CERTIK

Security Assessment

WEMIX Swap & DIOS(Dollar in and out Stabilizer)

CertiK Verified on Oct 15th, 2022

CertiK Verified on Oct 15th, 2022

WEMIX Swap & DIOS(Dollar in and out Stabilizer)

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 10/15/2022	N/A

CODEBASE

https://github.com/wemixarchive/weswap-core/tree/ad8c485d05a9701906dc4f58e949cd16080b0a23

https://github.com/wemixarchive/weswap-

...View All

COMMITS

ad8c485d05a9701906dc4f58e949cd16080b0a23

176869cbf42286369dccbb0875fbab51ee9211e7

f8ad5b7ba7549159377663f86aaa61c06e975227

...View All

Vulnerability Summary

24	13	0	1	10	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
5	Major	1 Resolved, 4 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
2	Medium	1 Resolved, 1 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
9	Minor	6 Resolved, 3 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
8	Informational	5 Resolved, 1 Partially Resolved, 2 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS

WEMIX SWAP & DIOS(DOLLAR IN AND OUT STABILIZER)

CODEBASE

WEMIX SWAP & DIOS(DOLLAR IN AND OUT STABILIZER)

Repository

https://github.com/wemixarchive/weswap-core/tree/ad8c485d05a9701906dc4f58e949cd16080b0a23
https://github.com/wemixarchive/weswap-periphery/tree/176869cbf42286369dccbb0875fbab51ee9211e7
https://github.com/wemixarchive/DIOS/tree/f8ad5b7ba7549159377663f86aaa61c06e975227

Commit

ad8c485d05a9701906dc4f58e949cd16080b0a23 176869cbf42286369dccbb0875fbab51ee9211e7
f8ad5b7ba7549159377663f86aaa61c06e975227

AUDIT SCOPE

WEMIX SWAP & DIOS(DOLLAR IN AND OUT STABILIZER)

46 files audited ● 9 files with Acknowledged findings ● 1 file with Partially Resolved findings
● 4 files with Resolved findings ● 32 files without findings

ID	File	SHA256 Checksum
WFB	contracts/WeswapFactory.sol	c8844ac1bd4f7fd04d5931cfee3582adc7336cf0e7e166b9ccc962597ae2219c
WPB	contracts/WeswapPair.sol	33773358a5a0453f9f5e3f78f7f256e491262b8a43898e9cb0bd60e2f7b71dc6
WZI	contracts/WeswapZapIn.sol	2f7ecebfd87d4fdda31a0e1b655093a95ad4b70b44d1017a5086e73866d8f73c
ZBB	contracts/ZapBase.sol	62607c3fb83e58b29e5db0b30376e19ef1ff8bb968b5fb46c2f3bd8ad7c0fb97
WVL	contracts/libraries/WeswapV2LiquidityMathLibrary.sol	404ad11987e8a368021cca36544b0232c729ed5af133ab6177b00c9bc9fe7669
WDD	contracts/tokens/WemixDollar.sol	39f4220b18e280cbf7b2aadaf1467bc833f7598671163b5851ff19bf13d29f7a
DIA	contracts/DollarInAndOutStaking.sol	0fcd3bcccbbe3db3a64c64694d381f160f95307f913bbb15cd95536404bffbd4
WUS	contracts/WUSDCTreasury.sol	5afc4612cd261c18965cf668e1dcca8e79c6bd4974306c2ee118824f38881931
WDE	contracts/WemixDollarExchange.sol	8dcd58d336d376eab1dfb9aed3c2a176c203e0c514e299609b22da52cdec8b81
WRB	contracts/WeswapRouter.sol	0755b09b3b6701e797f32b8c241496dd512b15c8013db84b30cae296097a2c9f
WZO	contracts/WeswapZapOut.sol	2a5720859601f433593084a5b32d27db3c509f54fa6e06d61e6f9672af484e1d
HWD	contracts/extensions/HellowWorld.sol	a6854da4ed15ac140c8c58730d97747a5c3ee49dd872dff87908683c52815a82
IWM	contracts/interfaces/IWWEMIX.sol	1270916cbb07c55efca6ef6c985e65c2727efc1b3a0c95f4abe8e4ee0481315c

ID	File	SHA256 Checksum
WWE	contracts/tokens/WWEMIX.sol	814478dd363522630a6ed20f54e85fbea9a93b065b01143594e5134c5a04c7fe
WER	contracts/WeswapERC20.sol	6ac73d542224af21f1a0f401a72a1a04c0c94cc3446bc609ae16c90d1b62173d
UTI	contracts/utils/Utils.sol	c46c12132159a5146e47fcb2a5258a7da27b02adbe910ecac569aa358ab264ed
MAT	contracts/libraries/Math.sol	5c2c6e1cd97bc282f63d86beb982a4194d7bc87885c09bb9f79c756518336aad
UQB	contracts/libraries/UQ112x112.sol	cf5e86db2163ad60674c6b75df19c21ba131264596c34b26cdf08cad980732c4
IER	contracts/interfaces/IERC20.sol	1f02452ca278e1ca22be98ae0e03ce912e87b656e6f95e3eba12ba12ba7954b5
IWC	contracts/interfaces/IWeswapCallee.sol	8cebdadeb2298bc06b36d90aa6d6ae4f5309f1bcb6de4109bca6f01139f9f702
IWE	contracts/interfaces/IWeswapERC20.sol	edeefbecfd43b9fc58741a1b799e6a571ca48843f2ea854af21ba56841501ec5
IWF	contracts/interfaces/IWeswapFactory.sol	7f7617481d945c2bfcf619683fde081123dea48b134f304906cd8fae47fed8d3
IWP	contracts/interfaces/IWeswapPair.sol	8d8c08464dc244d8440117551cd779e7e2a8ac6051a3272f6cf54d2f53cd2718
REA	contracts/interfaces/README.md	d1a64620dfcbbc7915c6e40143ca813ed615e539813481df227aef5c864a2a0e
IWW	contracts/interfaces/IWWEMIX.sol	e8537a128ba5e760744ab855b1d8f08a71cc1cf93c5930ecdd4d134ed5c3256c
IWR	contracts/interfaces/IWeswapERC20.sol	edeefbecfd43b9fc58741a1b799e6a571ca48843f2ea854af21ba56841501ec5
IWS	contracts/interfaces/IWeswapFactory.sol	780631263dd859da2937f2e87c267ef4cffd12115f3917aa4650b364703a5dc8
IWA	contracts/interfaces/IWeswapPair.sol	8d8c08464dc244d8440117551cd779e7e2a8ac6051a3272f6cf54d2f53cd2718
IWO	contracts/interfaces/IWeswapRouter.sol	5e2fafd76ede543521f6123be2079ab36e9d569350698d43b3c8d50cefa2df68

ID	File	SHA256 Checksum
IWZ	contracts/interfaces/IWeswapZapIn.sol	426e8c012b38c18ae3de1e562164fdc852d7d63a72aa0f5733f6ba420ea88423
IZO	contracts/interfaces/IWeswapZapOut.sol	2ffe5f37c11b27547bde68832cc23f741564d15a906ddd2cf9e3ea55ade9c661
RED	contracts/interfaces/README.md	215791baab037a4c6449723760d63762eb3451d2ca84141de3decb0327f25810
MAH	contracts/libraries/Math.sol	608bf284cf01f9e926b4f5819b9c4352ee3d7857b164828b309d59519da2bb43
THB	contracts/libraries/TransferHelper.sol	f6258ce3e09a4fd2caa0ced302fcc301443cf4ee48344adde82be446ac7fbfcd
WLB	contracts/libraries/WeswapLibrary.sol	976ffede2ecfa093a5e721c6801957d7dbfda7cedca12c852d1762b6fef73b7c
IDI	contracts/interfaces/IDollarInAndOutStaking.sol	24de2de5f0dea010a39618471e6ab1c86d27e03de6a36819ccca9fdae9b85fe7
IWU	contracts/interfaces/IWUSDC.sol	b4efc50c1b4a7b0193b707d3fe06dfe82bf5403583e331cbc705e49e4415817a
IWD	contracts/interfaces/IWUSDCTreasury.sol	3d622746810f785062fb52b51829cd8185b84cfd98012549b32adfa8a7a2c195
IWI	contracts/interfaces/IWemixDollar.sol	775bc3ea2f9d90c9de83555f0f319776bf476efc393395793661be2faaf36671
IDE	contracts/interfaces/IWemixDollarExchange.sol	b2502bd45ee2a7b83b110405b3dfa193b41c78226d99a49d3d3676b58bb17124
IEC	contracts/interfaces/IWeswapERC20.sol	2c0f216fb985198dff0450b5398a48b40092fefa7fcdf93e8306cdb17c7bf5f8
IFD	contracts/interfaces/IWeswapFactory.sol	b6025fa94f5ebe5ddee1e53c82832b405ece3f060ef3ebe181d59adf6efe05a2
IPD	contracts/interfaces/IWeswapPair.sol	8d8c08464dc244d8440117551cd779e7e2a8ac6051a3272f6cf54d2f53cd2718
IRD	contracts/interfaces/IWeswapRouter.sol	7aaa51cd9ce448744d2a17042bfeeba7c4ce56af51fc4cd080296b29b33955c9
BDI	contracts/libraries/Babylonian.sol	73adc4124fb9abcef2d5aa0a88cacdf7b4997c3d59047bafbf24e71e232a6836

ID	File	SHA256 Checksum
WLD	contracts/libraries/WeswapLibrary.sol	976ffede2ecfa093a5e721c6801957d7dbfda7cedca12c852d1762b6fef73b7c

APPROACH & METHODS

WEMIX SWAP & DIOS(DOLLAR IN AND OUT STABILIZER)

This report has been prepared for Wemix to discover issues and vulnerabilities in the source code of the WEMIX Swap & DIOS(Dollar in and out Stabilizer) project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS

WEMIX SWAP & DIOS(DOLLAR IN AND OUT STABILIZER)

24	0	5	2	9	8
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for WEMIX Swap & DIOS(Dollar in and out Stabilizer). Through this audit, we have uncovered 24 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
COR-01	Missing Zero Address Validation	Volatile Code	Minor	Resolved
COR-02	Unchecked ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Minor	Resolved
DIA-01	Financial Models	Logical Issue	Medium	Acknowledged
HWD-01	**Centralization Risks In HellowWorld.Sol**	**Centralization / Privilege**	**Major**	**Resolved**
WDD-01	**Centralization Related Risks In The `WhitelistAddress` Role**	**Centralization / Privilege**	**Major**	**Acknowledged**
WEM-01	**Centralization Related Risks In `owner` Role**	**Centralization / Privilege**	**Major**	**Acknowledged**
WEM-02	**Centralization Related Risks In The `breaker` And `breakerSetter` Role**	**Centralization / Privilege**	**Major**	**Acknowledged**
WFB-01	**Centralization Risks In `feeToSetter` Role**	**Centralization / Privilege**	**Major**	**Acknowledged**
WUS-01	Logical Issue Of `onlyWallet`	Logical Issue	Medium	Resolved
WUS-02	The Number Of `_quorum` And `_owners.length`	Logical Issue	Minor	Resolved

ID	Title	Category	Severity	Status
WUS-03	Unknown Implementation When Calling `executeTransaction()`	Logical Issue	Minor	Acknowledged
WVL-01	Divide Before Multiply	Mathematical Operations	Minor	Resolved
WZI-01	The Existence Of `_pairAddress` Should Be Checked	Logical Issue	Minor	Resolved
WZI-02	The `_FromTokenBContractAddress` Check	Logical Issue	Minor	Acknowledged
ZBB-01	Null Wallet Adddress	Logical Issue	Minor	Acknowledged
ZBB-02	Ineffective `isContract()` Check	Volatile Code	Minor	Resolved
COR-03	Missing Inheritance	Language Specific	Informational	Resolved
DIA-02	Code Comments In Two Languages	Coding Style	Informational	Resolved
DIA-03	Unused Event	Coding Style	Informational	Resolved
WEM-03	Missing Emit Events	Coding Style	Informational	Partially Resolved
WEM-04	Mathematical Calculations	Mathematical Operations	Informational	Acknowledged
WFB-02	Missing Error Messages	Coding Style	Informational	Resolved
WUS-04	Unlocked Compiler Version	Language Specific	Informational	Resolved
WZI-03	Explanation On The Use Of `_swapTarget	Logical Issue	Informational	Acknowledged

COR-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/DollarInAndOutStaking.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 86, 87, 88, 90, 91, 92; contracts/WemixDollarExchange.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 52, 53, 54	Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

```
86          WUSDC = _WUSDC;
```

- `_WUSDC` is not zero-checked before being used.

```
87          WEMIX$ = _WEMIX$;
```

- `_WEMIX$` is not zero-checked before being used.

```
88          weswapFactory = _weswapFactory;
```

- `_weswapFactory` is not zero-checked before being used.

```
90          WUSDCTreasury = _WUSDCTreasury;
```

```
91          feePool = _WUSDCTreasury;
```

- `_WUSDCTreasury` is not zero-checked before being used.

```
92          stabilityPool = _stabilityPool;
```

- `_stabilityPool` is not zero-checked before being used.

```
52          WUSDC = _WUSDC;
```

- `_WUSDC` is not zero-checked before being used.

```
53          WEMIX$ = _WEMIX$;
```

- `_WEMIX$` is not zero-checked before being used.

```
54          WUSDCTreasury = _WUSDCTreasury;
```

- `_WUSDCTreasury` is not zero-checked before being used.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

The team heeded our advice and resolved this issue in commit `d483b4b49d9e25bb4e9914743d24e6c6e9ad2540`.

COR-02 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	Minor	contracts/DollarInAndOutStaking.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 348, 356, 377, 407; contracts/WemixDollarExchange.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 69, 75	Resolved

Description

The return value of the transfer()/transferFrom() call is not checked.

```
348         IERC20(WEMIX$).transfer(feePool, _amount);
```

```
356         IERC20(WEMIX$).transfer(stabilityPool, _amount);
```

```
377         IERC20(_path[0]).transfer(weswapStablecoinsPair, _amounts[0]);
```

```
407         IERC20(WUSDC).transferFrom(WUSDCTreasury, address(this), _amounts[0]);
```

```
69        IWUSDC(WUSDC).transferFrom(WUSDCTreasury, msg.sender, _amount);
```

```
75        IWUSDC(WUSDC).transferFrom(msg.sender, WUSDCTreasury, _amount);
```

Recommendation

Since some ERC-20 tokens return no values and others return a `bool` value, they should be handled with care. We advise using the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `false` is returned, making it compatible with all ERC-20 token implementations.

Alleviation

The team heeded our advice and resolved this issue in commit `c6213dcc597ec02dfba31402c5f8d019753dc253`.

DIA-01 | FINANCIAL MODELS

Category	Severity	Location	Status
Logical Issue	Medium	contracts/DollarInAndOutStaking.sol (f8ad5b7ba7549159377663f86aaa61c06e975227)	Acknowledged

Description

In the project `Wemix`, the team designed a pair of tokens, the `WemixDollar` and `WUSDC`, they could always be swapped to each other by the ratio of 1:1. The codes of `WUSDC` is unknown, yet the `WemixDollar` could be unlimited minting, the reserves of the `WemixDollar-WUSDC` pool are always changing, hence there are huge arbitrage opportunities here that investors may change the higher price token to the lower and swap back by the 1:1 ratio to win profits. Hence these two tokens could be regarded as algorithm stable tokens.

Besides, the protocol wrapped the `addliquidity()` and `removeliquidity()` functions and provided the investors more functions to call, which they could add or remove liquidity with one or two of the paired tokens, or any other tokens, while more taxes will be charged as `goodwill` or `affiliation`. Also, the `WWEMIX` token is issued to replace the function of `WETH`.

Recommendation

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The financial model of this protocol is not in the scope of this audit.

In the real world, algorithm-stable tokens could be unstable by the sharp drop in price, we recommend the team constantly monitor the operation and the status of the whole project.

Alleviation

The team acknowledged this issue and they stated the following:

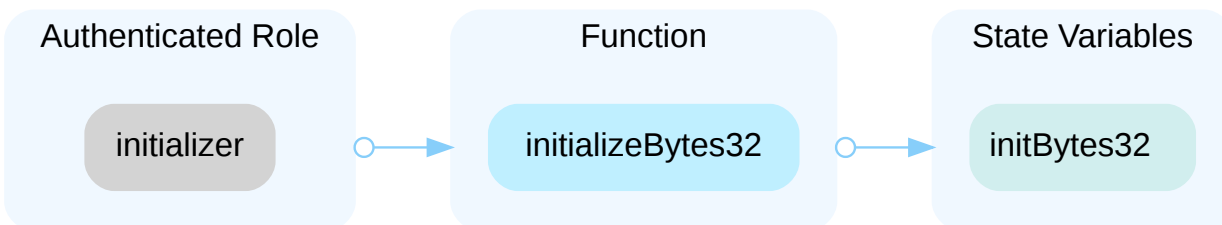
"DIOS is running on 100% backed stable coin (ex. USDC) and minting `Wemix` Dollar is always limited to the total amount of backed stable coin."

HWD-01 | CENTRALIZATION RISKS IN HELLOWWORLD.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/extensions/HellowWorld.sol (f8ad5b7ba754915937 7663f86aaa61c06e975227): 16	Resolved

Description

In the contract `HellowWorld` the role `initializer` has authority over the functions shown in the diagram below. Any compromise to the `initializer` account may allow the hacker to take advantage of this authority.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

The team removed the contract `HellowWorld` in the commit `d483b4b49d9e25bb4e9914743d24e6c6e9ad2540`.

WDD-01 CENTRALIZATION RELATED RISKS IN THE `WhitelistAddress` ROLE

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/tokens/WemixDollar.sol (f8ad5b7ba754915937 7663f86aaa61c06e975227): 51, 63**	**Acknowledged**

Description

In the contract `WemixDollar` , the role `WhitelistAddress` has authority over the following functions:

- function mint(), the privileged role could mint any amount of `WemixDollar` unlimitedly to anyone.
- function burn(), the privileged role could burn any amount of `WemixDollar` from anyone.

Any compromise to the `WhitelistAddress` account may allow a hacker to take advantage of this authority.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged this issue and they stated the following:

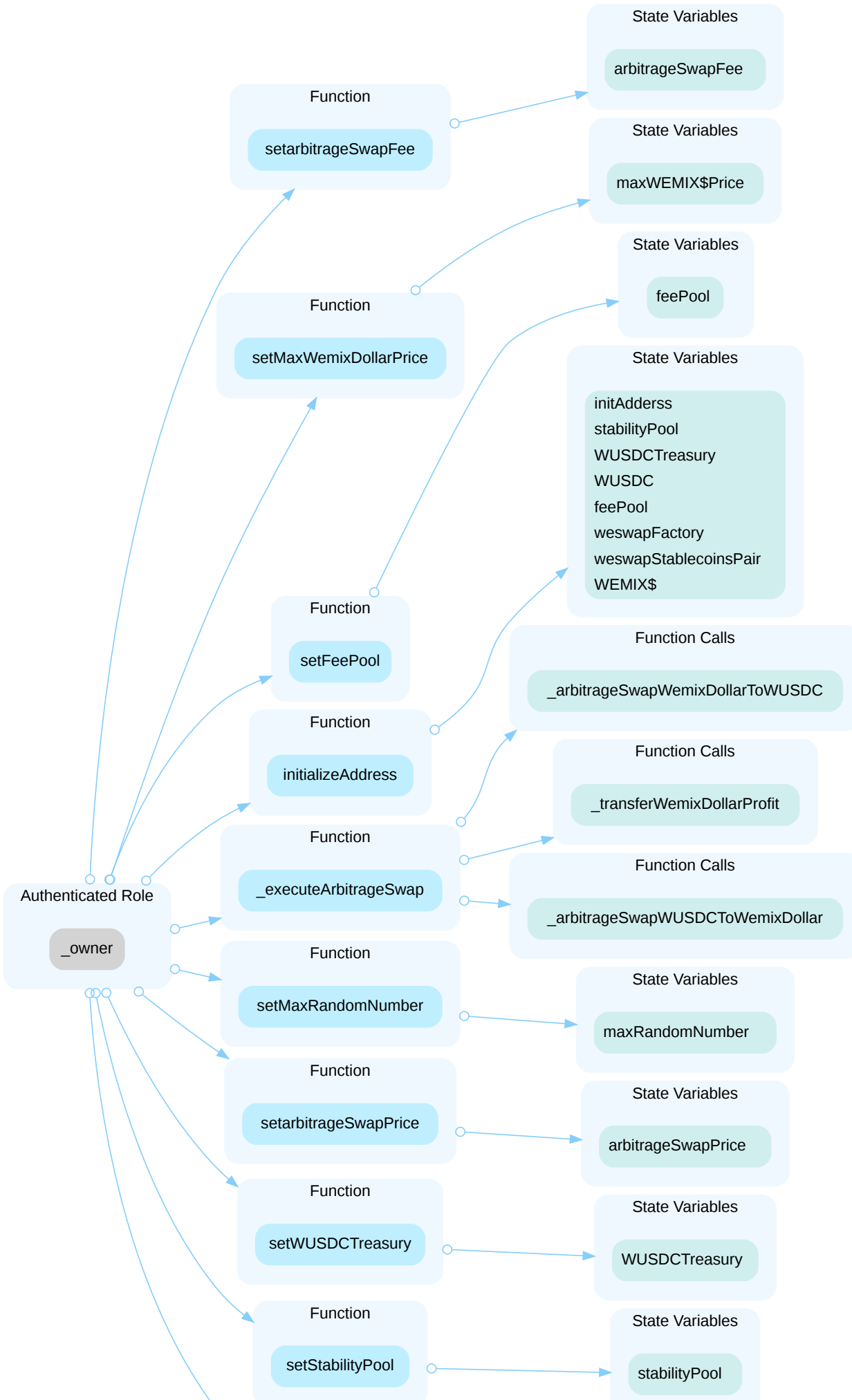
"They will adapt the multi-signature scheme to the contract owner's signature algorithm as a short-term solution and apply DAO as a long-term solution."

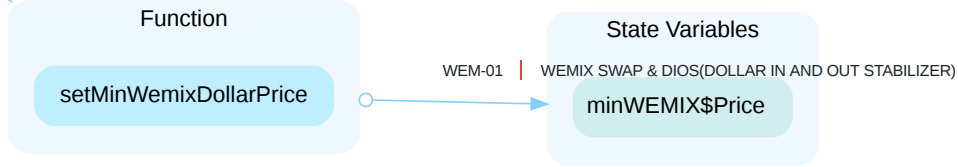
WEM-01 | CENTRALIZATION RELATED RISKS IN `owner` ROLE

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/DollarInAndOutStaking.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 85; contracts/WUSDCTreasury.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 59; contracts/WemixDollarExchange.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 51; contracts/tokens/WemixDollar.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 77; contracts/WeswapFactory.sol (ad8c485d05a9701906dc4f58e949cd16080b0a23): 36; contracts/ZapBase.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 75**	**Acknowledged**

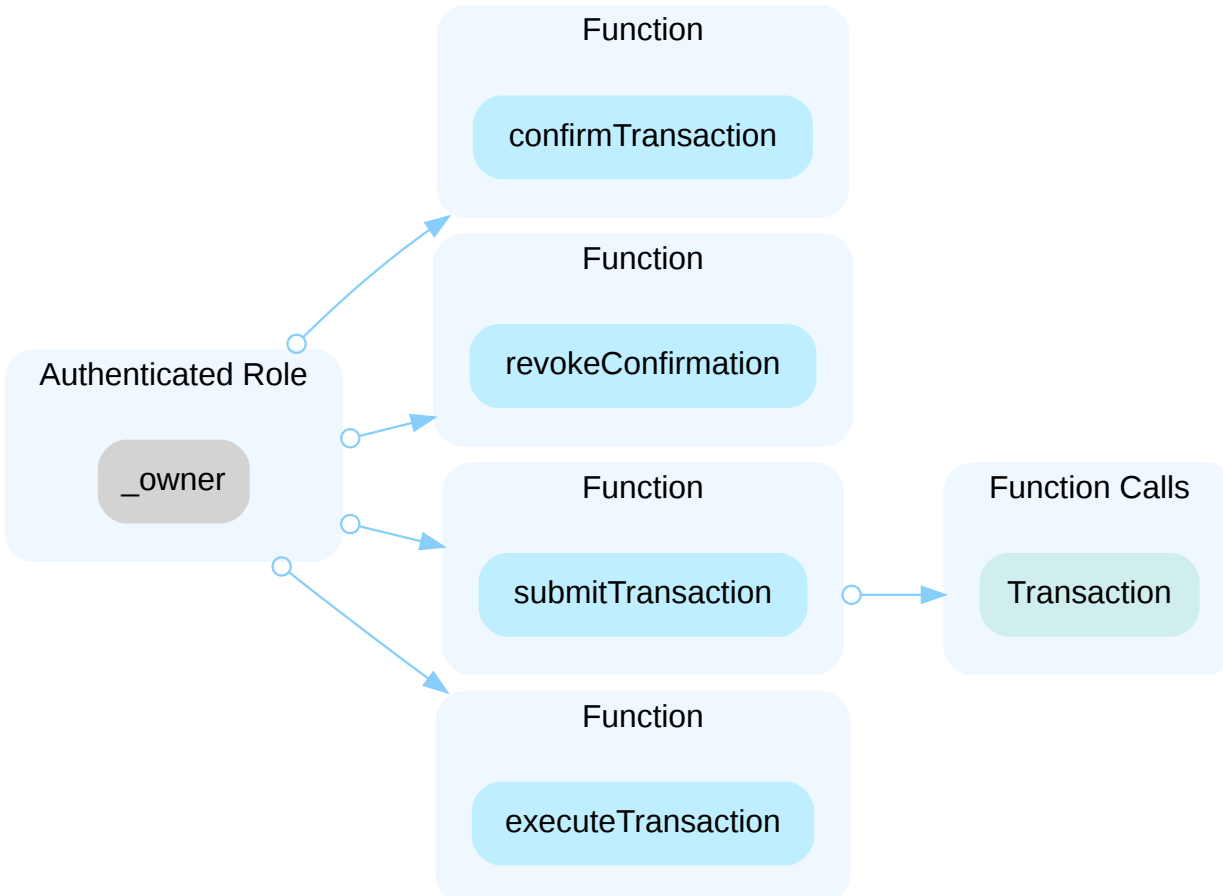
Description

In the contract `DollarInAndOutStaking` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.

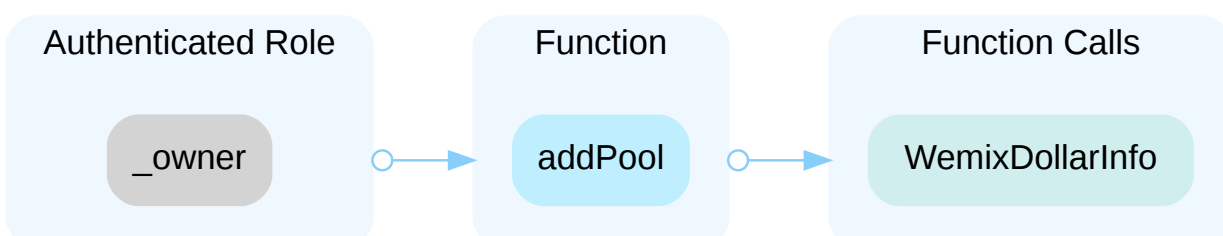

State Variables
arbitrageSwapFee
Function
setarbitrageSwapFee
State Variables
maxWEMIX$Price
Function
setMaxWemixDollarPrice
State Variables
feePool
State Variables
initAdderss
stabilityPool
WUSDCTreasury
WUSDC
feePool
weswapFactory
weswapStablecoinsPair
WEMIX$
Function
setFeePool
Function Calls
_arbitrageSwapWemixDollarToWUSDC
Function
initializeAddress
Function Calls
_transferWemixDollarProfit
Function
_executeArbitrageSwap
Function Calls
_arbitrageSwapWUSDCToWemixDollar
Authenticated Role
_owner
Function
setMaxRandomNumber
State Variables
maxRandomNumber
Function
setarbitrageSwapPrice
State Variables
arbitrageSwapPrice
Function
setWUSDCTreasury
State Variables
WUSDCTreasury
Function
setStabilityPool
State Variables
stabilityPool



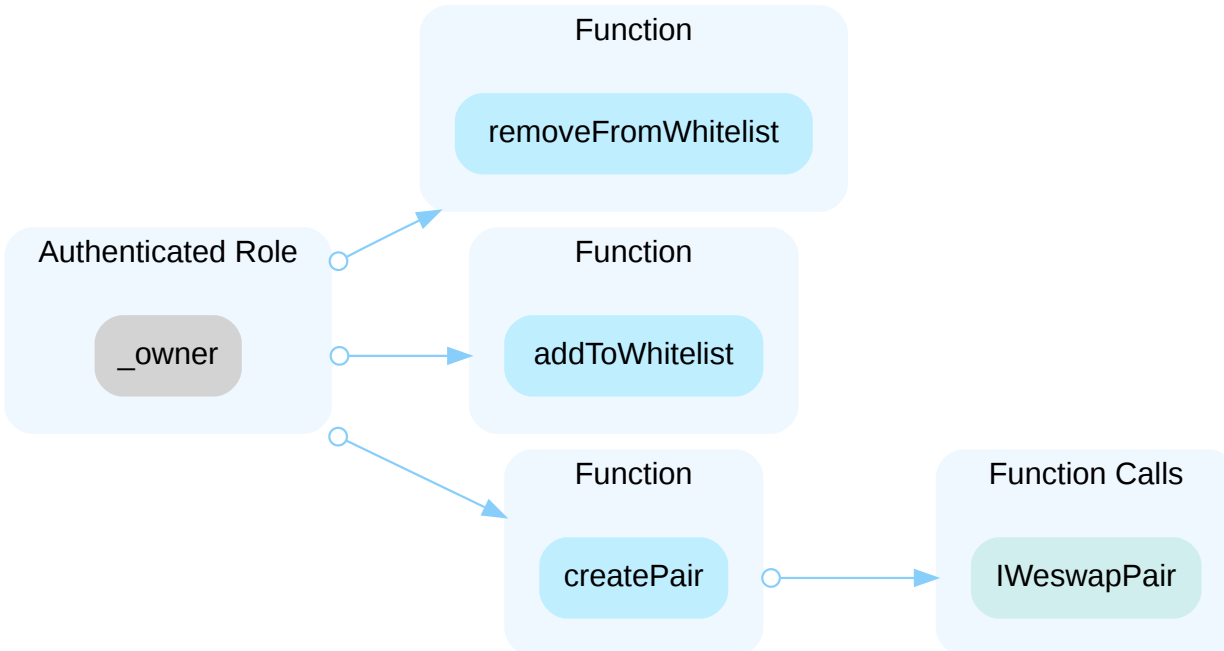
In the contract `WUSDCTreasury` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



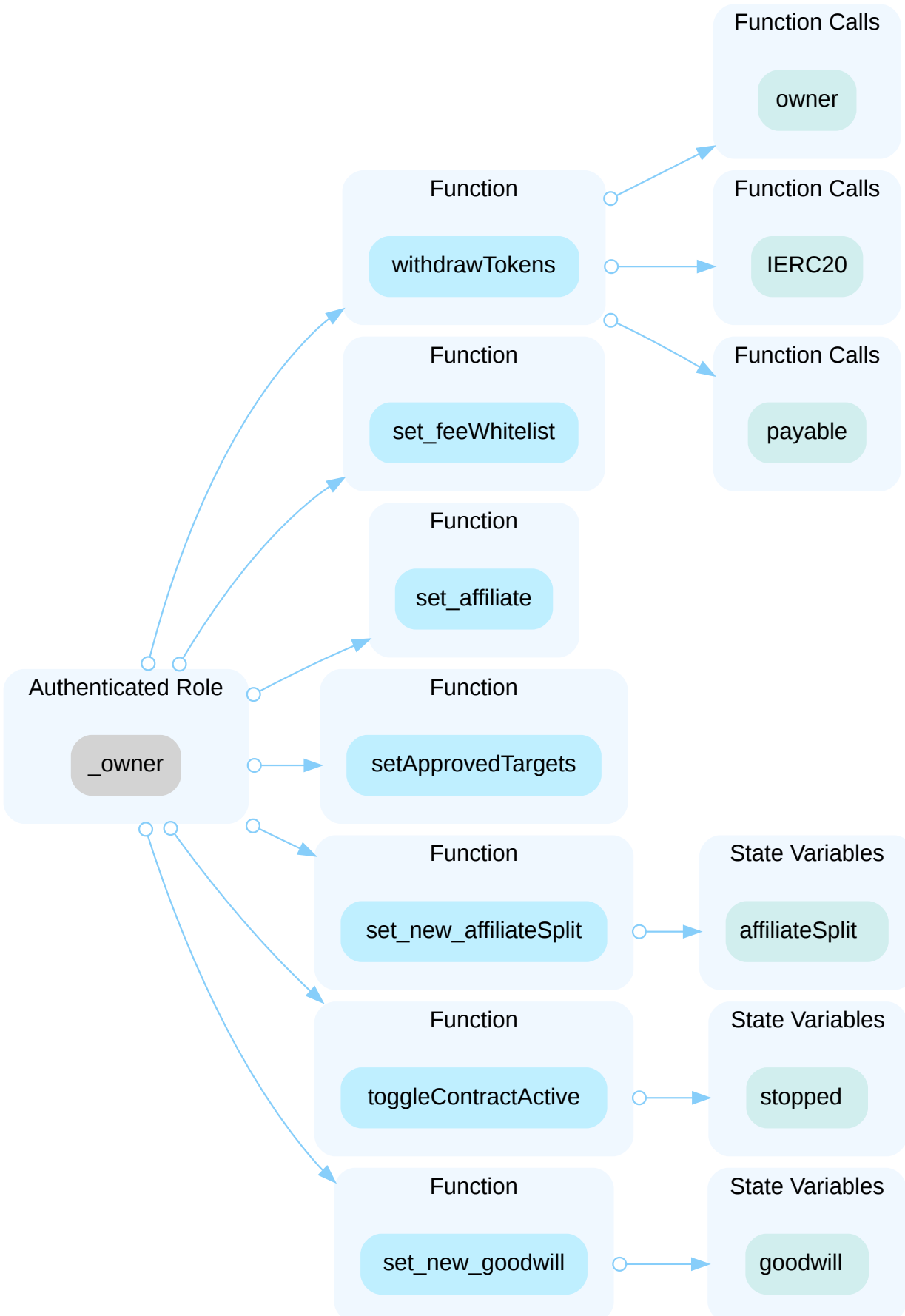
In the contract `WemixDollar` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



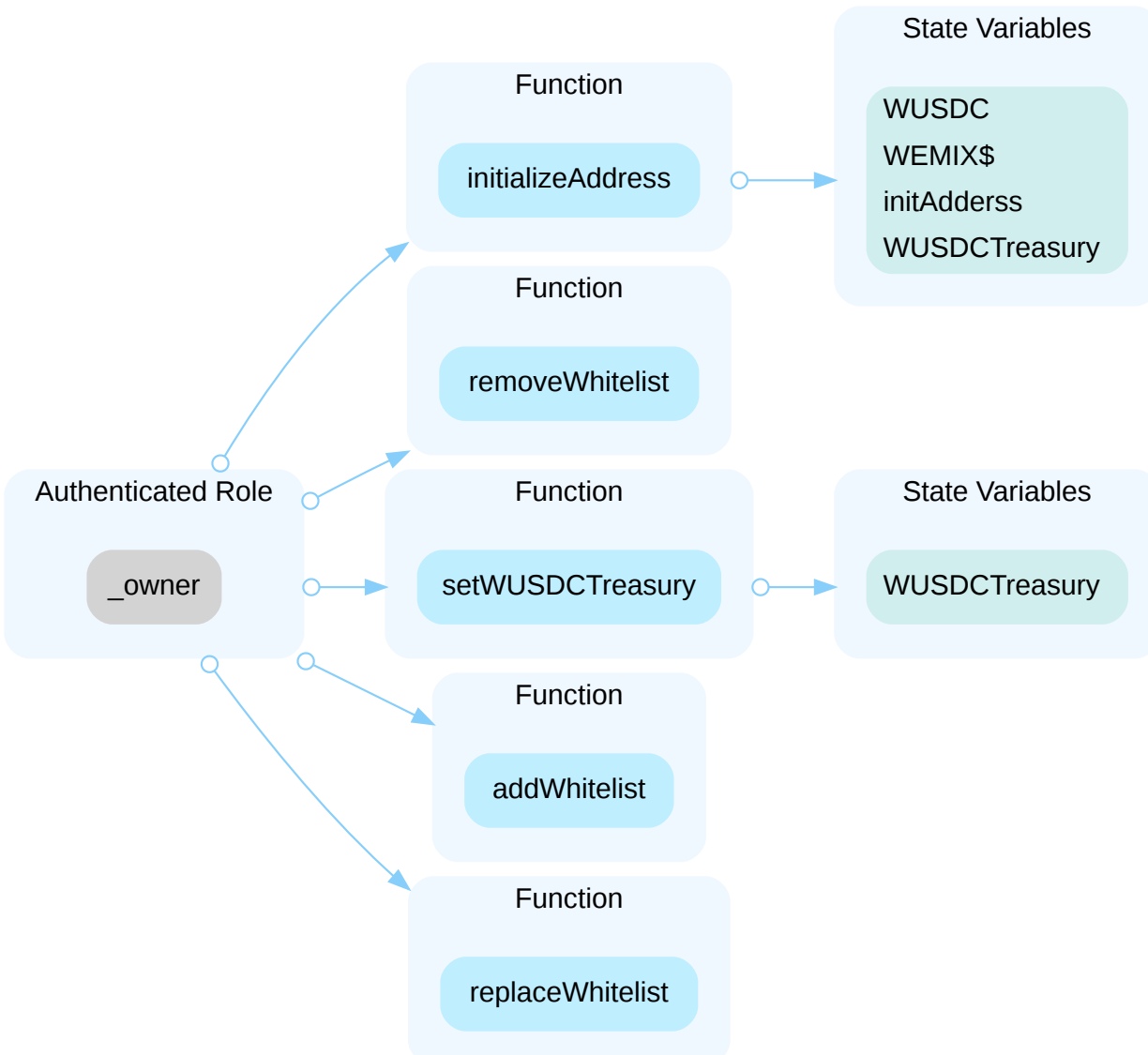
In the contract `WeswapFactory` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



In the contract `ZapBase` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.


Function Calls
owner
Function
withdrawTokens
Function Calls
IERC20
Function Calls
payable
Function
set_feeWhitelist
Function
set_affiliate
Authenticated Role
_owner
Function
setApprovedTargets
Function
set_new_affiliateSplit
State Variables
affiliateSplit
Function
toggleContractActive
State Variables
stopped
Function
set_new_goodwill
State Variables
goodwill

In the contract `WemixDollarExchange` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged this issue and they stated the following:

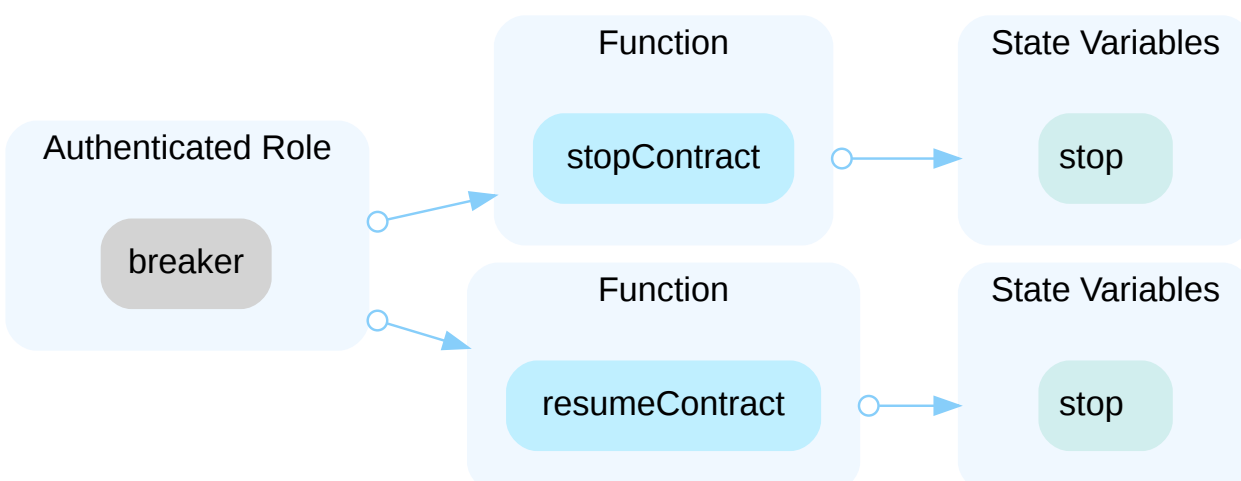
"They will adapt the multi-signature scheme to the contract owner's signature algorithm as a short-term solution and apply DAO as a long-term solution."

WEM-02 CENTRALIZATION RELATED RISKS IN THE `breaker` AND `breakerSetter` ROLE

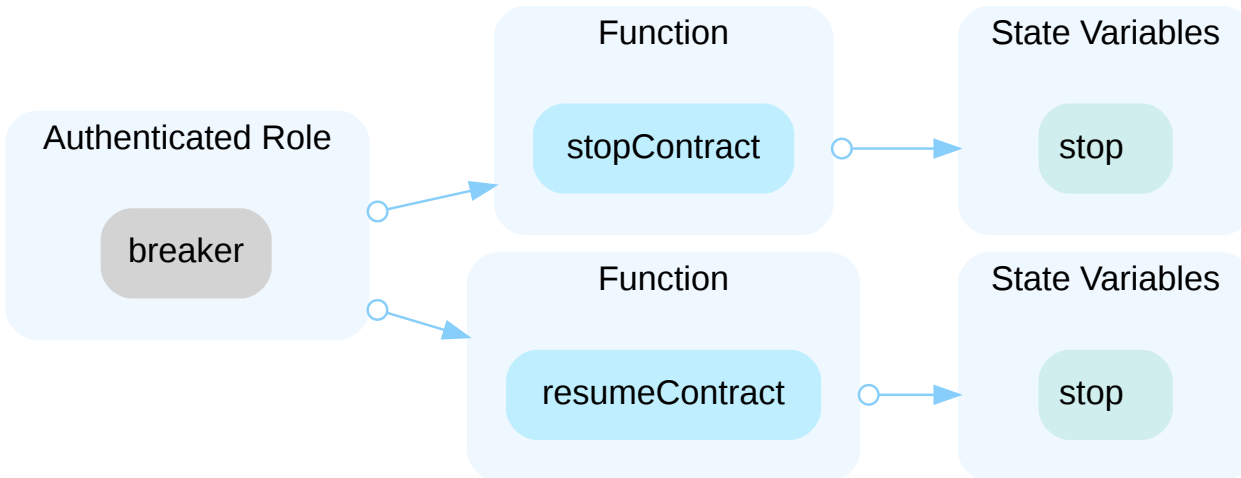
Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/DollarInAndOutStaking.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 191, 199; contracts/WemixDollarExchange.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 143, 151; contracts/WeswapFactory.sol (ad8c485d05a9701906dc4f58e949cd16080b0a23): 68; contracts/WeswapPair.sol (ad8c485d05a9701906dc4f58e949cd16080b0a23): 48	Acknowledged

Description

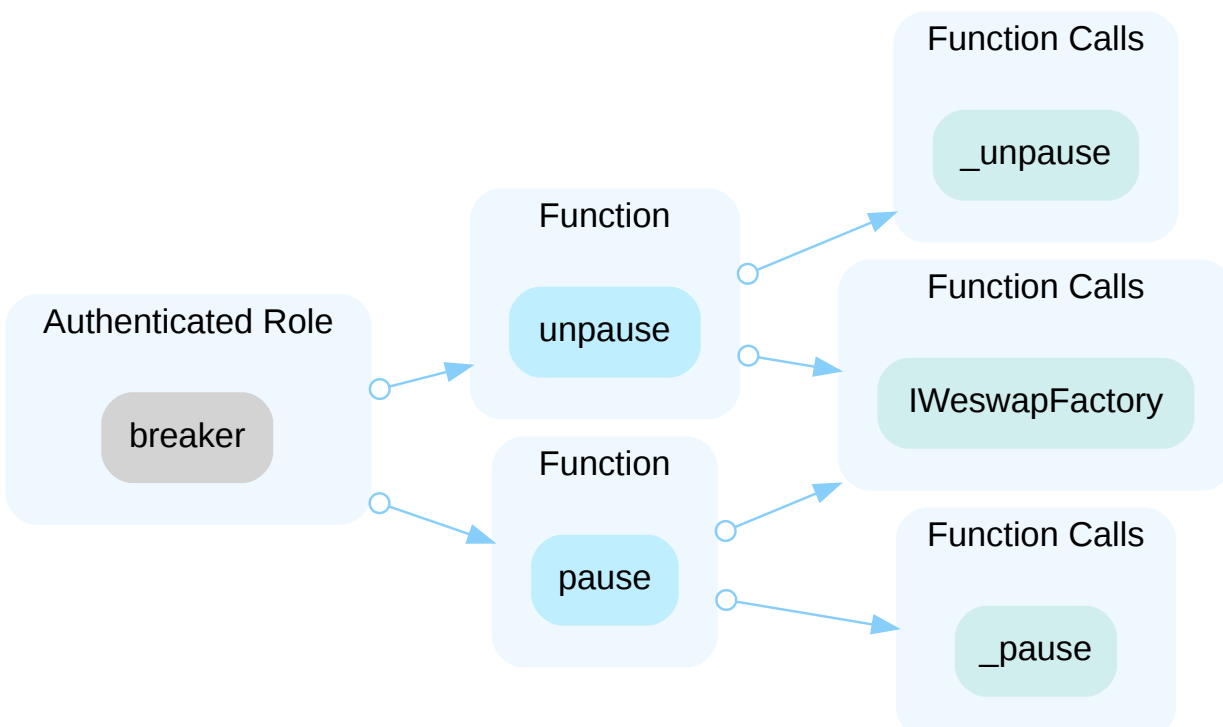
In the contract `DollarInAndOutStaking` the role `breaker` has authority over the functions shown in the diagram below. Any compromise to the `breaker` account may allow the hacker to take advantage of this authority.



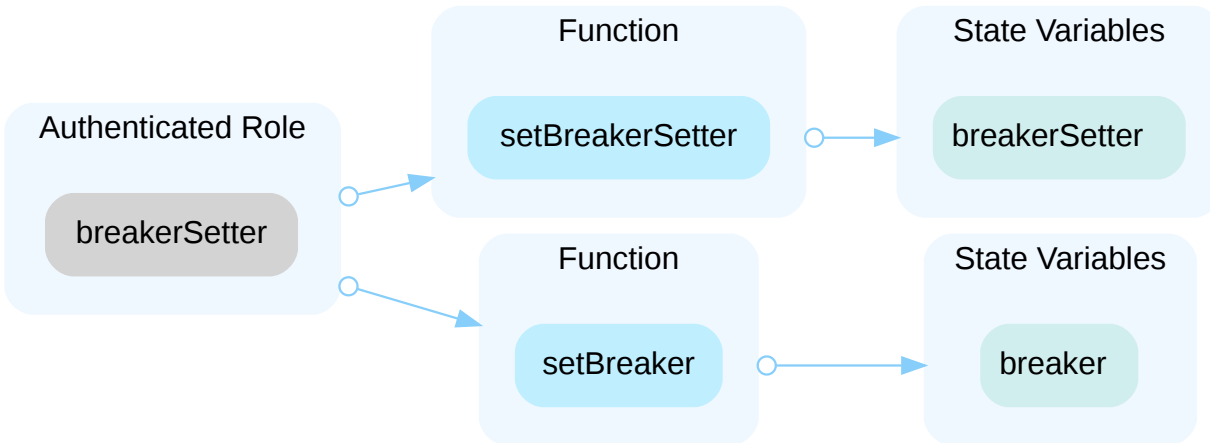
In the contract `WemixDollarExchange` the role `breaker` has authority over the functions shown in the diagram below. Any compromise to the `breaker` account may allow the hacker to take advantage of this authority.



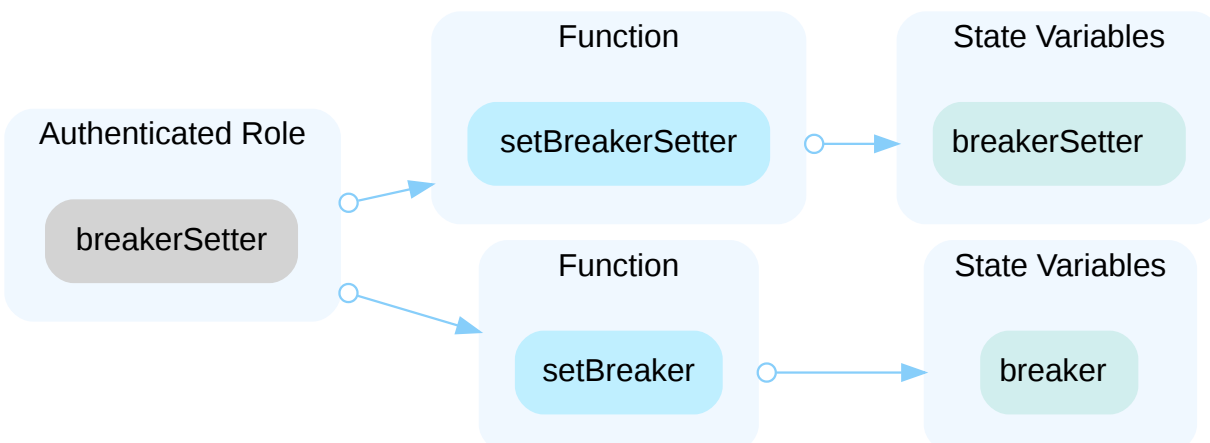
In the contract `WeswapPair` the role `breaker` has authority over the functions shown in the diagram below. Any compromise to the `breaker` account may allow the hacker to take advantage of this authority.



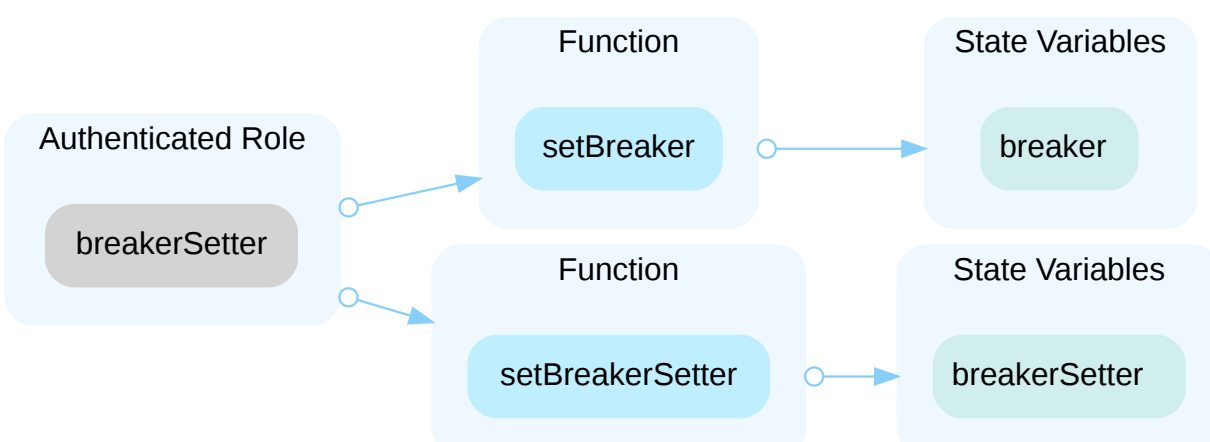
In the contract `DollarInAndOutStaking` the role `breakerSetter` has authority over the functions shown in the diagram below. Any compromise to the `breakerSetter` account may allow the hacker to take advantage of this authority.



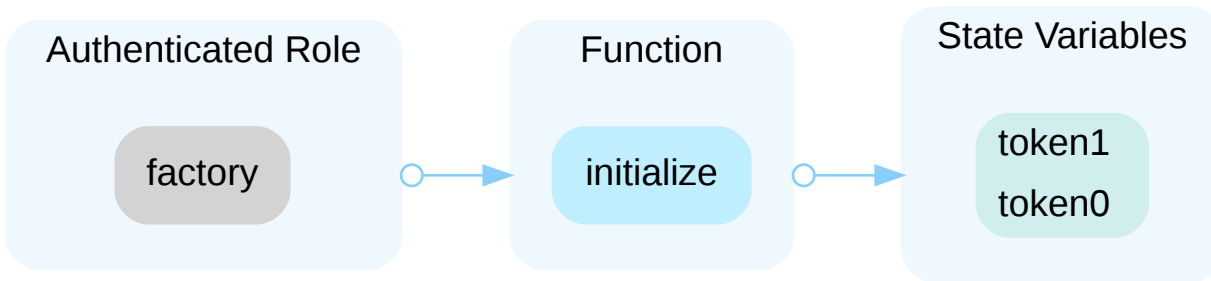
In the contract `WemixDollarExchange` the role `breakerSetter` has authority over the functions shown in the diagram below. Any compromise to the `breakerSetter` account may allow the hacker to take advantage of this authority and [fixme, describe what hacker can do and the impact].



In the contract `WeswapFactory` the role `breakerSetter` has authority over the functions shown in the diagram below. Any compromise to the `breakerSetter` account may allow the hacker to take advantage of this authority.



Also, in the contract `WeswapPair` the role `factory` has authority over the functions shown in the diagram below, however, this role would be granted to the `WeswapFactory` to make it work, which would lower the risk.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged this issue and they stated the following:

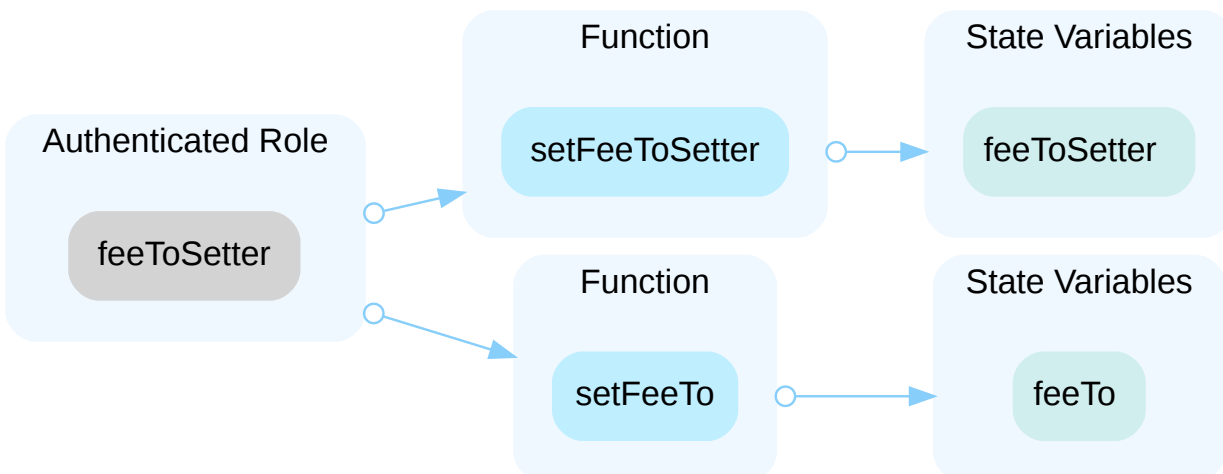
"They will adapt the multi-signature scheme to the contract owner's signature algorithm as a short-term solution and apply DAO as a long-term solution."

WFB-01 | CENTRALIZATION RISKS IN `feeToSetter` ROLE

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/WeswapFactory.sol (ad8c485d05a9701906dc4f58e949cd16080b0a23): 53**	**Acknowledged**

Description

In the contract `WeswapFactory` the role `feeToSetter` has authority over the functions shown in the diagram below. Any compromise to the `feeToSetter` account may allow the hacker to take advantage of this authority.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;

AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

The team acknowledged this issue and they stated the following:

"They will adapt the multi-signature scheme to the contract owner's signature algorithm as a short-term solution and apply DAO as a long-term solution."

WUS-01 | LOGICAL ISSUE OF `onlyWallet`

Category	Severity	Location	Status
Logical Issue	Medium	contracts/WUSDCTreasury.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 204~207	Resolved

Description

The modifier `onlyWallet` has such check:

```
require(msg.sender == address(this), "Treasury: Only Wallet can access.");
```

, which means the `msg.sender` should be the contract itself, however, the following related functions: `addOwner()`, `removeOwner()`, `replaceOwner()` and `changeQuorum()` are not externally called in this contract, as a result, these functions could never be called.

Recommendation

We advise the client to modify the aforementioned codes.

Alleviation

The team heeded our advice and resolved this issue in commit `d483b4b49d9e25bb4e9914743d24e6c6e9ad2540`.

WUS-02 | THE NUMBER OF `_quorum` AND `_owners.length`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/WUSDCTreasury.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 37~38	Resolved

Description

The `_quorum` should be larger than one to prevent one of the owners confirm and executing its own submitted transactions, besides, the `_owners.length` should be larger than two to guarantee it is a voting system.

Recommendation

We advise the client to modify the code as the aforementioned information.

Alleviation

The team heeded our advice and resolved this issue in commit `d483b4b49d9e25bb4e9914743d24e6c6e9ad2540`.

WUS-03 UNKNOWN IMPLEMENTATION WHEN CALLING `executeTransaction()`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/WUSDCTreasury.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 92	Acknowledged

Description

When calling `executeTransaction()`, the executing codes are passed by `transaction.data` and unknown to the audit scope. Besides, there are ETHs transferred to the `to` address, we would also like to know the source of the ETHs.

Recommendation

The scope of the audit treats unknown implementations as black boxes and assumes their functional correctness. However, in the real world, unknown implementations can lead to lost or stolen assets, hence we encourage the team to constantly monitor the statuses of these codes. These codes are not in the scope of this audit.

Alleviation

The team acknowledged this issue and added `payable` to the function `executeTransaction()`.

WVL-01 | DIVIDE BEFORE MULTIPLY

Category	Severity	Location	Status
Mathematical Operations	Minor	contracts/libraries/WeswapV2LiquidityMathLibrary.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 17, 19	Resolved

Description

Performing integer division before multiplication truncates the low bits, losing the precision of calculation.

```
17          uint256 invariant = reserveA * reserveB * truePriceTokenB /
truePriceTokenA;
```

```
19          uint256 leftSide = Babylonian.sqrt(53200 * invariant / 53067 + reserveA
* reserveA / 636804);
```

Recommendation

We recommend applying multiplication before division to avoid loss of precision.

Alleviation

The team heeded our advice and resolved this issue in commit `d483b4b49d9e25bb4e9914743d24e6c6e9ad2540`.

WZI-01 | THE EXISTENCE OF `_pairAddress` SHOULD BE CHECKED

Category	Severity	Location	Status
Logical Issue	Minor	contracts/WeswapZapIn.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 152	Resolved

Description

The existence of `_pairAddress` should be checked to ensure `token0` and `token1` are existing.

Recommendation

We advise the client to modify the code as the aforementioned information.

Alleviation

The team heeded our advice and resolved this issue in commit `4f9471a7d5d4a130d3a98115b0c8abdfa2b56725`.

WZI-02 | THE `_FromTokenBContractAddress` CHECK

Category	Severity	Location	Status
Logical Issue	Minor	contracts/WeswapZapIn.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 218~221	Acknowledged

Description

Given the `_FromTokenAContractAddress` is checked whether it is one of the paired tokens, we assume a check for `_FromTokenBContractAddress` should also be added to ensure logical completeness.

Recommendation

We would like to confirm with the client if this aligns with the original design.

Alleviation

The team acknowledged this issue and modified the code in commit `6314c5e26636436423c2de9a3bb5b84c730f558d` by adding a swap to the paired tokens if the `tokenA` and `tokenB` are not from the paired tokens.

ZBB-01 | NULL WALLET ADDDRESS

Category	Severity	Location	Status
Logical Issue	Minor	contracts/ZapBase.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 29	Acknowledged

Description

The `WEMIXAddress` as `0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE` is a null wallet address.

Recommendation

We would like to confirm with the client if this aligns with the original design.

Alleviation

The team acknowledged this issue and stated they used that address to express a native coin (ex, ETH) as a form of the token contract address.

ZBB-02 | INEFFECTIVE `isContract()` CHECK

Category	Severity	Location	Status
Volatile Code	Minor	contracts/ZapBase.sol (176869cbf42286369dccbb0875fbab51ee9211e7) : 195, 257	Resolved

Description

The implementation of the `isContract` check can not cover all scenarios. The check can be bypassed if the call is from the constructor of a smart contract or when the contract is destroyed. Because, in that case, the codesize will also be zero.

The "isContract" function in the OpenZeppelin "Address" library uses the same implementation, but comments mention that "it's unsafe to rely on the check and it can be bypassed". Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Address.sol

Recommendation

It is recommended to add the additional `msg.sender == tx.origin` check to cover all the scenarios. Do note that the check still works for the current EVM (London) version, but future updates to the EVM or EIP (ex. EIP-3074) might cause the check to become ineffective.

```
modifier notContract() {
    require((!_isContract(msg.sender)) && (msg.sender == tx.origin), "contract not allowed");
    _;
}

function _isContract(address addr) internal view returns (bool) {
    uint256 size;
    assembly {
        size := extcodesize(addr)
    }
    return size > 0;
}
```

Alleviation

The team heeded our advice and resolved this issue in commit `af9344ff4c5c2f3abbd77fba65f416a9ee01cc3b`.

COR-03 | MISSING INHERITANCE

Category	Severity	Location	Status
Language Specific	Informational	contracts/interfaces/IWWEMIX.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 6; contracts/tokens/WWEMIX.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 7	Resolved

Description

`WWEMIX` implements the interface `IWWEMIX`, but does not inherit from it.

```
7  contract WWEMIX is ERC20 {
```

```
6  interface IWWEMIX is IERC20 {
```

Recommendation

Consider inheriting from the missing interface or contract.

Alleviation

The team heeded our advice and resolved this issue in commit `a8063b9020d45fc103741d4666e08d4e5083add3`.

DIA-02 | CODE COMMENTS IN TWO LANGUAGES

Category	Severity	Location	Status
Coding Style	Informational	contracts/DollarInAndOutStaking.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 102	Resolved

Description

Code commented in both Korean and English.

Recommendation

We recommend commenting the code in one language for better readability and maintainability.

Alleviation

The team heeded our advice and resolved this issue in commit `d483b4b49d9e25bb4e9914743d24e6c6e9ad2540`.

DIA-03 | UNUSED EVENT

Category	Severity	Location	Status
Coding Style	Informational	contracts/DollarInAndOutStaking.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 479	Resolved

Description

```
 479     event ExecuteArbitrageSwapWithPrice(uint256[] amounts, address[] arbPath,
bool checkArbState, uint256 feePoolAmount, uint256 stabilityPoolAmount);
```

- `ExecuteArbitrageSwapWithPrice` is declared in `DollarInAndOutStaking` but never emitted.

Recommendation

We advise removing the unused events or emitting them in the intended functions.

Alleviation

The team heeded our advice and resolved this issue in commit `d0ad5da4be38920704da239eba66ad4f770e6453`.

WEM-03 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	Informational	contracts/WeswapFactory.sol (ad8c485d05a9701906dc4f58e949cd16080b0a23): 53, 58, 63, 68, 85, 93; contracts/WeswapPair.sol (ad8c485d05a9701906dc4f58e949cd16080b0a23): 48, 61, 102; contracts/WeswapRouter.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 25; contracts/ZapBase.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 75, 79, 86, 94, 105, 113, 148	Partially Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

The team heeded our advice and fixed this issue of the contract `WeswapFactory` in commit `54c854660fca99a6e3b64689cc19b1702696027f`.

WEM-04 | MATHEMATICAL CALCULATIONS

Category	Severity	Location	Status
Mathematical Operations	Informational	contracts/libraries/WeswapV2LiquidityMathLibrary.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 18~42; contracts/WeswapZapIn.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 449~474	Acknowledged

Description

Please provide with us more documentation about the design of these mathematical calculations.

Recommendation

Mathematical verifications are not in the scope of this audit.

Alleviation

The team explained these calculations as follows:

"The equation in `WeswapV2LiquidityMathLibrary.sol` calculates the maximum arbitrage profit and returns `amountIn`, which is the optimal input at the pair.

The equation in `WeswapZapIn.sol` calculates a Generalized ZapIn that adds liquidity to the swap pool with arbitrary amounts of two-pair assets. This equation also contains a fee of the deduction for swapping. 'reserveInA' and 'reserveInB' are the amounts of 'tokenA' and 'tokenB' in the pool, respectively, before adding liquidity. 'userInA' and 'userInB' are the amounts of user input of 'tokenA' and 'tokenB', respectively, which are used to add liquidity after swapping to match the pool's ratio. This function returns how many tokens should be swapped."

WFB-02 | MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	Informational	contracts/WeswapFactory.sol (ad8c485d05a9701906dc4f58e949cd16080b0a23): 77	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

The team heeded our advice and resolved this issue in commit `daaccb9b0cf901675b1a302be5ca7671f2113402`.

WUS-04 | UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	Informational	contracts/WUSDCTreasury.sol (f8ad5b7ba7549159377663f86aaa61c06e975227): 2	Resolved

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

Alleviation

The team heeded our advice and resolved this issue in commit `3d4fc9eef7c9884bcdeec74d294d78b22fb6a52f`.

WZI-03 | EXPLANATION ON THE USE OF `_SWAPTARGET

Category	Severity	Location	Status
Logical Issue	Informational	contracts/WeswapZapIn.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 292	Acknowledged

Description

In the function `_fillQuote()` , the `_swapTarget` address is used to receive ETH and execute some external functions which are unknown, could you please provide with us more information about the `_swapTarget` ?

Recommendation

We would like the client to provide with us more information.

Alleviation

The team acknowledged this issue and they stated the following:

"_swapTarget `and` swapData `make Zap more general. If` _FromTokenContractAddress `does not match with either token0 or token1,` _FromTokenContractAddress `is swapped into` intermediateToken `using` _swapTarget `and` swapData `. In addition, it converts ETH to WETH as` intermediateToken`."

OPTIMIZATIONS

WEMIX SWAP & DIOS(DOLLAR IN AND OUT STABILIZER)

ID	Title	Category	Severity	Status
COT-01	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	Resolved

COT-01 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/WeswapRouter.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 12, 13; contracts/WeswapZapIn.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 21; contracts/WeswapZapOut.sol (176869cbf42286369dccbb0875fbab51ee9211e7): 19	Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

The team heeded our advice and resolved this issue in commit `69ef1e7354619f36eff206e82c61900c4aa537f4`.

APPENDIX | WEMIX SWAP & DIOS(DOLLAR IN AND OUT STABILIZER)

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Mathematical Operations	Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

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 CertiK'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 CertiK 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. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK'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 CertiK 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 AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK 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, CERTIK 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, CERTIK 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 CERTIK NOR ANY OF CERTIK'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. CERTIK 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 CERTIK'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 MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK 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 CERTIK 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.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.