security review of the client-provided source code. We have conducted a thorough audit of the smart contracts code provided by Sweep Protocol.

Before starting the audit Sweep initiated a code-freeze and created an audit branch with all the relevant smart contracts codebase. The audit commenced with revision a5c0f31b0cf72f251e32554e6925b82ca41cc22d on the audit branch, which served as the starting point for our evaluation.

Throughout the audit process, our team has diligently reviewed the Sweep Protocol smart contracts, carefully scrutinizing the codebase for potential security vulnerabilities, logical flaws, and adherence to best practices. Our objective was to assess the robustness and integrity of the contracts, identifying any potential issues that could compromise the security and functionality of the system.

As part of the audit work, we engaged in a collaborative effort with Sweep Protocol, ensuring a smooth and effective communication channel between our team and theirs. This allowed for the timely resolution of any issues or concerns that arose during the audit. Sweep Protocol actively participated in addressing the identified issues, promptly implementing fixes and enhancements based on our recommendations.

The audit process involved a comprehensive analysis of the codebase, including but not limited to the smart contracts and associated libraries or dependencies. Our experienced auditors meticulously reviewed the code, examining key aspects such as contract architecture, data handling, access controls, input validation, and adherence to industry best practices.

During the audit, we uncovered several issues, which were promptly communicated to the Sweep Protocol team. We provided detailed reports outlining the identified vulnerabilities and flaws, along with recommended remediation measures. Sweep Protocol demonstrated a strong commitment to security and swiftly addressed the identified issues by

implementing the recommended fixes. This collaborative approach between our team and Sweep Protocol ensured a robust and secure smart contract implementation.

It is important to note that the audit conducted on Sweep smart contracts was focused solely on the code provided by Sweep Protocol. While we diligently assessed the code for potential vulnerabilities, logical flaws, and adherence to best practices, it is essential to acknowledge that unforeseen risks or vulnerabilities may exist beyond the scope of the audit.

Sweep Protocol's proactive engagement during the audit process, including addressing the identified issues, reflects their dedication to ensuring the security and reliability of their smart contracts. By actively participating in the audit and promptly resolving any identified issues, Sweep Protocol showcases a commitment to delivering a secure and trustworthy platform for their users.

In conclusion, the audit conducted on the Sweep Protocol smart contracts involved a comprehensive review of the codebase, analysis of potential vulnerabilities, and collaboration with the Sweep Protocol team to address and rectify identified issues. The proactive engagement from Sweep Protocol in fixing the identified issues demonstrates their commitment to maintaining a secure and robust smart contract implementation.

The contracts were updated by Sweep Protocol multiple times during the audit and re-checked by Sublime Group on later revision numbers. The list of all revisions checked during the audit can be found below:

1. a5c0f31b0cf72f251e32554e6925b82ca41cc22d

Below is a list of smart contracts included in the audit scope along with their respective sha256 checksums of the latest revision tested:

Filename	SHA-256
BaseSweep.sol	ad900fbd4001e3141815e47126253a474d04b16dd6085047422f9435b8d82c03
Sweep.sol	706657547f0764d733f8495af631bf370967475b1a23177bd6eb22be6d4ab720

Balancer.sol	fc69e74c639b3b8c9838404b1d68f1770f8efd2a168615db38fa8512ad9804ff
Stabilizer.sol	0f5a54df34f50d6c3ea38754ff92a822d8faa6c8466259dc8102b5e27f3bfdd3
UniswapAMM.sol	14b2550a830059ea4bd52b4d272b2a7951d1d4362bb3a2c4c28bed61b7a82faf
OffChainAsset.sol	deae89d084a57fd31be7bb3104a98a8e4297502c1d88a0b3704e9fff5591a8cb
UniV3Asset.sol	fcfe1ee8f4e5d2c310768b6ddc29694d9b96621a047a8d24412b25cba5f96764
TokenAsset.sol	ac28875c88f88ce0cdbe6004b5aea6add95a50b323c6ead0f3e0fcbf4983990e
AaveV3Asset.sol	3d9248bca916ff2c2d15f817eabe68d88542cf922cae93f81514606d889754bd
USDPlusAsset.sol	ee181df3a03eb4e76c04423a202af535e8758688647dde268cb372700d94cb2a
BakedAsset.sol	3b73944aa64b1c5acadb274b4f2024068ae69d6032ebe352ce85946ec0355639
GlpAsset.sol	aec4768d5f5d3082736bfcd72929f77b8cb76ea941c41a5a76f2ce5155aee576
GDAIAsset.sol	d96531e5b211239c890a77a3d52937c391351fa58cbdfc7cf608e4ddae375684
CompV2Asset.sol	c8074068ffba4c0e077e97c944be2a9f0b779389edb877e0fcc1cbca06cb2d30
MarketMaker.sol	14b978a368bc4afeee50e70d36035bca3c44b20e51a95dfcc1bb0d874920b009

Sweepr.sol	95e2c4458e01f061eccfbe44f7e45fdee233d6401beeef0e579ee66ee152c813
Governance.sol	b69156223659f9eb745cc20eb53edb2728dfae8c4c610d4964e29e29b9f7dcf5
Treasury.sol	0cf1daa6cdcc5a2726e97894858044f137fd9a7ac4327a304dc9183bc1cf53c5
TokenDistributor.sol	28ea11ad19ef63197b085c7f0c6c84c409192f0cf011696dd21fa20ef08942ea
Owned.sol	f584bbf9bd9066a58da767d6afc0e57ede9b47425ec7890e5b832ec510a4181a
ChainlinkPricer.sol	76e5da9bec60c5d4afb43ed7bf9742df91f495f411c22811ccdf3c6dbfe90428
LiquidityHelper.sol	e6e93b322834a9a06a4cf880cd19a8a51a4bffa467be6e5cc4c53fcff27cc2bd
TransferApproverBlacklist.sol	44f455afc07e67bb46e95d31e460d0247a9c0b3de5eae8a1f260be7e36e4c738
TransferApproverWhitelist.sol	e284a607793d71782c992208f6361c3ea7d43a4e4e2f6e86f48b45a7a97eb7a6

The repository included contracts copied from LayerZero Omnichain, we have analyzed that they were not changed in any significant way. We have treated them as any other library during this audit..

Findings Summary

Our primary focus during the audit was to identify potential vulnerabilities in the Sweep Protocol smart contracts, specifically related to business logic, arithmetic operations, and integration with other protocols and tokens. Our objective was to ensure that the protocol remains secure, preventing any unauthorized access or fund theft. Additionally, we have provided Sweep Protocol with guidance on clean code practices and industry best practices wherever applicable.

Throughout the audit process, we diligently analyzed the smart contracts codebase, resulting in the identification of several issues with varying severity levels. We provided Sweep Protocol with comprehensive guidance and recommendations on security enhancements, best practices, and maintaining clean code standards. Collaboratively, we worked with Sweep Protocol to address and rectify all the issues discovered during the audit.

We have evaluated the overall security of the system to be satisfactory, nevertheless we did identify a range of security issues, with severity levels ranging from critical to low. This included issues that could negatively affect user funds. It is worth noting that, although user funds were potentially at risk, the nature of the issues required certain preconditions to be met and the architecture of the system limited the risk to a finite subset of the TVL. We have worked together with Sweep to ensure that all the issues are understood and resolved. We have also advised the team on how to implement best practices in order to avoid such vulnerabilities in the future.

We have identified a few key areas for the security of the protocol such as the dependency on LayerZero omnichain and the numerous stabilizer contracts. We address these areas in the general recommendations section of the report.

The test coverage of the protocol was determined to be good. However, we recommend adding additional tests to improve test coverage in certain areas. A detailed section in the report outlines all the issues identified within the tests.

Additionally, we have found the protocol documentation to be satisfactory. We have found a few missing items in the documentation and advised Sweep Protocol to update and expand the documentation in these areas.

In conclusion, the audit process has uncovered several issues of varying severity within the Sweep Protocol smart contracts codebase. However, we are pleased to report that Sweep has diligently resolved all of the reported issues and that the overall security of the system is satisfactory, with no critical issues threatening user funds. We have collaborated closely with Sweep Protocol to address and resolve the identified issues, reinforcing their commitment to maintaining a secure protocol.

Below is a summary of all the issues found during the audit divided into respective severity categories ranging from critical to suggestion.

Severity	Number of issues	Remaining after Audit
Critical	1	-
High	3	-
Medium	1	-
Low	7	-
Recommendation	3	-

List of Issues Found:

ISSUE-1 | Misleading minter address validation

Severity	Revision found	Status
Low	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved by Sweep

Description:

Sweep and BaseSweep contract contains a minterMint function that takes a “minter” (Address of a minter) parameter, however only msg.sender is validated as a validMinter, the parameter "minter" is simply treated as a mintTo address. This can be misleading and lead to improper minter validation.

Recommendation:

Even though minters are currently properly validated the implementation can be misleading and in a worst case scenario lead to minting sweep to a wrong address in the future. We recommend removing the parameter or properly validating the address..

```
function minterMint(
   address minter,
   uint256 amount
) public override validMinter(msg.sender) whenNotPaused {
   if (address(amm) != address(0) && !isMintingAllowed())
       revert MintNotAllowed();

   super.minterMint(minter, amount);
}
```

ISSUE-2 | Missing sender validation in ERC721Received callback

Severity	Revision found	Status
Critical	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved by Sweep

Description:

In UniV3Asset.sol there is an external onERC721Received callback that does not validate the msg.sender is as expected the nonfungiblePositionManager. In case there are no deposits yet (tokenId_ is 0) anyone can call the callback and set an arbitrary tokenId_. This in turn can lead to liquidity being added for the wrong id and a loss of funds the next time invest is called on the contract.

Recommendation:

Msg.sender should be properly validated inside the onERC721Received callback.

```
function onERC721Received(
    address,
    address,
    uint256 tokenId_,
    bytes calldata
) external override returns (bytes4) {
    if (tokenId > 0) revert AlreadyMinted();
    _createDeposit(tokenId_);

    return this.onERC721Received.selector;
}
```

ISSUE-3 | Misleading name of access control modifier

Severity	Revision found	Status
Recommendation	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved By Sweep

Description:

Modifier onlyMultisig on Sweep accepts both Multisig and owner addresses. The name could be changed to onlyMultisigOrOwner to better reflect the actual logic.

Recommendation:

Although it's highly unlikely that this can lead to any security issue it would be best to rename the modifier to onlyMultisigOrOwner to better reflect the actual logic.

```
modifier onlyMultisig() {
    if (msg.sender != owner() && msg.sender != fastMultisig)
        revert NotMultisig();
    _;
}
```

ISSUE-4 | Infinite slippage swap in stabilizers.

Severity	Revision found	Status
High	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Partially Resolved

Description:

TokenAsset.sol and GDAIAsset.sol contracts (stabilizers) contain a flaw in their invest and divest function implementation. Both contracts invoke swapExactInput in the invest and divest functions. However, amountOutMinimum as well as sqrtPriceLimitX96 is set to 0 in these calls. This could expose the protocol to front running, sandwiching or another type of price manipulation attacks

Recommendation:

We suggest using proper values for slippage protection in production code to avoid issues mentioned above. One solution would be to set these values based on trusted expected oracle price and configurable max slippage.

```
function _invest(uint256 usdxAmount, uint256) internal override {
   uint256 usdxBalance = usdx.balanceOf(address(this));
   if (usdxBalance < usdxAmount) usdxAmount = usdxBalance;

   TransferHelper.safeApprove(address(usdx), SWEEP.amm(), usdxAmount);
   uint256 investedAmount = amm().swapExactInput(address(usdx), address(token), usdxAmount,
0);

   emit Invested(investedAmount, 0);
}
```

```
function swapExactInput(
   address tokenA,
   address tokenB,
   uint256 amountIn,
   uint256 amountOutMin
) public returns (uint256 amountOut) {
   // Approval
   TransferHelper.safeTransferFrom(
       tokenA,
       msg.sender,
       address(this),
       amountIn
   );
```

```
    TransferHelper.safeApprove(tokenA, address(ROUTER), amountIn);

    ISwapRouter.ExactInputSingleParams memory swapParams = ISwapRouter
        .ExactInputSingleParams({
            tokenIn: tokenA,
            tokenOut: tokenB,
            fee: poolFee,
            recipient: msg.sender,
            // TODO: will this hardcoded 200 work for every network?
            deadline: block.timestamp + DEADLINE_GAP,
            amountIn: amountIn,
            amountOutMinimum: amountOutMin,
            sqrtPriceLimitX96: 0
        });

    amountOut = ROUTER.exactInputSingle(swapParams);
}
```

ISSUE-5 | No slippage protection in GlpAsset

Severity	Revision found	Status
Low	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Partially Resolved

Description:

GlpAsset.sol calls mintAndStakeGlp/unstakeAndRedeemGlp in its invest and divest functions. In both cases 0 is supplied as the minUsdg and minGlp value. This could expose the protocol to attacks directed at minting the asset at an unfavorable price.

Recommendation:

Current implementation of Glp reward router uses an oracle internally and has atomic attacks (e.g. front running) mitigations in place. That being said it would be a good idea to supply the min values thus reducing the trust put into the third party project.

```
function _invest(uint256 usdxAmount, uint256) internal override {
   uint256 usdxBalance = usdx.balanceOf(address(this));
   if(usdxBalance < usdxAmount) usdxAmount = usdxBalance;

   TransferHelper.safeApprove(
       address(usdx),
       address(glpManager),
       usdxAmount
   );
   rewardRouter.mintAndStakeGlp(address(usdx), usdxAmount, 0, 0);

   emit Invested(usdxAmount, 0);
}
```

ISSUE-6 | No slippage protection in UniV3Asset

Severity	Revision found	Status
High	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Partially Resolved

Description

UniV3Asset.sol calls increaseLiquidity and decreaseLiquidity on Uniswap V3 nonfungiblePositionManager in its invest and divest functions. In both cases 0 is supplied as the amount0Min and amount1Min value. This could potentially expose the protocol to frontrunning attacks designed to execute the mint call at an inaccurate price.

Recommendation:

We suggest using proper values for slippage protection in production code to avoid issues mentioned above.

```
if (tokenId == 0) {
    (, liquidity_, amount0, amount1) = _mint(amountAdd0, amountAdd1);
} else {
    (liquidity_, amount0, amount1) = nonfungiblePositionManager
        .increaseLiquidity(
            INonfungiblePositionManager.IncreaseLiquidityParams({
                tokenId: tokenId,
                amount0Desired: amountAdd0,
                amount1Desired: amountAdd1,
                amount0Min: 0,
                amount1Min: 0,
                deadline: block.timestamp + 60 // Expiration: 1 hour from now
            })
        );
    liquidity += liquidity_;
}
```

ISSUE-7 | MarketMaker single sided liquidity min amount

Severity	Revision found	Status
low	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved by Sweep

Description:

Similar to the UniV3Asset issue in MarketMaker addSingleLiquidity function no amount0Min and amount1Min values are provided. In this instance the price of the asset is known at call time and only one of the assets is provided in the position so there shouldn't be any risk of a frontrunning attack.

Recommendation:

Nevertheless, since the protocol expects a known amount of USDx token to be added as liquidity we recommend to set that value as the amountMin parameter.

```
function addSingleLiquidity(
   uint256 minPrice,
   uint256 maxPrice,
   uint256 usdxAmount,
   uint24 poolFee
) internal {

(...)

   TransferHelper.safeApprove(
       address(usdx),
       address(nonfungiblePositionManager),
       usdxAmount
   );

(...)

   nonfungiblePositionManager.mint(
           INonfungiblePositionManager.MintParams({
               token0: token0,
               token1: token1,
               fee: poolFee,
               tickLower: minTick,
               tickUpper: maxTick,
               amount0Desired: amount0Mint,
               amount1Desired: amount1Mint,
               amount0Min: 0,
               amount1Min: 0,
               recipient: address(this),
               deadline: block.timestamp
           })
       );
```

ISSUE-8 | Invested & Divested events consistency

Severity	Revision found	Status
Recommendation	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved by Sweep

Description:

Invested/Divested events lack consistency in supplied parameter values. Depending on the asset contract the supplied value to the Invest event can be the usdx amount “in” or the token amount “out”. Similarly, for Divest sometimes it's usdx “out”, sometimes it's the token “in” amount. For example, in TokenAsset the Invested event's usdxAmount field is supplied with the actual "out" token amount. For GlpAsset the Invest event’s usdxAmount is actually supplied with "in" usdx and "out" amount is ignored, on the other hand the Divest event’s usdxAmount param is supplied with the glpAmonut sold.

Recommendation:

A closer look should be taken at the values supplied to the Invest and Divest events. Most importantly the values should be denominated in the same token as Events definition suggests. There should also be consistency in whether the amount “in” or “out” is logged in the event.

Event definition (usdxAmount):

```
event Invested(uint256 indexed usdxAmount, uint256 indexed sweepAmount);
event Divested(uint256 indexed usdxAmount, uint256 indexed sweepAmount);
```

GLP amount in GlpAsset:

```
function _divest(uint256 usdxAmount) internal override {

    (...)

    emit Divested(glpAmount, 0);
}
```

Token amount in TokenAsset:

```
function _invest(uint256 usdxAmount, uint256) internal override {

    (...)

    uint256 investedAmount = amm().swapExactInput(address(usdx), address(token), usdxAmount,
0);

    emit Invested(investedAmount, 0);
}
```

USDX amount in TokenAsset:

```
function _divest(uint256 usdxAmount) internal override {

    (...)

    TransferHelper.safeApprove(address(token), SWEEP.amm(), tokenAmount);
    uint256 divested = amm().swapExactInput(
        address(token),
        address(usdx),
        tokenAmount,
        0
    );

    emit Divested(divested, 0);
}
```

ISSUE-9 | Governance delay values should be adjusted

Severity	Revision found	Status
Medium	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved by Sweep

Description:

At the time of conducting the audit the delay times configured for the Governor module were deemed to be unsafe. Proposal delay was set to "1" and voting period to "300". Low values could increase the risk of malicious proposals slipping through.

Recommendation:

While high quorum setting definitely lowers the chance of a malicious proposal slipping through, we think that safer values should be set for the delay times of the Governance module. A new proposal should have at least a total delay of 3-7 days before it's voted in.

ISSUE-10 | Bad equity ratio validation upon withdraw

Severity	Revision found	Status
High	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved by Sweep

Description:

We have identified a potential issue with equity ratio calculation in the withdraw function of a stabilizer. In case the min_eq_ratio is set to 0, the borrower can simply borrow a maximum amount of sweep allowed for the stabilizer, he can then proceed to sell it to usdx and then withdraw everything, leaving the protocol at a loss of the borrowed sweep. This happens because when withdrawing everything from a stabilizer the new total_value is calculated as 0 and in such cases the eq_ratio is always calculated as zero as well (due to a fast exit from the function). In such case, when the min_eq ratio is also zero the borrower can withdraw all the borrowed funds from the contract. The actual eq ratio should be negative in this instance and in fact trying to withdraw total_value - 1 will revert as expected, because the fast exit condition will not be satisfied and the eq ratio will be calculated as a negative number.

Recommendation:

We recommend fixing this logic so that the eq. Ratio is calculated properly. Furthermore it would be a good idea to validate the min_eq ratio to be > 0. We haven't classified this issue as critical because it relies on two preconditions. The attacker being a privileged borrower and the min equity ratio being configured to 0 for the stabilizer. However, It's worth noting that we have uncovered one Stabilizer with the ratio configured to 0.

```
function _calculateEquityRatio(
    uint256 sweepDelta,
    uint256 usdDelta
) internal view returns (int256) {
    uint256 currentValue_ = currentValue();
    uint256 sweepDeltaInUsd = SWEEP.convertToUSD(sweepDelta);
    uint256 totalValue = currentValue_ + sweepDeltaInUsd - usdDelta;

    if (totalValue == 0) return 0;

    (...)
```

```
function withdraw(
    address token,
    uint256 amount
) external onlyBorrower whenNotPaused validAmount(amount) {
    if (token != sweepAddress && token != address(usdx))
        revert InvalidToken();

    if (amount > IERC20Metadata(token).balanceOf(address(this)))
        revert NotEnoughBalance();

    if (sweepBorrowed > 0) {
        uint256 usdAmount = token == sweepAddress ?
            SWEEP.convertToUSD(amount) : amm().tokenToUSD(amount);
        int256 currentEquityRatio = _calculateEquityRatio(0, usdAmount);
        if (currentEquityRatio < minEquityRatio)
            revert EquityRatioExcessed();
    }

    TransferHelper.safeTransfer(token, msg.sender, amount);

    emit Withdrawn(token, amount);
}
```

ISSUE-11 | Stabilizer liquidate function operations order

Severity	Revision found	Status
Low	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved by Sweep

Description:

The liquidate function can be adjusted a bit to better follow the checks, effects and interactions pattern to avoid any potential for reentrancy and cross-function reentrancy. The function involves transferring out the tokens to the function caller (a liquidator). The function is external and has no access control. Even though the tokens used in liquidation should be trusted it's possible that down the line a token can slip through that allows for some form of reentrancy. The function can be adjusted to transfer in only the missing sweep token needed to liquidate and then transfer out all of the remaining assets as the last call of the function.

Recommendation:

We recommend adjusting the order of operations in the function. Even though the risk of reentrancy is low.

```
function _liquidate(address token) internal {
    if (!isDefaulted()) revert NotDefaulted();
    address self = address(this);

    uint256 sweepToLiquidate = getLiquidationValue();
    (uint256 usdxBalance, uint256 sweepBalance) = _balances();
    uint256 tokenBalance = IERC20Metadata(token).balanceOf(self);
    // Gives all the assets to the liquidator first
    TransferHelper.safeTransfer(sweepAddress, msg.sender, sweepBalance);
    TransferHelper.safeTransfer(address(usdx), msg.sender, usdxBalance);
    TransferHelper.safeTransfer(token, msg.sender, tokenBalance);

    // Takes SWEEP from the liquidator and repays as much debt as it can
    TransferHelper.safeTransferFrom(
        sweepAddress,
        msg.sender,
        self,
        sweepToLiquidate
    );

    _repay(sweepToLiquidate);

    emit Liquidated(msg.sender);
}
```

ISSUE-12 | Missing zero address check

Severity	Revision found	Status
Low	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Partially Resolved

Description:

OffChainAsset - wallet address and BaseSweep - fastMultiSig address are not validated for zero address upon being set. The check is missing both in contract constructors and respective setter functions.

Recommendation:

It is recommended to check for 0 address accounts where possible. Especially in case of the wallet address an accidental 0 address could lead to a loss of funds.

```
function setWallet(
    address wallet_
) external onlyBorrower onlySettingsEnabled {
    wallet = wallet_;
}
```

ISSUE-13 | ITransferApprover interface is not implemented

Severity	Revision found	Status
Recommendation	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Resolved by Sweep

Description:

TransferApproverBlacklist and TransferApproverWhitelist contracts do not actually inherit from the ITransferApprover interface.

Recommendation:

We recommend to make TransferApproverBlacklist and TransferApproverWhitelist inherit from the ITransferApprover interface.

ISSUE-14 | Ignored return values

Severity	Revision found	Status
Low	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	Partially Resolved

Description:

In some of the asset contracts (ETSAsset, GDAIAsset, GLPASSET, USDPLUSL, Univ3Asset) the invest/divest functions ignore the return value of the underlying asset "mint/deposit/redeem" function.

Recommendation:

It would be a good idea to do at least a sanity check and test if the actual minted/redeemed amount is not 0 (especially for GDAIAsset), also as mentioned in other issues in this audit - test for slippage where applicable. Optionally the actual minted amount could be logged in an event.

ISSUE-15 | Reentrancy protection

Severity	Revision found	Status
Low	a5c0f31b0cf72f251e32554e6925b82ca41cc22d	???

Description:

Although strictly speaking this is a recommendation, we have decided to mark it as low severity since its impact on security is significant. The Stabilizer contracts contain multiple external functions, most of which require the caller to be a privileged borrower. They interact internally with multiple tokens and external protocols. Although we have not found a clear reentrancy or cross function reentrancy attack vector, we feel that there is a risk that a flaw in one of the underlying protocols or a malicious token can put Sweep at risk of a reentrancy attack. We acknowledge that this would require the Sweep Team approving a Stabilizer based on a vulnerable token or a flaw in an underlying protocol or a malicious borrower. Nevertheless we feel that as the protocol grows the risk justifies adding reentrancy protection on Stabilizers external functions.

Recommendation:

We recommend implementing reentrancy protection on Stabilizers external functions, excluding functions requiring admin privilege.

Test coverage:

The protocol codebase contained tests that covered most of the protocol functionality. The test coverage was deemed commendable, however during the audit we have identified that a few areas of the codebase are not tested (with 0% coverage). This was either due to broken tests or tests missing altogether. We recommend fixing this issue and implementing the missing tests.

```
-------------------------------|----------|----------|----------|----------|----------------|
File                           |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-------------------------------|----------|----------|----------|----------|----------------|
 AMM/                          |      100 |      100 |      100 |      100 |                |
  IAMM.sol                     |      100 |      100 |      100 |      100 |                |
  UniswapAMM.sol               |      100 |      100 |      100 |      100 |                |
 Assets/                       |    62.18 |    48.78 |      100 |     63.5 |                |
  AaveV3Asset.sol              |      100 |      100 |      100 |      100 |                |
  BackedAsset.sol              |        0 |        0 |      100 |        0 |... 134,136,138 |
  CompV2Asset.sol              |        0 |        0 |      100 |        0 |... 161,163,165 |
  ETSAsset.sol                 |        0 |        0 |      100 |        0 |... 136,138,140 |
  GDAIAsset.sol                |    92.45 |       50 |      100 |       94 |      59,60,178 |
  GlpAsset.sol                 |    86.49 |       50 |      100 |    88.57 |  63,64,133,135 |
  OffChainAsset.sol            |      100 |      100 |      100 |      100 |                |
  TokenAsset.sol               |      100 |       75 |      100 |      100 |                |
  USDPlusAsset.sol             |        0 |        0 |      100 |        0 |... 115,117,119 |
  UniV3Asset.sol               |    85.48 |       65 |      100 |    89.09 |... 115,116,118 |
 Assets/Aave/                  |      100 |      100 |      100 |      100 |                |
  IAaveV3Pool.sol              |      100 |      100 |      100 |      100 |                |
 Assets/Compound/              |      100 |      100 |      100 |      100 |                |
  ICompComptroller.sol         |      100 |      100 |      100 |      100 |                |
  IcUSDC.sol                   |      100 |      100 |      100 |      100 |                |
 Assets/GDAI/                  |      100 |      100 |      100 |      100 |                |
  IGToken.sol                  |      100 |      100 |      100 |      100 |                |
  IOpenTradesPnlFeed.sol       |      100 |      100 |      100 |      100 |                |
 Assets/GMX/                   |      100 |      100 |      100 |      100 |                |
  IGlpManager.sol              |      100 |      100 |      100 |      100 |                |
  IRewardRouter.sol            |      100 |      100 |      100 |      100 |                |
  IRewardTracker.sol           |      100 |      100 |      100 |      100 |                |
 Assets/Overnight/             |      100 |      100 |      100 |      100 |                |
  IExchanger.sol               |      100 |      100 |      100 |      100 |                |
  IHedgeExchanger.sol          |      100 |      100 |      100 |      100 |                |
 Balancer/                     |    92.19 |      100 |      100 |    91.97 |                |
  Balancer.sol                 |      100 |      100 |      100 |      100 |                |
  MarketMaker.sol              |    85.51 |      100 |      100 |    84.93 |... 214,257,318 |
 Governance/                   |    61.22 |    42.86 |      100 |    56.25 |                |
  Governance.sol               |    69.23 |        0 |      100 |    57.14 |... 28,39,48,57 |
  Sweepr.sol                   |    58.33 |       50 |      100 |    58.33 | 37,39,43,45,49 |
  TokenDistributor.sol         |    64.29 |       50 |      100 |    64.29 | 34,43,44,46,48 |
  Treasury.sol                 |       50 |       50 |      100 |     37.5 | 38,39,41,43,53 |
 Stabilizer/                   |    98.29 |    94.29 |      100 |    94.48 |                |
  IStabilizer.sol              |      100 |      100 |      100 |      100 |                |
  Stabilizer.sol               |    98.29 |    94.29 |      100 |    94.48 |... 459,461,465 |
 Sweep/                        |    88.89 |    84.09 |      100 |    85.15 |                |
  BaseSweep.sol                |    93.88 |    85.71 |      100 |    89.47 |... 2,63,67,303 |
  ISweep.sol                   |      100 |      100 |      100 |      100 |                |
  Sweep.sol                    |    82.93 |    81.25 |      100 |    79.55 |... 126,143,151 |
 Sweep/TransferApprover/       |     37.5 |        0 |      100 |    28.57 |                |
  ITransferApprover.sol        |      100 |      100 |      100 |      100 |                |
  TransferApproverBlacklist.sol |     37.5 |        0 |      100 |    28.57 | 31,33,41,49,51 |
  TransferApproverWhitelist.sol |     37.5 |        0 |      100 |    28.57 | 31,33,41,49,51 |
-------------------------------|----------|----------|----------|----------|----------------|
All files                      |    77.94 |    73.58 |      100 |     77.7 |                |
-------------------------------|----------|----------|----------|----------|----------------|
```

Static analysis

During our audit of the Sweep Protocol smart contracts, we conducted a thorough static code analysis using a set of our proprietary rules. This analysis aimed to identify any known bugs or recurring attack vectors. The results revealed that all issues detected were false positives, indicating the absence of actual vulnerabilities or bugs. Our automated examination applied specific rules designed to detect common coding mistakes, security vulnerabilities, and attack patterns. We carefully verified each reported issue and determined that they did not pose real security risks. This static analysis provides additional assurance of the codebase's thorough examination and reinforces confidence in the security and reliability of the Sweep Protocol smart contracts. For more detailed information, please refer to the comprehensive report provided. In conclusion, the static code analysis confirmed the absence of genuine vulnerabilities or bugs, further enhancing the security assessment's credibility.

```
Scanning 48 files with 42 solidity rules.
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48/48 tasks 0:00:00

┌─────────┐
│ Results │
└─────────┘

Findings:

  /sweepr-contracts/contracts/Assets/GlpAsset.sol
         Price oracle can be manipulated via flashloan, price multiplied by some potentially
         manipulatable factor

         176┆ return (price * 10 ** usdx.decimals()) / glpManager.PRICE_PRECISION();


  /sweepr-contracts/contracts/Assets/OffChainAsset.sol
         Unprotected token argument possible reentrancy

         143┆ function payback(address token, uint256 amount) external {


  /sweepr-contracts/contracts/Balancer/Balancer.sol
         Price oracle can be manipulated via flashloan, price multiplied by some potentially
         manipulatable factor

         106┆ return targetPrice * uint256(priceUnit) / (10 ** SWEEP.decimals());


  /sweepr-contracts/contracts/Governance/TokenDistributor.sol
         Unprotected token argument possible reentrancy

         76┆ function recover(
         77┆       address tokenAddress,
         78┆       uint256 tokenAmount
         79┆ ) external onlyGov {
```

```
  /sweepr-contracts/contracts/Governance/Treasury.sol
        Unprotected token argument possible reentrancy

        52¦ function sendToken(address token, address receiver, uint256 amount) external onlyGov {



  /sweepr-contracts/contracts/Stabilizer/Stabilizer.sol
        Unprotected token argument possible reentrancy

        556¦ function withdraw(
        557¦     address token,
        558¦     uint256 amount
        559¦ ) external onlyBorrower whenNotPaused validAmount(amount) {

┌──────────────┐

│ Scan Summary │

└──────────────┘

Some files were skipped or only partially analyzed.
  Scan was limited to files tracked by git.

Ran 42 rules on 48 files: 6 findings.
```

Slither analysis

Although it is not a key offering of Sublime and its usefulness is limited, we have conducted static analysis of Sweep Protocol smart contracts using the industry standard Slither tool. Despite its limitations, it can be useful in identifying areas otherwise missed by the auditors. Below you can find a summary of the analysis.

We have analyzed slithers report and concluded that only 4 out of the around 900 issues reported are true positive and 2 of them were previously identified in our report.

After further analysis we have determined that all of the issues cas be classified as low severity, the two new issues were added to our report.

```
Total number of contracts in source files: 41
Number of contracts in dependencies: 100
Number of contracts in tests       : 7
Source lines of code (SLOC) in source files: 3127
Source lines of code (SLOC) in dependencies: 5779
Source lines of code (SLOC) in tests       : 514
Number of  assembly lines: 0
Number of optimization issues: 11
Number of informational issues: 647
Number of low issues: 204
Number of medium issues: 78
Number of high issues: 9

Use: Openzeppelin-Ownable, Openzeppelin-ERC20
ERCs: ERC2612, ERC721, ERC20, ERC165

+----------------------------+-------------+---------------------+-------------------+-------------+--------------------+
|            Name            | # functions |         ERCS        |     ERC20 info    | Complex code |      Features      |
+----------------------------+-------------+---------------------+-------------------+-------------+--------------------+
|             IAMM           |      9      |                     |                   |      No     |                    |
|          UniswapAMM        |      9      |                     |                   |      No     | Tokens interaction |
|            IPool           |      2      |                     |                   |      No     |                    |
|         AaveV3Asset        |      52     |                     |                   |      No     | Tokens interaction |
|         BackedAsset        |      53     |                     |                   |      No     | Tokens interaction |
|         CompV2Asset        |      55     |                     |                   |      No     | Tokens interaction |
|       ICompComptroller     |      4      |                     |                   |      No     |                    |
|            IcUSDC          |      11     |        ERC20        |     ∞ Minting     |      No     |                    |
|                            |             |                     | Approve Race Cond. |             |                    |
|                            |             |                     |                   |             |                    |
|          ETSAsset          |      53     |                     |                   |      No     | Tokens interaction |
|          IGToken           |      13     |                     |                   |      No     |                    |
|      IOpenTradesPnlFeed    |      1      |                     |                   |      No     |                    |
|         GDAIAsset          |      56     |                     |                   |      No     | Tokens interaction |
|         IGlpManager        |      3      |                     |                   |      No     |                    |
|        IRewardRouter       |      5      |                     |                   |      No     |                    |
|        IRewardTracker      |      5      |                     |                   |      No     |                    |
|          GlpAsset          |      55     |                     |                   |      No     | Tokens interaction |
|        OffChainAsset       |      56     |                     |                   |      No     | Tokens interaction |
|          IExchanger        |      4      |                     |                   |      No     |                    |
|       IHedgeExchanger      |      6      |                     |                   |      No     |                    |
|          TokenAsset        |      53     |                     |                   |      No     | Tokens interaction |
|         USDPlusAsset       |      52     |                     |                   |      No     | Tokens interaction |
|          UniV3Asset        |      59     |                     |                   |      No     | Tokens interaction |
|           Balancer         |      14     |                     |                   |      No     | Tokens interaction |
|         MarketMaker        |      56     |                     |                   |      No     | Tokens interaction |
|        SweepGovernor       |     118     |        ERC165       |                   |      No     |    Receive ETH     |
```

SweeprCoin	139	ERC20,ERC165,ERC2612	∞ Minting Approve Race Cond.	No	Send ETH Ecrecover Receive ETH Send ETH Ecrecover Tokens interaction Assembly
TokenDistributor	6			No	Tokens interaction
Treasury	5			No	Receive ETH Send ETH
OmnichainGovernanceExecutor	47			No	Tokens interaction Receive ETH Send ETH Assembly
OmnichainProposalSender	19			No	Receive ETH Send ETH
ChainlinkPricer	2			No	
IStabilizer	5			No	
ISweep	35	ERC20	No Minting Approve Race Cond.	No	
SweepCoin	142	ERC20,ERC165	Pausable No Minting Approve Race Cond.	No	Receive ETH Send ETH Assembly Upgradeable
ITransferApprover	1			No	
TransferApproverBlacklist	16			No	
TransferApproverWhitelist	16			No	
LiquidityHelper	5			No	Tokens interaction

General security recommendations

In order to ensure a continued safety of the protocol we want to address Sweep Protocol with these general security recommendations.

We have identified several key areas of risk in Sweep Protocol that the Team needs to be cautious about in order to guarantee a secure operation of the protocol. First and foremost the dependency on LayerZeros Omnichain Fungible Token standard bring's great opportunity for sweep to exist on multiple chains. However as with every cross chain technology there is an element of risk involved, especially where security is traded off in favor of user experience.

We recommend Sweep Protocol to follow on LayerZeros announcements, updates and security advisories in order to ensure the protocols security. The team should implement in a timely manner any updates and security fixes related to OFT. Sweep should monitor the results of any security audits of LayerZero. We advise conducting an independent audit of LayerZero in the context of Sweep protocol.

The second key area are the Stabilizer contracts. As the protocol grows and more stabilizer instances are added, we recommend Sweep to verify the privileged borrowers in a diligent way. Whenever interacting with new tokens or protocols caution should be taken to verify their integrity and potential vulnerabilities such as any callbacks allowing for reentrancy.

The team should conduct followup security audits whenever adding new Stabilizer implementations or otherwise extending the protocol.

We would like to encourage Sweep to maintain a comprehensive Event Logging and Monitoring: Emit events for important state changes, transactions, or errors to enable effective monitoring and analysis. Implement a monitoring system that can detect and alert on suspicious activities, such as unexpected contract behavior.

We want to encourage Sweep to regularly review and update the dependencies and external libraries used within the codebase. Promptly apply updates to ensure that the protocol remains protected against known vulnerabilities and exploits.

Finally we advise creating an Active Security Response Plan: Develop and maintain a comprehensive incident response plan specifically tailored for sweep protocol.. Clearly

define the steps to be taken in the event of a security incident, including containment, communication, and recovery procedures. Regularly update and test the incident response plan to ensure its effectiveness in handling potential security breaches or vulnerabilities.

Integration with Other Protocols

In the process of auditing the Sweep Protocol contracts, we have identified that the project directly or indirectly interfaces with, or is based upon, a list of existing protocols. These protocols include Aave, Backed Finance, Compound v2, Overnight Finance, gTrade, GMX, Uniswap v3, LayerZero OFT. The integration of multiple protocols within a project brings both opportunities and potential risks. This summary highlights the significance integrating with other protocols has for protocol security and emphasizes the limitations of the audit scope.

Enhancing Interoperability:

Integrating with well-established protocols enables the project to leverage existing functionalities and tap into a broader ecosystem. This interoperability provides numerous benefits, including enhanced liquidity, access to additional services, and the ability to leverage established communities. By integrating with these protocols, the project positions itself to deliver a more comprehensive and seamless experience for its users. At the same time, each integration brings another layer of complexity into the project and makes it more difficult for the team to ensure project security.

Risk Assessment:

During our audit, we diligently analyzed the Sweep Protocol contracts for known issues or common mistakes that may arise when interacting with the aforementioned protocols. Our objective was to identify any potential vulnerabilities or risks within the contracts that could impact the integration. However, it is important to note that our audit scope was specifically limited to the Sweep Protocol contracts and did not encompass a comprehensive verification of the integrated protocols themselves.

Ongoing Risk Management:

To mitigate potential risks associated with protocol integrations, it is imperative for the project to maintain a proactive approach to risk management. This includes staying informed about updates, upgrades, and potential vulnerabilities of the integrated protocols. By actively monitoring the security landscape and engaging in ongoing risk assessments, the project can promptly address any emerging threats or vulnerabilities, safeguarding the interests of its users and the overall integrity of the system.

The integration of the project with other protocols expands its capabilities and potential. While we have analyzed the Sweep Protocol contracts for issues related to these integrated protocols, it is important to reiterate that our audit scope did not encompass the verification of the integrated protocols themselves. Therefore, it is crucial for project stakeholders to undertake independent audits of the integrated protocols to ensure their security and reliability. By adopting a proactive approach to risk management and ongoing assessments, the project can effectively navigate the complexities of integration and deliver a secure and robust experience for its users.