METATRUST

Draft
Security Assessment for

131-2022-05-backd (1OK-FLP) (1Negative-SP)

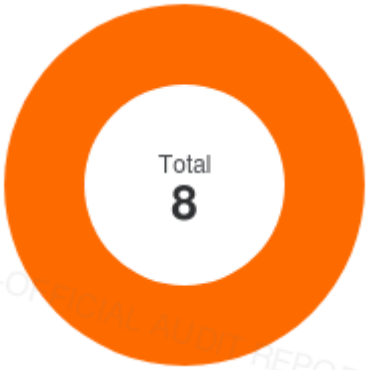
July 23, 2023

Executive Summary

Overview	
Project Name	131-2022-05-backd (1OK-FLP) (1Negative-SP)
Codebase URL	https://github.com/daoyuan14/2022-05-backd/
Scan Engine	AI Analyzer
Scan Time	2023/07/23 21:39:24
Commit Id	bffa09a

Total	
Critical Issues	0
High risk Issues	8
Medium risk Issues	0
Low risk Issues	0
Informational Issues	0

Critical Issues	The issue can cause large economic losses, large-scale data disorder, loss of control of authority management, failure of key functions, or indirectly affect the correct operation of other smart contracts interacting with it.
High Risk Issues	The issue puts a large number of users' sensitive information at risk or is reasonably likely to lead to catastrophic impacts on clients' reputations or serious financial implications for clients and users.
Medium Risk Issues	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk Issues	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational Issue	The issue does not pose an immediate risk but is relevant to security best practices or Defence in Depth.



Critical Issues	0%	0
High risk Issues	100%	8
Medium risk Issues	0%	0
Low risk Issues	0%	0
Informational Issues	0%	0

Summary of Findings

MetaScan security assessment was performed on **July 23, 2023 21:39:24** on project **131-2022-05-backd (1OK-FLP) (1Negative-SP)** with the repository **https://github.com/daoyuan14/2022-05-backd/** on branch **default branch**. The assessment was carried out by scanning the project's codebase using the scan engine **AI Analyzer**. There are in total **8** vulnerabilities / security risks discovered during the scanning session, among which **0** critical vulnerabilities, **8** high risk vulnerabilities, **0** medium risk vulnerabilities, **0** low risk vulnerabilities, **0** informational issues.

ID	Description	Severity
MSA-001	MWE-200: Insecure LP Token Value Calculation	High risk
MSA-002	MWE-200: Insecure LP Token Value Calculation	High risk
MSA-003	MWE-200: Insecure LP Token Value Calculation	High risk
MSA-004	MWE-200: Insecure LP Token Value Calculation	High risk
MSA-005	MWE-200: Insecure LP Token Value Calculation	High risk
MSA-006	MWE-200: Insecure LP Token Value Calculation	High risk
MSA-007	MWE-200: Insecure LP Token Value Calculation	High risk
MSA-008	MWE-200: Insecure LP Token Value Calculation	High risk

Findings

Critical (0)

No Critical vulnerabilities found here

High risk (8)

1. MWE-200: Insecure LP Token Value Calculation

High risk Security Analyzer

Liquidity token value/price can be manipulated to cause flashloan attacks.

File(s) Affected

protocol/contracts/swappers/SwapperRouter.sol #162-191

```
162     function _swapForWeth(address token_) internal returns (uint256 amountOut) {
163         if (token_ == address(_WETH)) return _WETH.balanceOf(address(this));
164 
165         // Handling ETH -> WETH
166         if (token_ == address(0)) {
167             uint256 ethBalance_ = address(this).balance;
168             if (ethBalance_ == 0) return 0;
169             _WETH.deposit{value: ethBalance_}();
170             return ethBalance_;
171         }
172 
173         // Handling Curve Pool swaps
174         ICurveSwapEth curvePool_ = curvePools[token_];
175         if (address(curvePool_) != address(0)) {
176             uint256 amount_ = IERC20(token_).balanceOf(address(this));
177             if (amount_ == 0) return 0;
178             _approve(token_, address(curvePool_));
179             (uint256 wethIndex_, uint256 tokenIndex_) = _getIndices(curvePool_, token_);
180             curvePool_.exchange(
181                 tokenIndex_,
182                 wethIndex_,
183                 amount_,
184                 _minWethAmountOut(amount_, token_)
185             );
186             return _WETH.balanceOf(address(this));
187         }
188 
189         // Handling ERC20 -> WETH
190         return _swap(token_, address(_WETH), IERC20(token_).balanceOf(address(this)));
191     }
```

protocol/contracts/swappers/SwapperRouter.sol #125-155

```
125     function swap(
126         address fromToken_,
127         address toToken_,
128         uint256 amountIn_
129     ) public payable override returns (uint256 amountOut) {
130         // Validating ETH value sent
131         require(msg.value == (fromToken_ == address(0) ? amountIn_ : 0), Error.INVALID_AMOUNT);
132         if (amountIn_ == 0) {
133             emit Swapped(fromToken_, toToken_, 0, 0);
134             return 0;
135         }
136 
137         // Handling swap between the same token
138         if (fromToken_ == toToken_) {
139             if (fromToken_ == address(0)) {
140                 payable(msg.sender).transfer(amountIn_);
141             }
142             emit Swapped(fromToken_, toToken_, amountIn_, amountIn_);
143             return amountIn_;
144         }
145 
146         // Transferring to contract if ERC20
147         if (fromToken_ != address(0)) {
148             IERC20(fromToken_).safeTransferFrom(msg.sender, address(this), amountIn_);
149         }
150 
151         // Swapping token via WETH
152         uint256 amountOut_ = _swapWethForToken(toToken_, _swapForWeth(fromToken_));
153         emit Swapped(fromToken_, toToken_, amountIn_, amountOut_);
154         return _returnTokens(toToken_, amountOut_);
155     }
```

Recommendation

Do not use AMM pool or custom liquidity calculation to caculate LP token value/price.

2. MWE-200: Insecure LP Token Value Calculation

High risk Security Analyzer

Liquidity token value/price can be manipulated to cause flashloan attacks.

File(s) Affected

protocol/contracts/swappers/SwapperRouter.sol #414-425

```
414     function _minTokenAmountOut(uint256 wethAmount_, address token_)
415         internal
416         view
417         returns (uint256 minAmountOut)
418     {
419         uint256 priceInEth_ = _getPriceInEth(token_);
420         if (priceInEth_ == 0) return 0;
421         return
422             wethAmount_.scaledDiv(priceInEth_).scaledMul(slippageTolerance).scaleTo(
423                 IERC20Full(token_).decimals()
424             );
425     }
```

protocol/contracts/swappers/SwapperRouter.sol #198-227

```
198     function _swapWethForToken(address token_, uint256 amount_)
199         internal
200         returns (uint256 amountOut)
201     {
202         if (amount_ == 0) return 0;
203         if (token_ == address(_WETH)) return amount_;
204 
205         // Handling WETH -> ETH
206         if (token_ == address(0)) {
207             _WETH.withdraw(amount_);
208             return amount_;
209         }
210 
211         // Handling Curve Pool swaps
212         ICurveSwapEth curvePool_ = curvePools[token_];
213         if (address(curvePool_) != address(0)) {
214             _approve(address(_WETH), address(curvePool_));
215             (uint256 wethIndex_, uint256 tokenIndex_) = _getIndices(curvePool_, token_);
216             curvePool_.exchange(
217                 wethIndex_,
218                 tokenIndex_,
219                 amount_,
220                 _minTokenAmountOut(amount_, token_)
221             );
222             return IERC20(token_).balanceOf(address(this));
223         }
224 
225         // Handling WETH -> ERC20
226         return _swap(address(_WETH), token_, amount_);
227     }
```

Recommendation

Do not use AMM pool or custom liquidity calculation to caculate LP token value/price.

3. MWE-200: Insecure LP Token Value Calculation

High risk Security Analyzer

Liquidity token value/price can be manipulated to cause flashloan attacks.

File(s) Affected

protocol/contracts/swappers/SwapperRouter.sol #433-444

```
433     function _minWethAmountOut(uint256 tokenAmount_, address token_)
434         internal
435         view
436         returns (uint256 minAmountOut)
437     {
438         uint256 priceInEth_ = _getPriceInEth(token_);
439         if (priceInEth_ == 0) return 0;
440         return
441             tokenAmount_.scaledMul(priceInEth_).scaledMul(slippageTolerance).scaleFrom(
442                 IERC20Full(token_).decimals()
443             );
444     }
```

protocol/contracts/swappers/SwapperRouter.sol #162-191

```
162     function _swapForWeth(address token_) internal returns (uint256 amountOut) {
163         if (token_ == address(_WETH)) return _WETH.balanceOf(address(this));
164 
165         // Handling ETH -> WETH
166         if (token_ == address(0)) {
167             uint256 ethBalance_ = address(this).balance;
168             if (ethBalance_ == 0) return 0;
169             _WETH.deposit{value: ethBalance_}();
170             return ethBalance_;
171         }
172 
173         // Handling Curve Pool swaps
174         ICurveSwapEth curvePool_ = curvePools[token_];
175         if (address(curvePool_) != address(0)) {
176             uint256 amount_ = IERC20(token_).balanceOf(address(this));
177             if (amount_ == 0) return 0;
178             _approve(token_, address(curvePool_));
179             (uint256 wethIndex_, uint256 tokenIndex_) = _getIndices(curvePool_, token_);
180             curvePool_.exchange(
181                 tokenIndex_,
182                 wethIndex_,
183                 amount_,
184                 _minWethAmountOut(amount_, token_)
185             );
186             return _WETH.balanceOf(address(this));
187         }
188 
189         // Handling ERC20 -> WETH
190         return _swap(token_, address(_WETH), IERC20(token_).balanceOf(address(this)));
191     }
```

Recommendation

Do not use AMM pool or custom liquidity calculation to caculate LP token value/price.

4. MWE-200: Insecure LP Token Value Calculation

High risk Security Analyzer

Liquidity token value/price can be manipulated to cause flashloan attacks.

File(s) Affected

protocol/contracts/pool/LiquidityPool.sol #504-525

```
504     function depositFor(
505         address account,
506         uint256 depositAmount,
507         uint256 minTokenAmount
508     ) public payable override notPaused returns (uint256) {
509         if (depositAmount == 0) return 0;
510         uint256 rate = exchangeRate();
511 
512         _doTransferIn(msg.sender, depositAmount);
513         uint256 mintedLp = depositAmount.scaledDiv(rate);
514         require(mintedLp >= minTokenAmount && mintedLp > 0, Error.INVALID_AMOUNT);
515 
516         lpToken.mint(account, mintedLp);
517         _rebalanceVault();
518 
519         if (msg.sender == account || address(this) == account) {
520             emit Deposit(msg.sender, depositAmount, mintedLp);
521         } else {
522             emit DepositFor(msg.sender, account, depositAmount, mintedLp);
523         }
524         return mintedLp;
525     }
```

Recommendation

Do not use AMM pool or custom liquidity calculation to caculate LP token value/price.

5. MWE-200: Insecure LP Token Value Calculation

High risk Security Analyzer

Liquidity token value/price can be manipulated to cause flashloan attacks.

File(s) Affected

protocol/contracts/pool/LiquidityPool.sol #608-616

```
608     function exchangeRate() public view override returns (uint256) {
609         uint256 totalUnderlying_ = totalUnderlying();
610         uint256 totalSupply = lpToken.totalSupply();
611         if (totalSupply == 0 || totalUnderlying_ == 0) {
612             return ScaledMath.ONE;
613         }
614 
615         return totalUnderlying_.scaledDiv(totalSupply);
616     }
```

protocol/contracts/pool/LiquidityPool.sol #533-559

```
533     function redeem(uint256 redeemLpTokens, uint256 minRedeemAmount)
534         public
535         override
536         returns (uint256)
537     {
538         require(redeemLpTokens > 0, Error.INVALID_AMOUNT);
539         ILpToken lpToken_ = lpToken;
540         require(lpToken_.balanceOf(msg.sender) >= redeemLpTokens, Error.INSUFFICIENT_BALANCE);
541 
542         uint256 withdrawalFee = addressProvider.isAction(msg.sender)
543             ? 0
544             : getWithdrawalFee(msg.sender, redeemLpTokens);
545         uint256 redeemMinusFees = redeemLpTokens - withdrawalFee;
546         // Pay no fees on the last withdrawal (avoid locking funds in the pool)
547         if (redeemLpTokens == lpToken_.totalSupply()) {
548             redeemMinusFees = redeemLpTokens;
549         }
550         uint256 redeemUnderlying = redeemMinusFees.scaledMul(exchangeRate());
551         require(redeemUnderlying >= minRedeemAmount, Error.NOT_ENOUGH_FUNDS_WITHDRAWN);
552 
553         _rebalanceVault(redeemUnderlying);
554 
555         lpToken_.burn(msg.sender, redeemLpTokens);
556         _doTransferOut(payable(msg.sender), redeemUnderlying);
557         emit Redeem(msg.sender, redeemUnderlying, redeemLpTokens);
558         return redeemUnderlying;
559     }
```

Recommendation

Do not use AMM pool or custom liquidity calculation to caculate LP token value/price.

6. MWE-200: Insecure LP Token Value Calculation

High risk Security Analyzer

Liquidity token value/price can be manipulated to cause flashloan attacks.

File(s) Affected

protocol/contracts/pool/LiquidityPool.sol #711-737

```
711     function _rebalanceVault(uint256 underlyingToWithdraw) internal {
712         IVault vault = getVault();
713 
714         if (address(vault) == address(0)) return;
715         uint256 lockedLp = staker.getStakedByActions();
716         uint256 totalUnderlyingStaked = lockedLp.scaledMul(exchangeRate());
717 
718         uint256 underlyingBalance = _getBalanceUnderlying(true);
719         uint256 maximumDeviation = totalUnderlyingStaked.scaledMul(getMaxReserveDeviationRatio());
720 
721         uint256 nextTargetBalance = totalUnderlyingStaked.scaledMul(getRequiredReserveRatio());
722 
723         if (
724             underlyingToWithdraw > underlyingBalance ||
725             (underlyingBalance - underlyingToWithdraw) + maximumDeviation < nextTargetBalance
726         ) {
727             uint256 requiredDeposits = nextTargetBalance + underlyingToWithdraw - underlyingBalance;
728             vault.withdraw(requiredDeposits);
729         } else {
730             uint256 nextBalance = underlyingBalance - underlyingToWithdraw;
731             if (nextBalance > nextTargetBalance + maximumDeviation) {
732                 uint256 excessDeposits = nextBalance - nextTargetBalance;
733                 _doTransferOut(payable(address(vault)), excessDeposits);
734                 vault.deposit();
735             }
736         }
737     }
```

protocol/contracts/pool/LiquidityPool.sol #533-559

```
533     function redeem(uint256 redeemLpTokens, uint256 minRedeemAmount)
534         public
535         override
536         returns (uint256)
537     {
538         require(redeemLpTokens > 0, Error.INVALID_AMOUNT);
539         ILpToken lpToken_ = lpToken;
540         require(lpToken_.balanceOf(msg.sender) >= redeemLpTokens, Error.INSUFFICIENT_BALANCE);
541 
542         uint256 withdrawalFee = addressProvider.isAction(msg.sender)
543             ? 0
544             : getWithdrawalFee(msg.sender, redeemLpTokens);
545         uint256 redeemMinusFees = redeemLpTokens - withdrawalFee;
546         // Pay no fees on the last withdrawal (avoid locking funds in the pool)
547         if (redeemLpTokens == lpToken_.totalSupply()) {
548             redeemMinusFees = redeemLpTokens;
549         }
550         uint256 redeemUnderlying = redeemMinusFees.scaledMul(exchangeRate());
551         require(redeemUnderlying >= minRedeemAmount, Error.NOT_ENOUGH_FUNDS_WITHDRAWN);
552 
553         _rebalanceVault(redeemUnderlying);
554 
555         lpToken_.burn(msg.sender, redeemLpTokens);
556         _doTransferOut(payable(msg.sender), redeemUnderlying);
557         emit Redeem(msg.sender, redeemUnderlying, redeemLpTokens);
558         return redeemUnderlying;
559     }
```

Recommendation

Do not use AMM pool or custom liquidity calculation to caculate LP token value/price.

7. MWE-200: Insecure LP Token Value Calculation

High risk Security Analyzer

Liquidity token value/price can be manipulated to cause flashloan attacks.

File(s) Affected

protocol/contracts/oracles/ChainlinkOracleProvider.sol #50-80

```
50      function _getPrice(
51          address asset_,
52          address denomination_,
53          bool revert_
54      ) internal view returns (uint256) {
55          try _feedRegistry.latestRoundData(asset_, denomination_) returns (
56              uint80 roundID_,
57              int256 price_,
58              uint256 startedAt,
59              uint256 timeStamp_,
60              uint80 answeredInRound_
61          ) {
62              require(timeStamp_ != 0, Error.ROUND_NOT_COMPLETE);
63              require(block.timestamp <= timeStamp_ + stalePriceDelay, Error.STALE_PRICE);
64              require(price_ != 0, Error.NEGATIVE_PRICE);
65              require(answeredInRound_ >= roundID_, Error.STALE_PRICE);
66
67              return uint256(price_).scaleFrom(_feedRegistry.decimals(asset_, denomination_));
68          } catch Error(string memory reason) {
69              if (revert_) revert(reason);
70
71              if (denomination_ == Denominations.USD) {
72                  return
73                      (_getPrice(asset_, Denominations.ETH, true) *
74                          _getPrice(Denominations.ETH, Denominations.USD, true)) / 1e18;
75              }
76              return
77                  (_getPrice(asset_, Denominations.USD, true) * 1e18) /
78                  _getPrice(Denominations.ETH, Denominations.USD, true);
79          }
80      }
```

Recommendation

Do not use AMM pool or custom liquidity calculation to caculate LP token value/price.

8. MWE-200: Insecure LP Token Value Calculation

High risk Security Analyzer

Liquidity token value/price can be manipulated to cause flashloan attacks.

File(s) Affected

protocol/contracts/tokenomics/VestedEscrowRevocable.sol #84-91

```
84      function balanceOf(address _recipient) external view override returns (uint256) {
85          uint256 timestamp = block.timestamp;
86          uint256 timeRevoked = revokedTime[_recipient];
87          if (timeRevoked != 0) {
88              timestamp = timeRevoked;
89          }
90          return _balanceOf(_recipient, timestamp);
91      }
```

protocol/contracts/swappers/SwapperRouter.sol #162-191

```
162     function _swapForWeth(address token_) internal returns (uint256 amountOut) {
163         if (token_ == address(_WETH)) return _WETH.balanceOf(address(this));
164
165         // Handling ETH -> WETH
166         if (token_ == address(0)) {
167             uint256 ethBalance_ = address(this).balance;
168             if (ethBalance_ == 0) return 0;
169             _WETH.deposit{value: ethBalance_}();
170             return ethBalance_;
171         }
172
173         // Handling Curve Pool swaps
174         ICurveSwapEth curvePool_ = curvePools[token_];
175         if (address(curvePool_) != address(0)) {
176             uint256 amount_ = IERC20(token_).balanceOf(address(this));
177             if (amount_ == 0) return 0;
178             _approve(token_, address(curvePool_));
179             (uint256 wethIndex_, uint256 tokenIndex_) = _getIndices(curvePool_, token_);
180             curvePool_.exchange(
181                 tokenIndex_,
182                 wethIndex_,
183                 amount_,
184                 _minWethAmountOut(amount_, token_)
185             );
186             return _WETH.balanceOf(address(this));
187         }
188
189         // Handling ERC20 -> WETH
190         return _swap(token_, address(_WETH), IERC20(token_).balanceOf(address(this)));
191     }
```

Recommendation

Do not use AMM pool or custom liquidity calculation to caculate LP token value/price.

Medium risk (0)

No Medium risk vulnerabilities found here

Low risk (0)

No Low risk vulnerabilities found here

Informational (0)

No Informational vulnerabilities found here

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without MetaTrust's prior written consent in each instance.

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 MetaTrust 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. MetaTrust's position is that each company and individual are responsible for their own due diligence and continuous security. MetaTrust's 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.

The assessment services provided by MetaTrust is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS 131-2022-05-backd (1OK-FLP) (1Negative-SP) Security Assessment AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW,

MetaTrust HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, MetaTrust SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, MetaTrust MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, MetaTrust PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER MetaTrust NOR ANY OF MetaTrust'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. MetaTrust WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT MetaTrust'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING 131-2022-05-backd (1OK-FLP) (1Negative-SP) Security Assessment MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST MetaTrust WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF MetaTrust CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST MetaTrust WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.