SECURITY REVIEW OF AMBIT FINANCE

Summary

Auditors: 0xWeiss (Marc Weiss), 0xKato

Peer Reviewers: Lucas Martin, Nisedo

Client: Ambit Finance

Report Delivered: 16 February, 2024

About 0xWeiss

0xWeiss is an independent security researcher. In-house auditor/security engineer in *Ambit Finance* and *Tapioca DAO*. Security Researcher at Paladin Blockchain S-ecurity and ASR at Spearbit DAO. Reach out on Twitter @0xWeisss .

Protocol Summary

Ambit is cutting-edge, cross-chain DeFi protocol offering users simple yields on stablecoin deposits, sustainable money market lending, and risk-defined portfolio investment strategies - all within a user-friendly environment.

Protocol Name	Ambit Finance
Language	Solidity
Codebase	https://github.com/ambitfi/ambitfi-contracts
Commit	1d085e88cac086b55948646316ca8c16a13ce1bd
Previous Audits	Yes, 2. Reports with commits: *https://docs.ambit.finance/audits/paladin-nov23.pdf* *https://docs.ambit.finance/audits/guardian-dec23.pdf*

Audit Summary

Ambit Finance engages **0xWeiss** continuously to review the security of its codebase and consult about architectural decisions.

A 3-week time-boxed security assessment was performed.

At the end, there were 16 issues identified.

All findings have been recorded in the following report. Notice that the examined smart contracts are not resistant to internal exploitation.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

Vulnerability Summary

Severity	Total	Pending	Acknowledged	Par. resolved	Resolved
HIGH	**0**	**0**	**0**	**0**	**0**
MEDIUM	**7**	**0**	**4**	**0**	**3**
LOW	**9**	**0**	**5**	**0**	**4**
INF	**0**	**0**	**0**	**0**	**0**

Severity Classification

Severity	Classification
HIGH	Exploitable, causing loss/manipulation of assets or data.
MEDIUM	Risk of future exploits that may or may not impact the smart contract execution.
LOW	Minor code errors that may or may not impact the smart contract execution.
INF	No impact issues. Code improvement

Methodology

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

AUDIT SCOPE

contracts/protocol/core/AssetStorage.sol
contracts/protocol/core/AddressRegistry.sol
contracts/protocol/tokens/TokenVesting.sol
contracts/protocol/faucet/Faucet.sol
contracts/protocol/vault/SnapshotLib.sol
contracts/protocol/faucet/FaucetERC20.sol
contracts/protocol/vault/YieldVault.sol
contracts/protocol/vault/Vault.sol
contracts/protocol/vault/FlashLender.sol
contracts/protocol/market/Market.sol
contracts/protocol/market/DiscountModel.sol
contracts/protocol/market/DynamicInterestRateModel.sol
contracts/protocol/vault/DepositorVault.sol
contracts/protocol/portfolio/PortfolioStorage.sol
contracts/protocol/market/MarketLiquidation.sol
contracts/protocol/market/FixedInterestRateModel.sol
contracts/protocol/portfolio/TokenRewardHooks.sol
contracts/protocol/portfolio/BaseCustodian.sol
contracts/protocol/portfolio/YieldBearingCustodian.sol
contracts/protocol/portfolio/Portfolio.sol
contracts/protocol/portfolio/Custodian.sol
contracts/protocol/portfolio/RewardEpochLib.sol

contracts/protocol/portfolio/CustodianMigrator.sol
contracts/protocol/market/MarketStorage.sol
contracts/protocol/market/Liquidator.sol
contracts/protocol/vault/DepositorVaultStorage.sol
contracts/protocol/vault/DepositorVaultMigrator.sol
contracts/protocol/vault/DepositorVaultToken.sol
contracts/protocol/vault/LinearDistributedYieldVault.sol
contracts/protocol/faucet/FaucetMarketplaceAdapter.sol
contracts/protocol/governance/Treasury.sol
contracts/protocol/governance/Executable.sol
contracts/protocol/governance/Governor.sol
contracts/protocol/oracle/DepositorVaultTokenPriceOracle.sol
contracts/protocol/oracle/FallbackPriceOracle.sol
contracts/protocol/oracle/ChainlinkAggregatorPriceOracle.sol
contracts/protocol/security/AuthorizedAccessControl.sol
contracts/protocol/security/AccessControlList.sol
contracts/protocol/security/AdminAccessControl.sol
contracts/protocol/lens/ProtocolLens.sol
contracts/protocol/marketplace/MarketplacePurchaser.sol
contracts/protocol/loyalty/BoostModule.sol
contracts/protocol/marketplace/SpotMarketMarketplaceAdapter.sol
contracts/protocol/marketplace/MarketplaceVendor.sol
contracts/protocol/lens/AccountLens.sol
contracts/protocol/marketplace/DepositorVaultMarketplaceAdapter.sol

contracts/protocol/utils/Sweepable.sol
contracts/protocol/utils/Migratable.sol
contracts/protocol/loyalty/LoyaltyLib.sol
contracts/protocol/utils/Pausable.sol
contracts/protocol/loyalty/LoyaltyHooks.sol
contracts/protocol/loyalty/Loyalty.sol
contracts/protocol/loyalty/FirstLoanBoostModule.sol
contracts/protocol/loyalty/LoyaltyStorage.sol

Findings and Resolutions

ID	Category	Severity	Status
GLOBAL-M1	Rate manipulation	🟡 **MEDIUM**	Acknowledged
DV-M1	User Loss	🟡 **MEDIUM**	Resolved
DV-M2	Protocol malfunction	🟡 **MEDIUM**	Acknowledged
DV-M3	User Loss	🟡 **MEDIUM**	Resolved
DV-M4	Protocol malfunction	🟡 **MEDIUM**	Resolved
VW-M1	User Loss	🟡 **MEDIUM**	Acknowledged
L-M1	User Loss	🟡 **MEDIUM**	Acknowledged
GLOBAL-L1	Un-used errors	🟢 **LOW**	Resolved
GLOBAL-L2	Naming	🟢 **LOW**	Acknowledged
DV-L1	Missing checks	🟢 **LOW**	Resolved
BC-L1	Natspec	🟢 **LOW**	Resolved
LENS-L1	Incorrect behaviour	🟢 **LOW**	Acknowledged
LL-L1	Natspec	🟢 **LOW**	Resolved
CAPO-L1	Composability	🟢 **LOW**	Acknowledged
CAPO-L	Incorrect behavior	🟢 **LOW**	Acknowledged
FPO-L1	Incorrect behavior	🟢 **LOW**	Acknowledged

[GLOBAL-M1] Protocol is vulnerable when *_distributionWindow* is set to 0.

Severity	Category	Status
MEDIUM	Rate manipulation	Acknowledged

Description

The yield in the depositor vault is distributed in a linear way depending on the *_distributionWindow* specified. This will calculate how much yield to distribute in any block from previous donations to the contract.

If this *_distributionWindow* is set to 0, it can open multiple ways of manipulating balances, and the exchange rate of AUSD from the depositor vault oracle.

```
  uint256 exchangeRate = vault.getExchangeRate();

    uint256 price = exchangeRate.mulDiv(denominatorPrice.normalize(decimal
s), scalar).normalize(decimals, USDMath.DECIMALS);
```

There could be a scenario where you could in theory donate a large amount to increase the price of AUSD, borrow the maximum against it, and then withdraw what you donated.

Recommendation

To not open any of these attack vectors, do never set *_distributionWindow* to 0. Very important to keep in mind also for any possible forks of the codebase in the future.

Resolution

Acknowledged. The distribution window is controlled through admin control, so the risk is mitigated. Additionally, a large donation would increase the price of AUSD, but removing the donation wouldn't decrease the price of AUSD as the donation would be shared amongst all depositors.

[DV-M1] *setDistributionWindow* distributes the yield post being updated.

Severity	Category	Status
MEDIUM	User Loss	Resolved

Description

The yield in the depositor vault is distributed in a linear way depending on the *_distributionWindow* specified. This will calculate how much yield to distribute in any block from previous donations to the contract.

When updating the *_distributionWindow* though, the call to *distribute()* happens after the window is updated affecting on the calculation of previous yield that was supposed to be distributed:

```
function setDistributionWindow(uint256 distributionWindow) virtual public
{
    _distributionWindow = distributionWindow;

    distribute();

    emit SetDistributionWindow(msg.sender, distributionWindow);
  }
```

Recommendation

Move the *distribute()* call before updating the window.

```
function setDistributionWindow(uint256 distributionWindow) virtual public
{
+    distribute();
    _distributionWindow = distributionWindow;

-    distribute();

    emit SetDistributionWindow(msg.sender, distributionWindow);
  }
```

Resolution

Fixed at commit: db960b765c85aab42c4465335071b6a6f81730ff

[DV-M2] Supply caps can be bypassed through donating on the vault.

Severity	Category	Status
MEDIUM	Protocol malfunction	Acknowledged

Description

Currently, AMBIT is using supply caps to control the amount of assets that can be on the vault to have a healthy and secure launch.

```
function setMaxSupply(uint256 maxSupply) external onlyAdmin {
  _maxSupply = maxSupply;
```

This cap is enforced everytime assets are deposited into the vault, in this specfic case, it would be USDT:

```
if (getTotalAssets() + amount > _maxSupply) {
     revert Errors.DepositorVault_MaximumSupplyExceeded(_maxSupply);
   }
```

Though this cap is not enforced when donating, allowing the distributed yield to be added to the total assets after being issued.

```
return (totalAssets + yield, totalYield - yield);
```

Recommendation

Enforce the supply caps also while donating, if the issued yield + the total assets is more than the usdt cap, do not issue the remainder yield until the *`getTotalAssets() + yield <= _maxSupply`*.

Resolution

Acknowledged. This is accepted as is but implementing this would cause downstream problems in that it would stop users being able to repay their loans as yield is donated during the loan repayment.

[DV-M3] Snapshot data might be inaccurate if there hasn’t been a deposit or withdrawal for over a week.

Severity	Category	Status
MEDIUM	User Loss	Resolved

Description

There is an edge-case where no snapshots have been taken for over a week, which means there have been no supplies or withdrawals.

If there has been a donation beforehand and there have been no deposits/withdrawals, the yield will be distributed after the distribution window period (currently set to 1 week).

This will change the totalAssets() in the vault, but no snapshot will be taken, leading to inaccurate data.

Recommendation

Add an edge-case if statement that if there hasn’t been a snapshot in the distribution window period and there is yield from a previous donation, snapshot it.

Resolution

Fixed at commit: f10c207e43c898d9eb65c624aa0b8f5193fa1475

[DV-M4] Sanctioned users can still interact with the vault by leveraging the vault marketplace.

Severity	Category	Status
MEDIUM	Protocol malfunction	Resolved

Description

Ambit uses a sanctionable modifier which is a global sanction list for wallets that are linked to people like North Korean hackers, drug traffickers etc. This modifier is used on the depositor vault when calling *`deposit()`* and *`withdraw()`*.

This sanction list can be bypassed from the depositor vault marketplace as it takes msg.sender when depositing in the *`buy()`* function, allowing a sanctioned wallet to deposit assets in the vault.

Recommendation

Add the to:

```
+ function buy(uint256 amount, IMarketplaceAdapter.Parameters calldata par
ams) external returns (uint256) notSanctioned(msg.sender) {
+ function sell(uint256 amount, IMarketplaceAdapter.Parameters calldata pa
rams) external returns (uint256) notSanctioned(msg.sender) {
```

in the depositor vault marketplace.

Resolution

Fixed at commit: 340aa4bbb0be7a2a90d94f18720bfc94ef0d4ef7

[VW-M1] Truncation in the vesting wallet will eventually unlock 100% of the tokens one month later.

Severity	Category	Status
MEDIUM	User Loss	Acknowledged

Description

Given the truncation in the vesting calculation, investors/team members on the vesting wallet will receive their full vesting a month later than specified:

```
return (totalAllocation * (ellapsed / _interval * _interval)) / (duration(
));
```

Given the following scenario:

- start timestamp: 1708105605 (block.timestamp)
- durationSeconds: 157680000 (5 years)
- interval: 2630000 (1 month)

It takes 5 years and 1 month to fully release the whole vested amount.

Recommendation

Do not apply truncation so that the full amount is able to be claimed by the 5-year mark and not a month late or specify 4 years and 11 months as the *`durationSeconds`* given that everything will be vested on the 5-year mark.

Resolution

Acknowledged, though vesting durations will be configured accordingly.

[L-M1] Points are not claimed when accruing rewards which will claim less points than available when those are accrued.

Severity	Category	Status
MEDIUM	User Loss	Acknowledged

Description

When a user tries to claim rewards for their loyalty points, those points are not accrued beforehand. This will cause the claim of rewards to be stale if the user has points already accrued but has not claimed them separately.

Eventually, this will claim less rewards than the ones that should be claimed.

Recommendation

```
function claimRewards(address account, address token) external onlyRewardT
okens(token) whenNotPaused {

-    ILoyaltyStorage.UserReward memory userReward = accrueRewards(account,
token);
+    claimPoints(account);
+    ILoyaltyStorage.UserReward memory userReward = loyaltyStorage.getUser
Reward(account,token);

    uint256 amount = userReward.accrued;
    userReward.accrued = 0;

    ILoyaltyStorage loyaltyStorage = _registry.getLoyaltyStorage();
    loyaltyStorage.setUserReward(account, token, userReward);

    IERC20Metadata(token).safeTransfer(account, amount);

    emit ClaimRewards(account, token, msg.sender, amount);
  }
```

Resolution

Acknowledged. Leaving as is for now, will handle in the frontend.

[GLOBAL-L1] Un-used errors across codebase.

Severity	Category	Status
LOW	Un-used errors	Resolved

Description

The following errors are un-used across the codebase:

```
error Validation_LiquidationDiscountOutOfRange(BPS maxLTV);
 error Hooks_NotImplemented();
  error Portfolio_SlippageExceeded(uint256 actual, uint256 expected);
  error DepositorVault_FlashLoanAmountExceeded(uint256 available, uint256
amount);
    error DepositorVault_FlashLoanTokenNotSupported(address token);
    error DepositorVault_FlashLoanReceiverNotAllowed(address receiver);
    error DepositorVault_FlashLoanReceiverFailed(address receiver);
    error DepositorVault_WithdrawUnavailableInCurrentBlock(uint256 current
Block, uint256 lastUpdateBlock);

  // dynamic interest rate model
  error DynamicInterestRateModel_PrecisionTooLarge(uint256 decimals);

  // token vesting
  error TokenVesting_TimestampNotReached(uint256 timestamp);

  error TokenVesting_NotAvailable(uint256 timestamp);
```

Recommendation

Remove them.

Resolution

Fixed at commit: 248f30316a8bcd0d6ba0bfb1057b285cd0b64900

[GLOBAL-L2] Misleading naming convention for native transfers

Severity	Category	Status
LOW	Naming	Acknowledged

Description

Ambit will start deploying in BSC, therefore ETH is not the main currency of that chain. Also, when deploying multi-chain, the same might be the case for other chains.

The name that Ambit has in Portfolio, Sweepable, and Treasury funtions is misleading. The functions interacting with the native currency are: supplyETH, sweepETH…

```
function supplyETH() external payable
function sweepETH(uint256 amount) public
function transferETH(address recipient, uint256 amount) external
```

These functions should be renamed to *supplyNative*, *sweepNative*, *transferNative*

Recommendation

Update the naming convention to: *supplyNative*, *sweepNative*, *transferNative*

Resolution

Acknowledged. This has become a convention that is used on Solidity protocols so will keep as is for now.

[DV-L1] Missing a max cap for the donation fee.

Severity	Category	Status
LOW	Missing checks	Resolved

Description

The depositor vault has a donation fee which as of now can be set to as much as it is wanted eventually being able to steal from anyone donating if it is too high.

This fee is then sent to the treasury:

```
(uint256 feeAmount, address feeReceiver) = previewDonationFee(amount);

    if (feeAmount > 0) {
      underlyingAsset.safeTransferFrom(msg.sender, feeReceiver, feeAmount)
;
    }
```

Currently there is no max fee that can be set:

```
function setDonationFee(Fees.Parameters memory fee) external onlyAdmin {
    _donationFee = fee;

    emit SetDonationFee(msg.sender, fee);
  }
```

Recommendation

Add a max donation fee validation to not allow the value to be set to high:

```
function setDonationFee(Fees.Parameters memory fee) external onlyAdmin {
    _donationFee = fee;

+    if (BPS.unwrap(_donationFee.bps) > MAX_DONATION_FEE) {
+      revert Errors.Validation_LimitExceeded(
+        MAX_DONATION_FEE,
+        BPS.unwrap(_donationFee.bps)
+      );
+      }

    emit SetDonationFee(msg.sender, fee);
  }
```

Resolution

Fixed at commit: b3d05c76c890d6bd51969e65e63ebdfd100b03f6

[BC-L1] Incorrect NatSpec

Severity	Category	Status
LOW	NatSpec	Resolved

Description

On the *`function beforeSupply(address, uint256, uint256) internal virtual { }`* function in the Base Custodian contract, there is the following NatSpec:

`/// @dev called before the assets are transferred from the supplier.`

Meaning that this will be triggered before the user transfers the funds. When you go to the implementation of the *`supply()`* function, you can see how this is not correct, and in fact, it is called after transferring funds:

```
 _underlyingAsset.safeTransferFrom(supplier, address(this), amount);

    shares = previewSupply(amount);

    beforeSupply(supplier, amount, shares);
```

Recommendation

Change the comment to the following:

```
- /// @dev called before the assets are transferred from the supplier.
+ /// @dev called after the assets are transferred from the supplier.
```

Resolution

Fixed at commit: d24478a325236488cd4ff46a40e6717ac02db168

[LENS-L1] getHealthScore() incorrectly returns when the liabilities are 0

Severity	Category	Status
LOW	Incorrect behavior	Acknowledged

Description

If a user has never interacted with the protocol, the Lens contract will return a health score of a 1000 while it should return a healthscore of 0, as it has never interacted.

```
    IMarket market = _registry.getMarket();

    uint256 liabilities = market.getLiabilities(account);

    if (liabilities == 0) {
      return MAX_HEALTH_SCORE;
    }
    return calculateHealthScore(market.getBorrowLimit(account, false), lia
bilities);
  }
```

Recommendation

Update it so that it returns 0 for new accounts:

```
 if (liabilities == 0) {
+if (borrowLimit == 0){
+    return 0;
+}
 return MAX_HEALTH_SCORE;
    }
```

Resolution

Acknowledged. A health score of 1000 (perfect) is preferrable as a health score of 0 could then have a dual meaning, i.e., the user hasn't borrowed, or they have borrowed but their portfolio is now worth zero.

[LL-L1] Incorrect boost is specified.

Severity	Category	Status
LOW	NatSpec	Resolved

Description

Currently a 10x burning boost is specified while a 5x burning boost is being used:

```
// the boost to apply to burnt token points
  uint32 public constant BURN_BOOST = 50; // 10.0
```

Recommendation

Consider changing the comment to accurately represent the correct boost multiplier:

```
// the boost to apply to burnt token points
- uint32 public constant BURN_BOOST = 50; // 10.0
+ uint32 public constant BURN_BOOST = 50; // 5.0
```

Resolution

Fixed at commit: 3adf5938dd8a147b36984dad20efeeac3cc877dc

[CAPO-L1] Timeout can’t be adjusted.

Severity	Category	Status
LOW	Composability	Acknowledged

Description

Currently, on the price feeds, it a price is treated as stale if it surpasses the timeout:

```
return block.timestamp - timestamp > _timeout;
```

This *_timeout* is currently immutable and won’t be able to be updated if needed. As Chainlink is an external system and the accurate *_timeout* might be subject to change on the future, it should be able to be updated.

Recommendation

Add a permissioned setter function for setting the timeout if needed.

```
- uint256 private immutable _timeout;
+ uint256 private _timeout;
```

Resolution

Acknowledged. A new contract can be redeployed if a timeout change is needed.

[CAPO-L2] BNB/USD feed is used for WBNB/USD

Severity	Category	Status
LOW	Incorrect behavior	Acknowledged

Description

According to on-chain data, Ambit is using the price feed of BNB/USD from Chainlink: https://data.chain.link/bsc/mainnet/crypto-usd/bnb-usd to price WBNB.

While this is not a big problem, WBNB might have small deviation in prices over the time in comparison to BNB.

Recommendation

Request a WBNB/USD price feed to be added in Chainlink so the price of the correct asset can be fetched.

Resolution

Acknowledged, from a liquidation perspective, WBNB can be exchanged for BNB.

[FPO-L1] Most stale price can be used instead of most recent one.

Severity	Category	Status
LOW	Incorrect behavior	Acknowledged

Description

Ambit has a double oracle architecture, featuring Chainlink as the main oracle and Binance Oracles as secondary oracles.

If both oracles are stale, it will always return the Chainlink price, instead of the most recently updated.

```
 if (isStale) {
      (USD fallbackPrice, bool fallbackIsStale) = _fallbackOracle.getLates
tPrice();

      // return the most recent price
      return fallbackIsStale == false ? (fallbackPrice, false) : (price, i
sStale);
    }
```

Recommendation

Add logic to return the least stale price in case both are stale.

Resolution

Acknowledged, will resolve in a future release.

DISCLAIMER

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Marc Weiss to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk.

My position is that each company and individual are responsible for their own due diligence and continuous security. My goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze. Therefore, I do not guarantee the explicit security of the audited smart contract, regardless of the verdict.