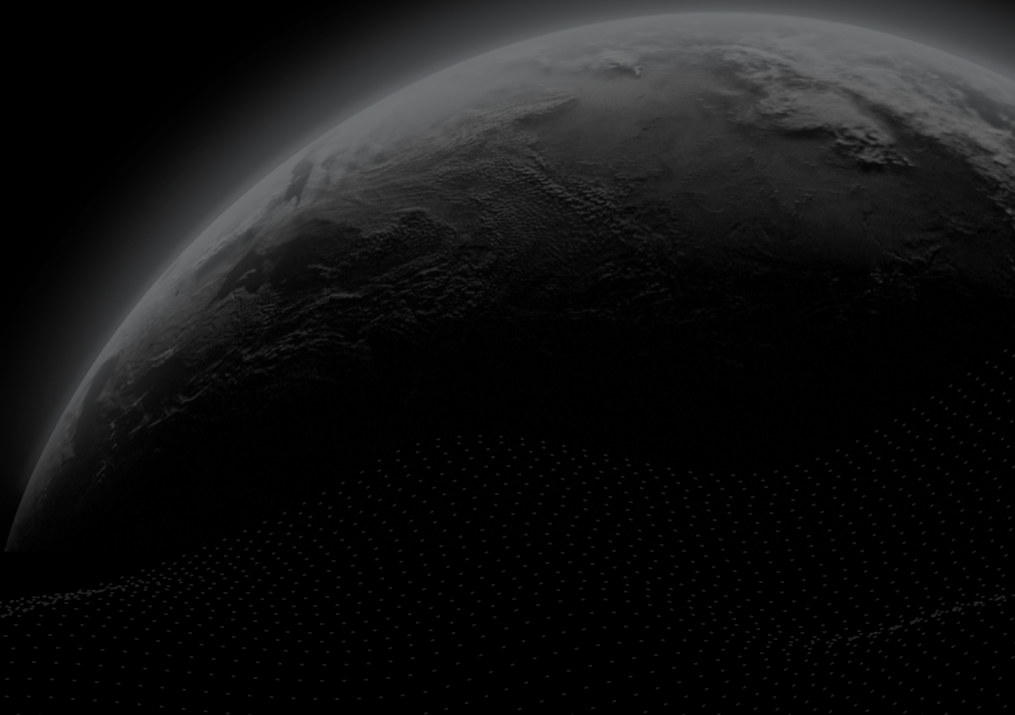




Security Assessment

**VirtuSwap**

CertiK Verified on Sept 14th, 2022





CertiK Verified on Sept 14th, 2022

## VirtuSwap

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

### Executive Summary

#### TYPES

Trading-AMM

#### ECOSYSTEM

Ethereum

#### METHODS

Manual Review, Static Analysis

#### LANGUAGE

Solidity

#### TIMELINE

Delivered on 09/14/2022

#### KEY COMPONENTS

N/A

#### CODEBASE

<https://github.com/Virtuswap/v1-core>

[...View All](#)

#### COMMITTS

[cbdeff22e79907490e68746c8bebf34317b172c](https://github.com/Virtuswap/v1-core/commit/cbdeff22e79907490e68746c8bebf34317b172c)

[...View All](#)

### Vulnerability Summary



13

Total Findings

11

Resolved

0

Mitigated

0

Partially Resolved

2

Acknowledged

0

Declined

0

Unresolved

#### 1 Critical

1 Resolved



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.

#### 3 Major

2 Resolved, 1 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.

#### 0 Medium

Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

#### 3 Minor

3 Resolved



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.

#### 6 Informational

5 Resolved, 1 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 | VIRTUSWAP

## I Summary

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

## I Findings

[CON-01 : Centralization Related Risks](#)

[CON-02 : Missing Override Specifier](#)

[CON-03 : Missing Input Validation](#)

[CON-06 : Missing Emit Events](#)

[CON-07 : Unlocked Compiler Version](#)

[PVU-01 : No validation check on `tokenOut` address](#)

[PVU-02 : swap functions use `msg.sender` as the contract address for `lvFlashSwapCallback`](#)

[RVT-01 : `vRouter` Does Not Guard Against MEV Attacks](#)

[RVT-02 : Discussion on `vRouter` contract `vFlashSwapCallback\(\)` function](#)

[RVT-03 : Missing Error Messages](#)

[RVT-04 : token unspecified in `quote\(\)` input](#)

[RVT-05 : Typo in Comment](#)

[SEC-01 : Function `\\_transfer\(\)` should be internal](#)

## I Optimizations

[CON-04 : Improper Usage of `public` and `external` Type](#)

[CON-05 : Memory Used over Calldata](#)

[RVH-01 : Variables That Could Be Declared as Immutable](#)

[SEC-02 : Variables That Could Be Declared as `constant`](#)

## I Appendix

## I Disclaimer

# CODEBASE | VIRTUSWAP

## Repository















<https://github.com/Virtuswap/v1-core>
















## Commit

[cbdeff22e79907490e68746c8bebf34317b172c](#)

## AUDIT SCOPE | VIRTUSWAP

29 files audited ● 3 files with Acknowledged findings ● 12 files with Resolved findings ● 14 files without findings

ID	File	SHA256 Checksum
● PVU	 contracts/vPair.sol	12ff7d6298f2b756c885ba6ff1c193da4f5b491a4549e39cf2bd8afb97be14ff
● VPA	 contracts/vPairFactory.sol	4fa56eae854512cec61bcc60abe27cc6394155b079812c569b68afc4d0978140
● RVT	 contracts/vRouter.sol	6d62973c86894e6bf64fc32199191a07c567b7c54d8c1250dc9119500bf99ede
● MUL	 contracts/base/multicall.sol	364756d1e1a41b0d233f58ea0e05640a1e6af2d15f378962292dff0266faf488
● IMV	 contracts/interfaces/IMulticall.sol	05bbba2d7ea2e7465548fc05a39e7ba2bd0c5fcb8e22a13a9907c714814ef802
● IFC	 contracts/interfaces/IvFlashSwapC allback.sol	cac1bf0dc79ea5e01d509aa359a078f21a40e4c706e19cc2ffd33315ca2b21f6
● IVP	 contracts/interfaces/IvPair.sol	85c5b6fcc11a655f85a240faa678cd5727e280210595dc3f776787271af051cb
● IFV	 contracts/interfaces/IvPairFactory.s ol	95751b65527185f4eb0f9a6bf69718bad6dd89e3c7e0a379949b8ec83ddb2b77
● IVR	 contracts/interfaces/IvRouter.sol	992ffada03855a09b3fa021757130158a2ecd6f6a8872d0bf871a54348b43850
● ISD	 contracts/interfaces/IvSwapPoolDe ployer.sol	e10e469831f99a446f0315277562b59fc584e9f6f86ea381d597a5353f08618f
● AVB	 contracts/libraries/poolAddress.sol	804bccd51cdd5dc7a5a2c91deefee2904fa03af8b921cc3f6ac7e25c5e64993c
● VSW	 contracts/libraries/vSwapLibrary.sol	519a6301a10eacc755018bc7d10d176a7ea9499547e73b5b375971ca7b85cb03
● RVH	 contracts/exchangeReserves.sol	388883f2517174a1272ba6f4fb8383b3e53f3cf31d2c7bcbe3f43a44367f0ce2
● TYE	 contracts/types.sol	912a98f2cb2769fee9c1dc6e764c9be499037c62c28c23e16d685986d781ecfb

ID	File	SHA256 Checksum
● SEC	 contracts/vSwapERC20.sol	979664996149f507c70158700af3d50000f1d96cde15601799ce1b17efc6e206
● IFS	 contracts/interfaces/IvFlashSwapCallback.sol	e12e98f2fe70091b5b3303060a1593deacc743cd4ac0d7e987a2127e8efac42
● IPV	 contracts/interfaces/IvPair.sol	1db84608eb8db7a2513572f79ea9aeebabd06eb83a6c197605fc65d51d02c2f3
● IPF	 contracts/interfaces/IvPairFactory.sol	51f9893c06eea4bd2c32d4007c4742c3116e1935edb2f950c058481f3512f664
● IRV	 contracts/interfaces/IvRouter.sol	8705d10a74efb65c1903d4d13c1f4a304f6d77fad07bd185a9da6c37385b1ccb
● ISP	 contracts/interfaces/IvSwapPoolDeployer.sol	29925d7ca180c07c11a325aa3b6147ec2433bcf0acb19182adfc1c10b5b7699a
● PAV	 contracts/libraries/PoolAddress.sol	e5da5972f8f72a9252acd146c27659e8271d8962fdd66f0fdc3105f8130d05d6
● SLV	 contracts/libraries/vSwapLibrary.sol	8778a59e2fac00554af8438f873b1f6211c0841a80fbfe438e2b3d3c73477bb1
● RVB	 contracts/exchangeReserves.sol	66407676e3babb7a66f698f08d56c82ea4e94d5c64ace291325c3dd1f798d4b2
● TYP	 contracts/types.sol	618f24493b5e7f7c4ed24379323bbb9e7a671a276c57c42453756c9faf9ab8d0
● PVB	 contracts/vPair.sol	39da31533d4ff541256ee0d05ad29d3f3322b971f81066bf886d83d4ab13f411
● PFV	 contracts/vPairFactory.sol	c6c9901f5f47b22afce45d6f9d0706a99e95624912814e6f1447db56d224b872
● RVU	 contracts/vRouter.sol	0e3494778e8f3d178edb099ed758bd9e80ca7e579e34069ac8fb92e6960e1048
● SER	 contracts/vSwapERC20.sol	979664996149f507c70158700af3d50000f1d96cde15601799ce1b17efc6e206
● SPD	 contracts/vSwapPoolDeployer.sol	316e872ef4772079ce74888c99d8986aa7dddb2977c5747a32e83960d09b845b

## APPROACH & METHODS | VIRTUSWAP

This report has been prepared for VirtuSwap to discover issues and vulnerabilities in the source code of the VirtuSwap 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 | VIRTUSWAP



13

Total Findings

1

Critical

3

Major

0

Medium

3

Minor

6

Informational

This report has been prepared to discover issues and vulnerabilities for VirtuSwap. Through this audit, we have uncovered 13 issues ranging from different severity levels. Utilizing Static Analysis techniques to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
<u>CON-01</u>	Centralization Related Risks	Centralization / Privilege	Major	● Acknowledged
<u>CON-02</u>	Missing Override Specifier	Compiler Error	Minor	● Resolved
<u>CON-03</u>	Missing Input Validation	Volatile Code	Minor	● Resolved
<u>CON-06</u>	Missing Emit Events	Coding Style	Informational	● Resolved
<u>CON-07</u>	Unlocked Compiler Version	Language Specific	Informational	● Resolved
<u>PVU-01</u>	No Validation Check On <code>tokenOut</code> Address	Logical Issue, Volatile Code	Major	● Resolved
<u>PVU-02</u>	Swap Functions Use <code>msg.sender</code> As The Contract Address For <code>IvFlashSwapCallback</code>	Language Specific	Informational	● Acknowledged
<u>RVT-01</u>	<code>vRouter</code> Does Not Guard Against MEV Attacks	Control Flow	Major	● Resolved
<u>RVT-02</u>	Discussion On <code>vRouter</code> Contract <code>vFlashSwapCallback()</code> Function	Logical Issue	Minor	● Resolved
<u>RVT-03</u>	Missing Error Messages	Coding Style	Informational	● Resolved
<u>RVT-04</u>	Token Unspecified In <code>quote()</code> Input	Coding Style	Informational	● Resolved

ID	Title		Category	Severity	Status
<u>RVT-05</u>	Typo In Comment		Coding Style	Informational	● Resolved
<u>SEC-01</u>	Function	<code>_transfer()</code> Should Be Internal	Logical Issue	Critical	● Resolved

## CON-01 | FINDING DETAILS

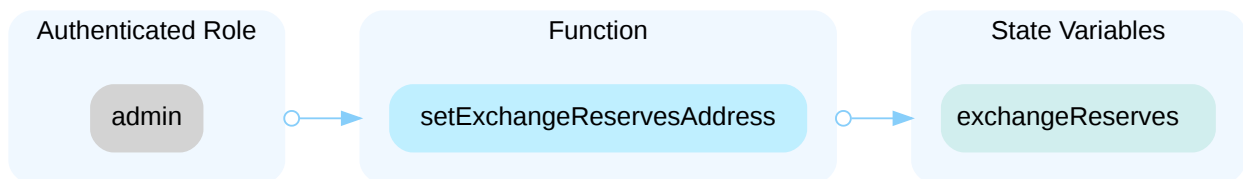
### Finding Title

Centralization Related Risks

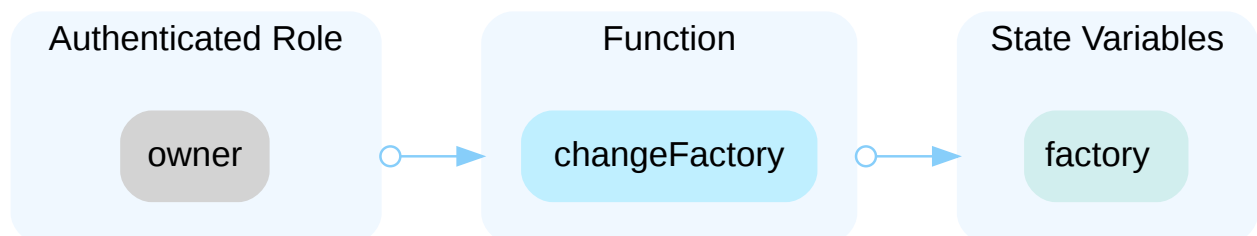
Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/vPair.sol: 421, 438, 442, 451, 459; contracts/vPairFactory.sol: 76; contracts/vRouter.sol: 341	Acknowledged

### Description

In the contract `vPairFactory` the role `admin` has authority over the functions shown in the diagram below, as well as authority over privileged functions referenced in the `vPair` contract. Any compromise to the `admin` account may allow the hacker to take advantage of this authority and set the address `exchangeReserves` to any address. Additionally, in the contract `vPair`, this may also allow the hacker to change whitelisted addresses, set the address `factory` to any address, and change the value of fee, max reserve ratio and max whitelist count.



In the contract `vRouter` 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 and set the address `factory` to any address.



### 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 ( $\frac{2}{3}$ ,  $\frac{3}{5}$ ) 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

[VirtuSwap] : "Agreed to postpone until mainnet deployment."

[Certik] : The client acknowledges the finding and is working to mitigate the risk through multi-signature wallets. At such a time that all features of the short-term recommendation are followed, the finding will be considered mitigated.

## CON-02 | FINDING DETAILS

### Finding Title

Missing Override Specifier

Category	Severity	Location	Status
Compiler Error	Minor	contracts/vPair.sol: 40; contracts/vPairFactory.sol: 76, 84; contracts/vRouter.sol: 80, 91	Resolved

### Description

The contract `vPair` declares the public state variable `reserves` and inherits from the interface `IvPair`, in which a read function for the variable `reserves` is declared, but this overriding public state variable `reserves` is missing `override` specifier.

In addition, the function `setExchangeReservesAddress()` and `getInitCodeHash()` in the contract `vPairFactory` and the function `swapToExactNative()` and `swapReserveToExactNative()` in the contract `vRouter` are also lacking the `override` specifier.

### Recommendation

We recommend the client add `override` to the overriding state variable and functions.

### Alleviation

[VirtuSwap]: Issue acknowledged. Changes have been reflected in the commit hash [4f3676a58ee085665d74ee7f75b47eff960cb040](#) and [a8c4465105981880c483fddd0f5e5f183ac1ab15](#). The function `getInitCodeHash` is temporary and will be removed after the audit.

## CON-03 | FINDING DETAILS

### Finding Title

Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	contracts/vPair.sol: 102, 104, 147, 149, 229, 231, 439; contracts/vRouter.sol: 341	Resolved

### Description

The following input is missing the check for the non-zero address.

```
438     function setFactory(address _factory) external onlyFactoryAdmin {
439         factory = _factory;
440     }
```

```
341     function changeFactory(address _factory) external override onlyOwner {
342         factory = _factory;
343     }
```

The following input `amountOut` is missing the check for value greater than 0 and less than reserve.

```
101     function swapNative(
102         uint256 amountOut,
```

```
146     function swapNativeToReserve(
147         uint256 amountOut,
```

```
228     function swapReserveToNative(
229         uint256 amountOut,
```

The following input `to` is missing the check that address does not equal to `token0` or `token1`.

```
104         address to,
```

```
149         address to,
```

```
231         address to,
```

## Recommendation

We recommend adding the check for the passed-in values to prevent unexpected errors.

## Alleviation

[Certik]: The client heeded the advice and made the changes corresponding to the recommendations listed above in commit [a5b26a44a067eef974370af44b7eed53df0b9e7e](#).

## CON-06 | FINDING DETAILS

### Finding Title

Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	contracts/vPair.sol: 438, 442, 451, 459; contracts/vPairFactory.sol: 7 6; contracts/vRouter.sol: 341	● Resolved

### Description

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

### Recommendation

We recommend emitting events for the sensitive functions that are controlled by centralization roles.

### Alleviation

[VirtuSwap] : Issue acknowledged. Changes have been reflected in the commit hash [1d3bbbcec5c084bdc158b004f1ea1ecc9ae24dbb](#).

## CON-07 | FINDING DETAILS

### Finding Title

Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	● Informational	contracts/base/multicall.sol: 3; contracts/exchangeReserves.sol: 2; contracts/interfaces/IMulticall.sol: 2; contracts/interfaces/IvFlashSwapCallback.sol: 1; contracts/interfaces/IvPair.sol: 1; contracts/interfaces/IvPairFactory.sol: 1; contracts/interfaces/IvRouter.sol: 1; contracts/interfaces/IvSwapPoolDeployer.sol: 1; contracts/libraries/poolAddress.sol: 2; contracts/libraries/vSwapLibrary.sol: 1; contracts/types.sol: 1; contracts/vPair.sol: 1; contracts/vPairFactory.sol: 1; contracts/vRouter.sol: 2; contracts/vSwapERC20.sol: 4	● Resolved

### Description

The contracts listed have an 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 recommend the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.2` the contract should contain the following line:

```
pragma solidity 0.8.2;
```

### Alleviation

[VirtuSwap]: Issue acknowledged. Changes have been reflected in the commit hash [87b54bcd77060c73be257b39208ab26094c4ea73](#) and [a8c4465105981880c483fddd0f5e5f183ac1ab15](#).

## PVU-01 | FINDING DETAILS

### Finding Title

No Validation Check On `tokenOut` Address

Category	Severity	Location	Status
Logical Issue, Volatile Code	● Major	contracts/vPair.sol: 101~102	● Resolved

### Description

The `swapNative()` function is meant to be used with the two tokens that compose the pair within the `vPair` instance. However, any address could be used as input for `tokenOut`, and, if the pair contract has a balance of that token, the function will successfully execute. Since this function updates reserve values for `token0` and `token1` based upon the `amountOut` value corresponding to the input `tokenOut` address, this could cause an incorrect calculation for the token reserves of the contract, which is subject to exploit.

### Recommendation

We recommend the client require that the `tokenOut` address for `swapNative()` be the address of either `token0` or `token1`.

### Alleviation

[VirtuSwap]: Issue acknowledged. Changes have been reflected in the commit hash [f3501107b07014f0896c944819fbe70642cd5c4b](#).

## PVU-02 | FINDING DETAILS

### Finding Title

Swap Functions Use `msg.sender` As The Contract Address For `IvFlashSwapCallback`

Category	Severity	Location	Status
Language Specific	● Informational	contracts/vPair.sol: 128~129, 183~184, 263~264	● Acknowledged

### Description

It is understood that `msg.sender` is used as the contract address for the interface `IvFlashSwapCallback` so that, in the event a call is made to any one of the referenced swap functions through the router contract, the router is the `msg.sender` and the function `vFlashSwapCallback` is called within the router. However, for any user that is not calling through the router to interact with a pairs contract, the specification of `msg.sender` for the contract address is limiting. This implementation requires that anyone choosing to use the flash swap utility must construct a function to call to the pair within the same flash swap contract, rather than allowing the user to make a call to the pair's swap function either directly or with a separate contract.

### Recommendation

We recommend the client consider modifying the pair contract's swap functions to accommodate a specified address used in the `IvFlashSwapCallback` interface, instead of using `msg.sender` as a default.

### Alleviation

[Certik]: The client acknowledges the finding and opts to make no changes.

[VirtuSwap]: "I intend to continue to use `msg.sender` as the address to invoke `vFlashSwapCallback`.

As stated, it may be limiting, but I see it as a safety restriction.

It is the user's responsibility to implement the `IvFlashSwapCallback` interface when interacting directly with the `vPair` contract for a flashswap. In the callback function, he can call other contracts and perform any logic he wants. Different types of errors may result from keeping it for user decisions.

UniswapV3 also hard-coded `msg.sender` for swap callbacks."

## RVT-01 | FINDING DETAILS

### Finding Title

`vRouter` Does Not Guard Against MEV Attacks

Category	Severity	Location	Status
Control Flow	● Major	contracts/vRouter.sol: 80–81, 91	● Resolved

### Description

The flow of a swap function in a DEX's router contract usually includes:

- giving the router approval to the user's token funds first,
- the router safe-transferring the tokens from the user to the pair contract, and
- the router function calling the pair's swap function immediately after the transfer.

Since the transfer to the pair and the swap function call are made in the same atomic transaction, the function call safely swaps tokens using the pair contract. In general it is considered dangerous for an externally-owned account (not using the flash swap callback logic) to interact directly with the pair contract, the primary reason being a user would need to sequentially transfer tokens to the pair contract and then call the swap function. In the time between these two transactions, an MEV bot can listen for the call to the swap function and front run the call. Since the successful execution of the swap function is not dependent upon who first deposits tokens into the contract, the original user will lose anything they contribute in this instance.

A router's functions for swapping, adding, and removing liquidity should both include a transfer of funds from the user to the pair contract, and a call to the pair's respective functions. This composition keeps end users safe from MEV bot front-running attacks. The swap functions in `vRouter` only includes such logic if a user includes nonempty `data` in bytes that is used in the router's `vFlashSwapCallback()` function to transfer tokens to the pair contract. As constructed, a user may interact with the router's swap functions in an unsafe manner and lose funds to MEV bot attacks.

### Recommendation

We recommend the client add into the cited router functions logic for safely transferring tokens to the pair contract that does not rely on the use of the `vFlashSwapCallback()` function.

### Alleviation

[Certik]: The client made changes to allow for users to complete atomic trades without requiring encoded data as input. Changes were completed in commit [cbdeff22e79907490e68746c8bebf34317b172c](#).

[VirtuSwap]: Here is a description of our atomic trade flow

- (Off-chain client side) Caller approves contract to spend tokens
- Caller invokes the router's Swap functions (swapReserveToExactNative/swapReserveToExactNative)
- The equivalent vPair swap function is invoked by the router contract with populated data parameters to invoke flashswap.
- vPair optimistically sends the output funds to the caller in case of ERC20 and to the router in case of WETH.
- vPair invokes flash callback to receive funds back, then the router collects funds from the caller (wrap for WETH).
- The vPair validates that there are enough funds back.
- vRouter unwraps WETH to ETH in case it is needed and settles the transaction.

#### Changes:

I have removed requirement for caller to provide bytes data

Now the data param is structured internally in vRouter and converted to bytes for vPair.

Instead of copying the token collection from the user into every swap function, it can be unified into a single event and remain an atomic transaction safe from MEV by keeping the vFlashCallback in the router.

As you stated it is not safe to call the vPair contract directly without a periphery contract to wrap the funds collection.

If an advanced user is directly interacting with the vPair contract, they should be aware of such risks before depositing funds.

Our UI only interacts with the vRouter contract.

## RVT-02 | FINDING DETAILS

### Finding Title

Discussion On `vRouter` Contract `vFlashSwapCallback()` Function

Category	Severity	Location	Status
Logical Issue	Minor	contracts/vRouter.sol: 52~53	Resolved

### Description

The flash swap callback feature of a typical pair contract intentionally uses an input of `data` in bytes because the use of the flash swap callback is ambiguous and left to the user to decide how the funds are applied and then paid back. The utility within the `vRouter` contract for function `vFlashSwapCallback()` is well-defined and only serves to pay back the amount of tokens owed to the pair contract. The logic for transferring tokens from the `msg.sender` to the pair should be included in the body of each router swap function. As is, the set up requires that a user encode this information first so it may be included in the router swap call, allowing the router to assist in paying back the pair contract through the function `vFlashSwapCallback()`. A separate finding has already established a need for using `safeTransferFrom` to handle moving tokens from the `msg.sender` to the pair contract within the swap functions of the router. Once that is completed, the purpose of using the router for flash swap callbacks may no longer be necessary.

### Recommendation

We recommend the client consider the information above and decide whether removing the function `vFlashSwapCallback()` may fit their intentions for the project with respect to the other changes made to the `vRouter` contract.

### Alleviation

[Certik]: The client made changes to allow for users to complete atomic trades without requiring encoded data as input. Changes were completed in commit [cbdeff22e79907490e68746c8bebf34317b172c](#).

[VirtuSwap]: Here is a description of the atomic trade flow:

- (Off-chain client side) Caller approves contract to spend tokens
- Caller invokes the router's Swap functions (`swapReserveToExactNative/swapReserveToExactNative`)
-

The equivalent vPair swap function is invoked by the router contract with populated data parameters to invoke flashswap.

- vPair optimistically sends the output funds to the caller in case of ERC20 and to the router in case of WETH.
- vPair invokes flash callback to receive funds back, then the router collects funds from the caller (wrap for WETH).
- The vPair validates that there are enough funds back.
- vRouter unwraps WETH to ETH in case it is needed and settles the transaction.

#### Changes:

- The requirement for caller to provide bytes data has been removed
- Now the data parameter is structured internally in vRouter and converted to bytes for vPair.
- Instead of copying the token collection from the user into every swap function, it can be unified into a single event and remain an atomic transaction safe from MEV by keeping the vFlashCallback in the router.

## **RVT-03** | FINDING DETAILS

### **Finding Title**

Missing Error Messages

Category	Severity	Location	Status
Coding Style	● Informational	contracts/vRouter.sol: 22	● 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**

[VirtuSwap] : Issue acknowledged. Changes have been reflected in the commit hash [6ad371ea413c1ac1626e97e7b8c5a0e7874cd76c](#).

## RVT-04 | FINDING DETAILS

### Finding Title

Token Unspecified In `quote()` Input

Category	Severity	Location	Status
Coding Style	● Informational	contracts/vRouter.sol: 279~280	● Resolved

### Description

Within the function `quote()` in the `vRouter` contract, it is not immediately apparent whether the input `amount` corresponds to `token0` or `token1` of the corresponding pair, until viewing the set up for input of the `quote()` function in the `vSwapLibrary` contract. Since this function is user-facing, code readability may be improved by specifying which token the uint `amount` corresponds to within the function.

### Recommendation

We recommend changing the naming of the uint256 value `amount` in the `quote()` function within `vRouter` so the user has a clear understanding of what value should be used as input within the function.

### Alleviation

`[VirtuSwap]` : Issue acknowledged. Changes have been reflected in the commit hash [44e3dd2946cf74bc8082b5da897ff21f14cdc0b5](#).

## RVT-05 | FINDING DETAILS

### Finding Title

Typo In Comment

Category	Severity	Location	Status
Coding Style	● Informational	contracts/vRouter.sol: 69~70	● Resolved

### Description

In the validation that `requiredBackAmount <= data.tokenInMax`, the corresponding comment reads as `VSWAP:REQUIRED_AMOUNT_EXCEEDS`. This comment should read `VSWAP:REQUIRED_AMOUNT_EXCEEDS`.

### Recommendation

We recommend making the change to the comment for better end-user readability.

### Alleviation

[VirtuSwap]: Issue acknowledged. Changes have been reflected in the commit hash [c46db0d4dfdd7a65755df5ad57667285d514da6f](#).

## SEC-01 | FINDING DETAILS

### Finding Title

Function `_transfer()` Should Be Internal

Category	Severity	Location	Status
Logical Issue	<span>●</span> Critical	contracts/vSwapERC20.sol: 248~273	<span>●</span> Resolved

### Description

The `_transfer` function is used to move tokens from a specified sender address to another specified receiver address without any restriction. It should only be used as an internal utility function by other public functions which do security checks. The consequence is anyone can call the `_transfer()` function to move tokens between arbitrary addresses.

### Recommendation

We recommend the client update the function visibility from `public` to `internal`.

### Alleviation

`[VirtuSwap]`: Issue acknowledged. Changes have been reflected in the commit hash [4966862d2f8b2d3e055ce8f2ec9b57abc7f4b3f0](#).

## OPTIMIZATIONS | VIRTUSWAP

ID	Title	Category	Severity	Status
<u>CON-04</u>	Improper Usage Of <code>public</code> And <code>external</code> Type	Gas Optimization	Optimization	● Resolved
<u>CON-05</u>	Memory Used Over Calldata	Gas Optimization	Optimization	● Resolved
<u>RVH-01</u>	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	● Resolved
<u>SEC-02</u>	Variables That Could Be Declared As <code>constant</code>	Gas Optimization	Optimization	● Resolved

## CON-04 | FINDING DETAILS

### Finding Title

Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/base/multicall.sol: 12; contracts/libraries/poolAddress.sol: 34; contracts/vPair.sol: 299, 309, 351, 375, 388; contracts/vRouter.sol: 242, 257~267, 267; contracts/vSwapERC20.sol: 108, 142, 169, 192, 216	● Resolved

### Description

`public` functions that are never called internally by the contract could be declared as `external`.

`external` functions which are called internally within the contract should have `public` visibility instead of using the `this.f()` pattern, as this requires a real CALL to be executed, which is more expensive.

### Recommendation

We recommend the client use the `external` attribute for public functions that are never called within the contract, and `public` attribute for external functions that are called internally (and externally) within the contract, making a direct call to the function by the name `f()` instead of the pattern `this.f()`.

### Alleviation

[VirtuSwap]: Issue acknowledged. Changes have been reflected in the commit hash [bd23f7ad671a5a63f5a38a648657aba10755b970](#).

## CON-05 | FINDING DETAILS

### Finding Title

Memory Used Over Calldata

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/vPair.sol: 105~106; contracts/vRouter.sol: 54~55	● Resolved

### Description

In the `vPair` contract, the function `swapNative()` has an input `data` of type `bytes` which is originally stored in `calldata` within the the contract interface. This function is overridden to use `memory` instead within the contract itself.

In the function `vFlashSwapCallback()` in `vRouter`, memory is again used over calldata, even though the `swapNativeToReserve()` and `swapReserveToNative()` functions store `data` in calldata.

### Recommendation

We recommend keeping the use of `calldata` rather than `memory` in order to save gas and retain consistency.

### Alleviation

`[VirtuSwap]`: Issue acknowledged. Changes have been reflected in the commit hash [17a6a5a0df34fc35b27de5eeab5b5403e9473b14](#).

## RVH-01 | FINDING DETAILS

### Finding Title

Variables That Could Be Declared As Immutable

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/exchangeReserves.sol: 10	● 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. An 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

`[VirtuSwap]`: Issue acknowledged. Changes have been reflected in the commit hash [0e6138f3b496b5a94d41159c2e7d7be25533ff74](#).

## SEC-02 | FINDING DETAILS

### Finding Title

Variables That Could Be Declared As `constant`

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/vSwapERC20.sol: 45, 46	● Resolved

### Description

The linked variables could be declared as `constant` since these state variables are never modified.

### Recommendation

We recommend declaring these variables as `constant` .

### Alleviation

[VirtuSwap] : Issue acknowledged. Changes have been reflected in the commit hash [bc3e7d6b39f5d6eab179a917cd2e959016a44e10](#).

## APPENDIX | VIRTUSWAP

### 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.
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.
Control Flow	Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.
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.
Compiler Error	Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

### 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.

