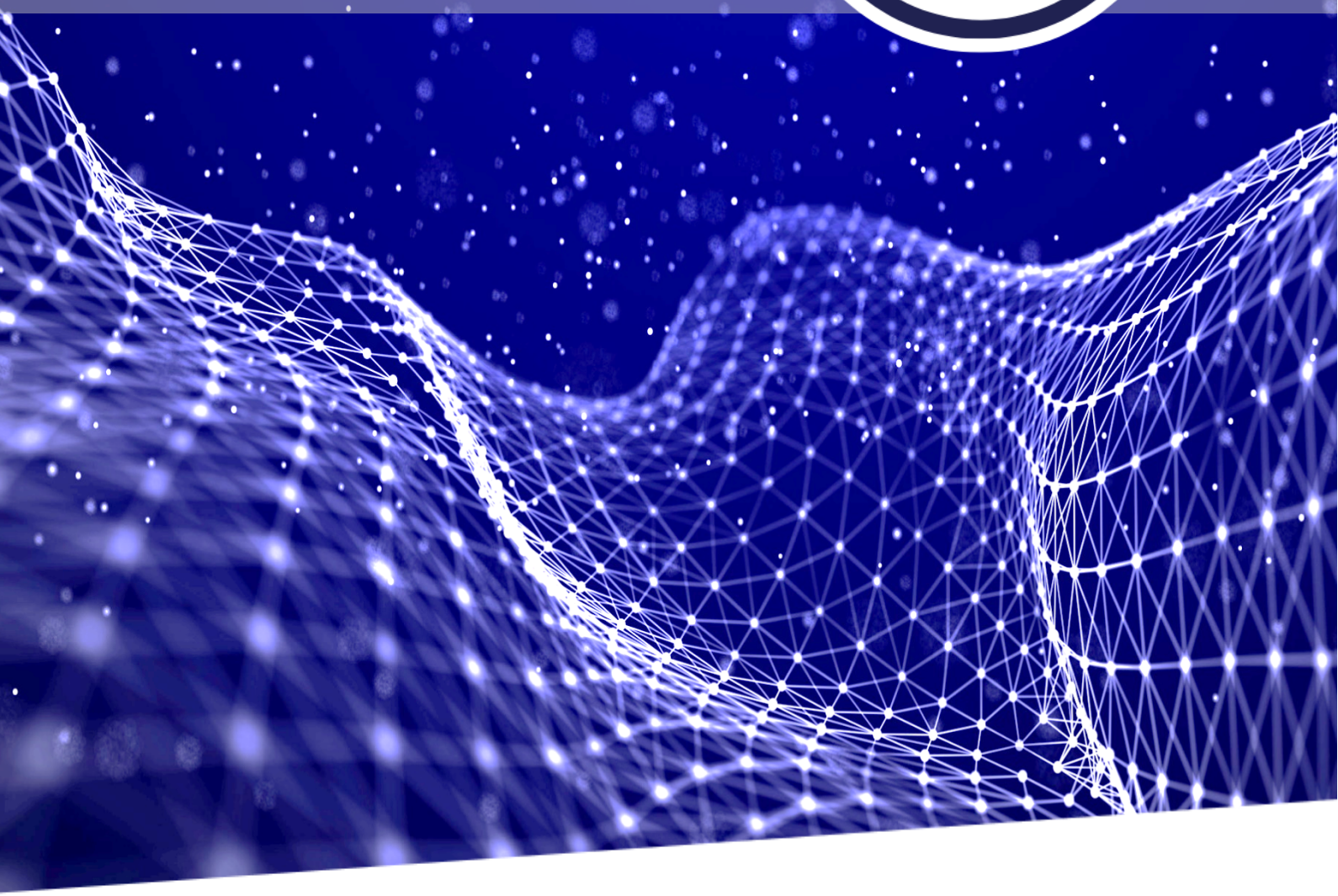
Warden Finance

Aegis and Aegis L2

Smart Contract Audit Report



Date Issued: *6 Oct 2021*

Version: *Final v1.0*

ValiX Consulting

Public

Table of Contents

Executive Summary

Overview

Valix conducted a smart contract audit to evaluate potential security issues of the **Aegis and Aegis L2 features of the WardenSwap version 1.5**. This audit report was published on *October 6, 2021*. The audit scope is limited to the **Aegis and Aegis L2 features.** Our security best practices strongly recommend that the **Warden Finance team** conduct a full security audit for both on-chain and off-chain components of its infrastructure and their interaction. A comprehensive examination has been performed during the audit process utilizing Valix's Formal Verification, Static Analysis, and Manual Review techniques.

About Aegis and Aegis L2

Aegis is the release version of Warden Finance that implements trading strategies learning algorithm into a smart contract (on chain's machine learning). Aegis uses the learning outcome to optimize future Best Rate swaps.

Aegis L2 is the release version that focuses on gas optimization on the Ethereum Layer 2 Scaling Solution like Optimistic Rollup (e.g., Optimism and Arbitrum). Aegis L2 uses custom data serialization and compression algorithms to reduce calldata bytes and storage slot usage.

Scope of Work

The security audit conducted does not replace the full security audit of the overall Warden Finance protocol. The scope is limited to the **Aegis and Aegis L2 features** and their related smart contracts.

The security audit covered the components at this specific state:

Item	Description
Components	▪ *WardenSwap1_5_Aegis smart contract* ▪ *WardenSwap1_5_Aegis_L2 smart contract* ▪ *WardenCosmoCore smart contract* ▪ *WardenDataDeserialize smart contract* ▪ *BytesLib smart contract library* ▪ *WardenDataSerialize smart contract* ▪ *Imported associated smart contracts and libraries*
GitHub Repository	▪ *https://github.com/Wardenswap/warden-swap*
Commit	▪ *b03ee7c3190415e62223ede8ee4ad21f4cca6691*

Reassessment Commit	▪ *5a8fefcfc466e8b8408846694114906f7c625651*
Audited Files	▪ *WardenSwap1_5_L2.sol* ▪ *interface/IWardenCosmicBrainForL2.sol* ▪ *interface/IWardenCosmoCore0_8.sol* ▪ *interface/IWardenPostTrade.sol* ▪ *libraries/IWETH.sol* ▪ *libraries/IWardenTradingRoute0_8.sol* ▪ *libraries/WardenCosmoCore.sol* ▪ *libraries/WardenDataDeserialize.sol* ▪ *library/arbitrum/IArbAddressTable.sol* ▪ *library/byte/BytesLib.sol* ▪ *tools/WardenDataSerialize.sol*
Excluded Files/Contracts	▪ *WardenSwap1_5_L2_dryrun.sol* ▪ *WardenCosmicBrain smart contract* ▪ *WardenPostTrade smart contract* ▪ *WETH smart contract* ▪ *WardenTradingRoute smart contract* ▪ *ArbAddressTable smart contract*

Remark: Our security best practices strongly recommend that the Warden Finance team conduct a full security audit for both on-chain and off-chain components of its infrastructure and the interaction between them.

Auditors

Phuwanai Thummavet

Sumedt Jitpukdebodin

Keerati Torach

Boonpoj Thongakaraniroj

Disclaimer

Our smart contract audit was conducted over a limited period and was performed on the smart contract at a single point in time. As such, the scope was limited to current known risks during the work period. The review does not indicate that the smart contract and blockchain software has no vulnerability exposure.

We reviewed the security of the smart contracts with our best effort, and we do not guarantee a hundred percent coverage of the underlying risk existing in the ecosystem. The audit was scoped only in the provided code repository. The on-chain code is not in the scope of auditing.

This audit report does not provide any warranty or guarantee, nor should it be considered an “approval” or “endorsement” of any particular project. This audit report should also not be used as investment advice nor provide any legal compliance.

Audit Result Summary

From the audit results and the remediation and response from the developer, Valix trusts that the **Aegis and Aegis L2 features** have sufficient security protections to be safe for use.

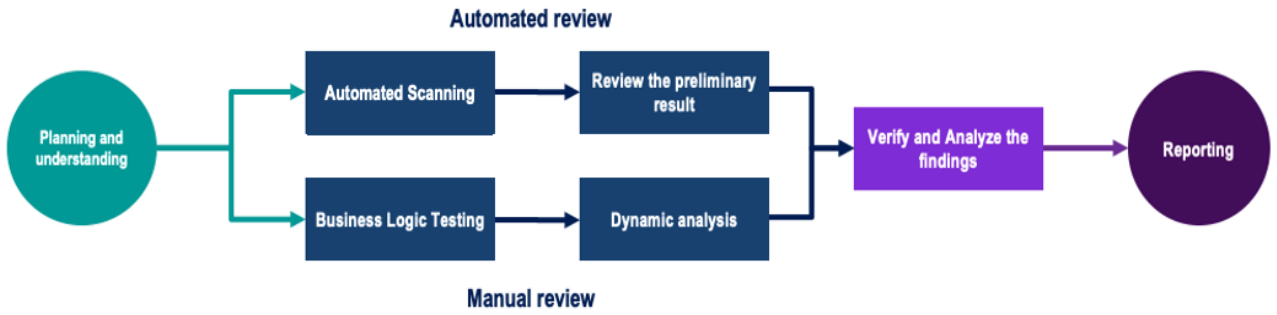
Initially, Valix was able to identify **19 issues** that were categorized from the "Critical" to "Informational" risk level in the given timeframe of the assessment. On the reassessment, 13 out of 19 issues were fixed. For the acknowledged issues, the Warden Finance team acknowledged each issue but decided to remain the original code. Below is the breakdown of the vulnerabilities found and their associated risk rating for each assessment conducted.

Target	Assessment Result					Reassessment Result				
	C	H	M	L	I	C	H	M	L	I
Aegis and Aegis L2	-	-	-	9	10	-	-	-	3	3

Note: *Risk Rating* **C** *Critical,* **H** *High,* **M** *Medium,* **L** *Low,* **I** *Informational*

Methodology

The smart contract security audit methodology is based on Smart Contract Weakness Classification and Test Cases (SWC Registry), CWE, well-known best practices, and smart contract hacking case studies. Manual and automated review approaches can be mixed and matched, including business logic analysis in terms of the malicious doer's perspective. Using automated scanning tools to navigate or find offending software patterns in the codebase along with a purely manual or semi-automated approach, where the analyst primarily relies on one's knowledge, is performed to eliminate the false-positive results.



Planning and Understanding

- Determine the scope of testing and understanding the application's purposes and workflows.
- Identify key risk areas, including technical and business risks.
- Determine which sections to review within the resource constraints and review method – automated, manual or mixed.

Automated Review

- Adjust automated source code review tools to inspect the code for known unsafe coding patterns.
- Verify the tool's output to eliminate false-positive results, and adjust and re-run the code review tool if necessary.

Manual Review

- Analyzing the business logic flaws requires thinking in unconventional methods.
- Identify unsafe coding behavior via static code analysis.

Reporting

- Analyze the root cause of the flaws.
- Recommend improvements for secure source code.

Audit Items

We perform the audit according to the following categories and test names.

Category	ID	Test Name
Security Issue	SEC01	*Authorization Through tx.origin*
	SEC02	*Business Logic Flaw*
	SEC03	*Delegatecall to Untrusted Callee*
	SEC04	*DoS With Block Gas Limit*
	SEC05	*DoS with Failed Call*
	SEC06	*Function Default Visibility*
	SEC07	*Hash Collisions With Multiple Variable Length Arguments*
	SEC08	*Incorrect Constructor Name*
	SEC09	*Improper Access Control or Authorization*
	SEC10	*Improper Emergency Response Mechanism*
	SEC11	*Insufficient Validation of Address Length*
	SEC12	*Integer Overflow and Underflow*
	SEC13	*Outdated Compiler Version*
	SEC14	*Outdated Library Version*
	SEC15	*Private Data On-Chain*
	SEC16	*Reentrancy*
	SEC17	*Transaction Order Dependence*
	SEC18	*Unchecked Call Return Value*
	SEC19	*Unexpected Token Balance*
	SEC20	*Unprotected Assignment of Ownership*
	SEC21	*Unprotected SELFDESTRUCT Instruction*
	SEC22	*Unprotected Token Withdrawal*
	SEC23	*Unsafe Type Inference*
	SEC24	*Use of Deprecated Solidity Functions*
	SEC25	*Use of Untrusted Code or Libraries*
	SEC26	*Weak Sources of Randomness from Chain Attributes*
	SEC27	*Write to Arbitrary Storage Location*

Category	ID	Test Name
Functional Issue	FNC01	*Arithmetic Precision*
	FNC02	*Permanently Locked Fund*
	FNC03	*Redundant Fallback Function*
	FNC04	*Timestamp Dependence*
Operational Issue	OPT01	*Code With No Effects*
	OPT02	*Message Call with Hardcoded Gas Amount*
	OPT03	*The Implementation Contract Flow or Value and the Document is Mismatched*
	OPT04	*The Usage of Excessive Byte Array*
	OPT05	*Unenforced Timelock on An Upgradeable Proxy Contract*
Developmental Issue	DEV01	*Assert Violation*
	DEV02	*Other Compilation Warnings*
	DEV03	*Presence of Unused Variables*
	DEV04	*Shadowing State Variables*
	DEV05	*State Variable Default Visibility*
	DEV06	*Typographical Error*
	DEV07	*Uninitialized Storage Pointer*
	DEV08	*Violation of Solidity Coding Convention*
	DEV09	*Violation of Token (ERC20) Standard API*

Risk Rating

To prioritize the vulnerabilities, we have adopted the scheme of five distinct levels of risk: **Critical**, **High**, **Medium**, **Low**, and **Informational**, based on OWASP Risk Rating Methodology. The risk level definitions are presented in the table.

Risk Level	Definition
Critical	The code implementation does not match the specification, and it could disrupt the platform.
High	The code implementation does not match the specification, or it could result in the loss of funds for contract owners or users.
Medium	The code implementation does not match the specification under certain conditions, or it could affect the security standard by losing access control.
Low	The code implementation does not follow best practices or use suboptimal design patterns, which may lead to security vulnerabilities further down the line.
Informational	Findings in this category are informational and may be further improved by following best practices and guidelines.

The **risk value** of each issue was calculated from the product of the **impact** and **likelihood values**, as illustrated in a two-dimensional matrix below.

- **Likelihood** represents how likely a particular vulnerability is exposed and exploited in the wild.
- **Impact** measures the technical loss and business damage of a successful attack.
- **Risk** demonstrates the overall criticality of the risk.

Impact \ Likelihood	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Informational

The shading of the matrix visualizes the different risk levels. Based on the acceptance criteria, the risk levels "Critical" and "High" are unacceptable. Any issue obtaining the above levels must be resolved to lower the risk to an acceptable level.

Findings

Review Findings Summary

The table below shows the summary of our assessments.

No.	Issue	Risk	Status	Functionality is in use
1	Potential Stealing of Leftover Ether and WETH	Low	Fixed	In use
2	Potential Lock of Ether	Low	Fixed	In use
3	Recommended Updating The `PostTrade` Contract With Extra Care	Low	Acknowledged	In use
4	Unchecking Duplication of Trading Route On The `addTradingRoute` Function	Low	Acknowledged	In use
5	Unchecking Duplication of Trading Route On The `updateTradingRoute` Function	Low	Acknowledged	In use
6	The Split Volume May Be Inconsistent With The Actual Amount	Low	Fixed	In use
7	Lack of Some Edge Case In Input Validation	Low	Fixed	In use
8	The Compiler May Be Susceptible To The Publicly Disclosed Bugs	Low	Fixed	In use
9	The Compiler Is Not Locked To A Specific Version	Low	Fixed	In use
10	Transparency Improvement For The `collectRemainingToken` Function	Informational	Fixed	In use
11	Transparency Improvement For The `collectRemainingEther` Function	Informational	Fixed	In use
12	Gas Optimization and Readability Improvement On The `_split2` Function	Informational	Fixed	In use
13	Inconsistent Comments/Error Messages With The Code	Informational	Fixed	In use
14	Recommended Explicit Trading Route Validation Checks	Informational	Acknowledged	In use
15	Unchecked Call Return Value	Informational	Acknowledged	In use
16	Gas Optimization On Redundant Code	Informational	Acknowledged	In use
17	Duplicate Function Implementation	Informational	Fixed	In use

No.	Issue	Risk	Status	Functionality is in use
18	Generic Typographic Error	Informational	Fixed	In use
19	Misleading Function Name	Informational	Fixed	In use

The statuses of the issues are defined as follows:

Fixed: The issue has been completely resolved and has no further complications.

Partially Fixed: The issue has been partially resolved.

Acknowledged: The issue's risk has been reported and acknowledged.

Detailed Result

This section provides all issues that we found in detail.

No. 1	Potential Stealing of Leftover Ether and WETH		
Risk	Low	Likelihood	Low
		Impact	Medium
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*_tradeStrategiesWithSafeGuard(IERC20, uint256, IERC20, uint256, uint256[], IERC20[], address, uint256) L: 196 - 285* *_splitTradesWithSafeGuard(uint256[], uint256[], IERC20, uint256, IERC20) L: 409 - 458* *tradeEthToWeth(address) L: 524 - 536* *tradeWethToEth(uint256, address) L: 544 - 558*		

Detailed Issue

The state variable `weth` (line no. 20) points to the external `WETH` contract responsible for wrapping the `Ether` (native ETH) to the ERC20 token, `WETH`, and vice versa.

In exchange, when we deposit the `Ether` to the `WETH` contract, the deposited `Ether` will be locked, and the same amount in `WETH` will be minted and returned to us. On the other hand, we can withdraw the locked `Ether` by transferring the same amount in `WETH` to the `WETH` contract to burn.

However, if the `weth` variable is initialized with the rogue (forged) `WETH` contract by mistake, the rogue contract could secretly steal the `Ethers` or `WETHs` leftover in the WardenSwap1_5_Aegis contract.

Note that the leftover `Ether` and `WETH` mean the tokens are mistakenly transferred by a user and locked in the WardenSwap1_5_Aegis contract.

Consider the following attack scenario for better understanding.

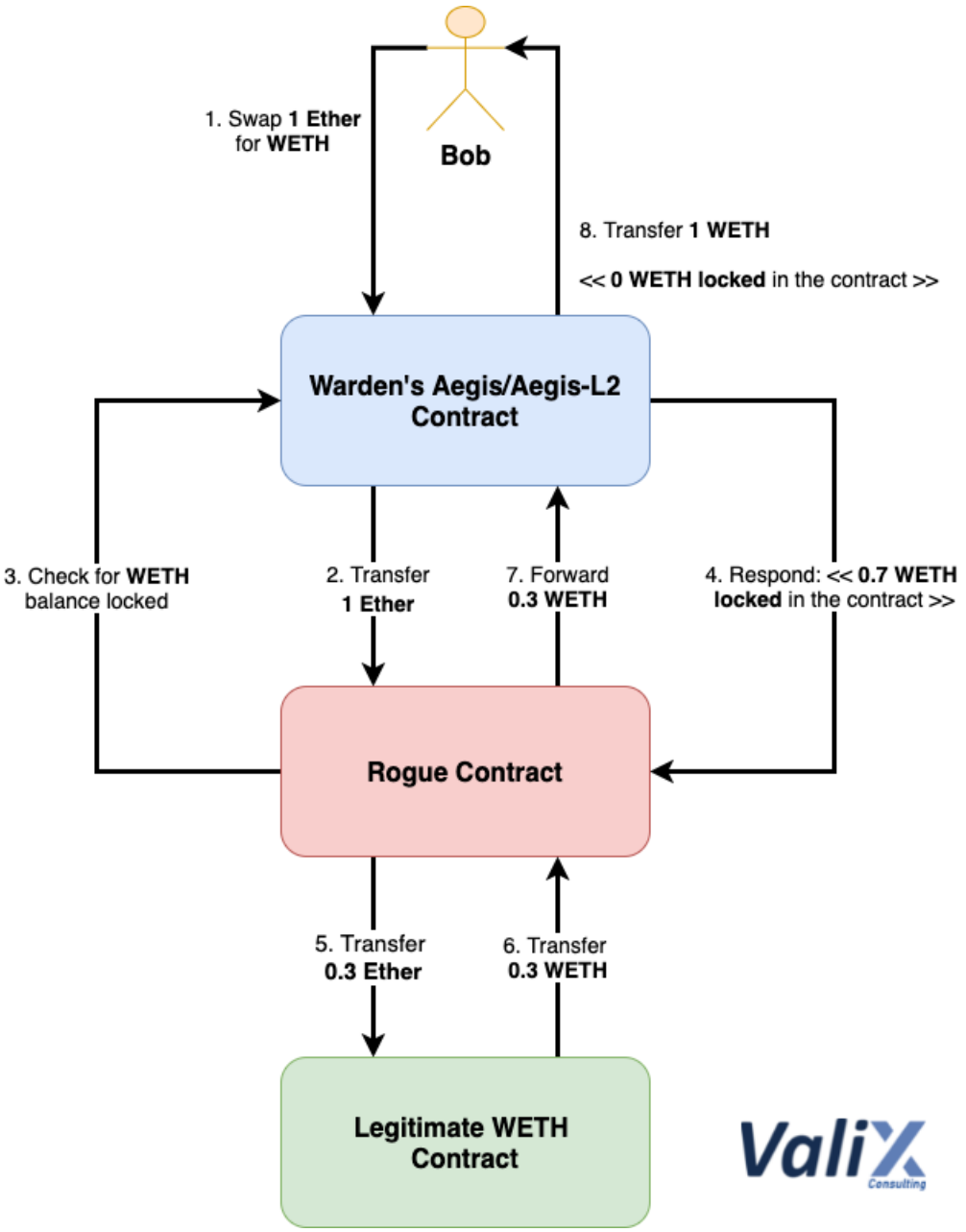


Figure 1.1 The **Rogue** contract can steal the leftover *Ether* and *WETH* from the **Aegis** contract

1. **Bob** initiates a swap transaction of *1 Ether* for *WETH* to **Aegis** contract
2. **Aegis** contract transfers *1 Ether* to **Rogue** contract
3. **Rogue** contract checks for the *WETH balance locked* in **Aegis** contract
4. **Aegis** contract has *0.7 WETH locked in*, for instance
5. **Rogue** contract transfers *0.3 Ether* (*instead of 1 Ether*) to **WETH** contract
6. **WETH** contract locks *0.3 Ether*, mints the same amount in *WETH*, and transfers *the minted WETH* to **Rogue** contract
7. **Rogue** contract transfers only *0.3 WETH* (*instead of 1 WETH*) to **Aegis** contract

8. **Aegis** contract transfers *1 WETH* to **Bob** (without checking its *WETH* balance)

The **Rogue** contract can privately steal all *Ether* and *WETH tokens* locked in the **Aegis** contract by swapping the tokens using the attack scenario above. This attack scenario can be triggered by an attacker itself or just a regular user.

The functions affected with this issue include *_tradeStrategiesWithSafeGuard* (line no's. 196 - 285), *_splitTradesWithSafeGuard* (line no's. 409 - 458), *tradeEthToWeth* (line no's. 524 - 536), and *tradeWethToEth* (line no's. 544 - 558). The following code snippets show the affected functions.

WardenSwap1_5_L2.sol

```
196 function _tradeStrategiesWithSafeGuard(
197    IERC20      _src,
198    uint256     _srcAmount,
199    IERC20      _dest,
200    uint256     _minDestAmount,
201    uint256[]   memory _subRoutes,
202    IERC20[]    memory _correspondentTokens,
203    address     _receiver,
204    uint256     _learnedId
205 )
206    private
207    returns(uint256 _destAmount)
208 {
209    require(_subRoutes.length - 1 == _correspondentTokens.length, "WardenSwap:
    routes and tokens length mismatched");
210    {
211        IERC20 adjustedSrc;
212        IERC20 adjustedDest = ETHER_ERC20 == _dest ? IERC20(address(weth)) :
    _dest;
213        address fromAddress;
214
215        // Wrap ETH
216        if (ETHER_ERC20 == _src) {
217            require(msg.value == _srcAmount, "WardenSwap: Ether source amount
    mismatched");
218            weth.deposit{value: _srcAmount}();
219
220            adjustedSrc = IERC20(address(weth));
221            fromAddress = address(this);
222        } else {
223            adjustedSrc = _src;
224            fromAddress = msg.sender;
225        }
226
227        // Record src/dest asset for later consistency check.
228        uint256 srcAmountBefore = adjustedSrc.balanceOf(fromAddress);
229        uint256 destAmountBefore = adjustedDest.balanceOf(address(this));
230
231        _destAmount = _tradeStrategies(
```

```
232             adjustedSrc,
233             _srcAmount,
234             adjustedDest,
235             _subRoutes,
236             _correspondentTokens,
237             fromAddress
238         );
239 
240         // Sanity check
241         // Recheck if src/dest amount correct
242         require(adjustedSrc.balanceOf(fromAddress) == srcAmountBefore -
    _srcAmount, "WardenSwap: source amount mismatched after trade");
243         require(adjustedDest.balanceOf(address(this)) == destAmountBefore +
    _destAmount, "WardenSwap: destination amount mismatched after trade");
244     }
245 
246 
247     // Unwrap ETH
248     if (ETHER_ERC20 == _dest) {
249         weth.withdraw(_destAmount);
250     }
251 
252     // Collect fee
253     _destAmount = _postTradeAndCollectFee(
254         _src,
255         _dest,
256         _srcAmount,
257         _destAmount,
258         msg.sender,
259         _receiver,
260         false
261     );
262 
263     // Throw exception if destination amount doesn't meet user requirement.
264     require(_destAmount >= _minDestAmount, "WardenSwap: destination amount is too
    low.");
265     if (ETHER_ERC20 == _dest) {
266         (bool success, ) = _receiver.call{value: _destAmount}(""); // Send back
    ether to sender
267         require(success, "WardenSwap: Transfer ether back to caller failed.");
268     } else { // Send back token to sender
269         _dest.safeTransfer(_receiver, _destAmount);
270     }
271 
272     uint256 learnedId = _learnedId;
273     if (0 == _learnedId) {
274         learnedId = cosmicBrain.train(_subRoutes, _correspondentTokens);
275     }
276     cosmicBrain.trainTradingPair(
277         _src,
278         _dest,
```

```
279         _srcAmount,
280         _destAmount,
281         learnedId
282     );
283 
284     emit Trade(address(_src), _srcAmount, address(_dest), _destAmount,
    msg.sender, _receiver, 0 != _learnedId, false);
285 }
```

Listing 1.1 The affected *_tradeStrategiesWithSafeGuard* function

WardenSwap1_5_L2.sol

```
409 function _splitTradesWithSafeGuard(
410     uint256[] memory  _learnedIds,
411     uint256[] memory  _volumns,
412     IERC20              _src,
413     uint256             _totalSrcAmount,
414     IERC20              _dest
415 )
416     private
417     returns(uint256 _destAmount)
418 {
419     IERC20 adjustedSrc;
420     IERC20 adjustedDest = ETHER_ERC20 == _dest ? IERC20(address(weth)) : _dest;
421     address fromAddress;
422 
423     // Wrap ETH
424     if (ETHER_ERC20 == _src) {
425         require(msg.value == _totalSrcAmount, "WardenSwap: Ether source amount
    mismatched");
426         weth.deposit{value: _totalSrcAmount}();
427 
428         adjustedSrc = IERC20(address(weth));
429         fromAddress = address(this);
430     } else {
431         adjustedSrc = _src;
432         fromAddress = msg.sender;
433     }
434 
435     // Record src/dest asset for later consistency check.
436     uint256 srcAmountBefore = adjustedSrc.balanceOf(fromAddress);
437     uint256 destAmountBefore = adjustedDest.balanceOf(address(this));
438 
439     _destAmount = _split2(
440         _learnedIds,
441         _volumns,
442         adjustedSrc,
443         _totalSrcAmount,
```

```
444         adjustedDest,
445         fromAddress
446     );
447
448     // Sanity check
449     // Recheck if src/dest amount correct
450     require(adjustedSrc.balanceOf(fromAddress) == srcAmountBefore -
    _totalSrcAmount, "WardenSwap: source amount mismatched after trade");
451     require(adjustedDest.balanceOf(address(this)) == destAmountBefore +
    _destAmount, "WardenSwap: destination amount mismatched after trade");
452
453
454     // Unwrap ETH
455     if (ETHER_ERC20 == _dest) {
456         weth.withdraw(_destAmount);
457     }
458 }
```

Listing 1.2 The affected *_splitTradesWithSafeGuard* function

WardenSwap1_5_L2.sol

```
524 function tradeEthToWeth(
525     address     _receiver
526 )
527     external
528     payable
529     nonReentrant
530     returns(uint256 _destAmount)
531 {
532     weth.deposit{value: msg.value}();
533     IERC20(address(weth)).safeTransfer(_receiver, msg.value);
534     _destAmount = msg.value;
535     emit Trade(address(ETHER_ERC20), msg.value, address(weth), _destAmount,
    msg.sender, _receiver, false, false);
536 }
```

Listing 1.3 The affected *tradeEthToWeth* function

WardenSwap1_5_L2.sol

```
544 function tradeWethToEth(
545     uint256     _srcAmount,
546     address     _receiver
547 )
548     external
549     nonReentrant
550     returns(uint256 _destAmount)
551 {
552     IERC20(address(weth)).safeTransferFrom(msg.sender, address(this),
    _srcAmount);
553     weth.withdraw(_srcAmount);
554     (bool success, ) = _receiver.call{value: _srcAmount}(""); // Send back ether
    to sender
555     require(success, "WardenSwap: Transfer ether back to caller failed.");
556     _destAmount = _srcAmount;
557     emit Trade(address(weth), _srcAmount, address(ETHER_ERC20), _destAmount,
    msg.sender, _receiver, false, false);
558 }
```

Listing 1.4 The affected *tradeWethToEth* function

The root cause of this issue is because the affected functions did not verify the balance of *Ether* or *WETH* after the *Ether* wrapping or unwrapping process.

Recommendations

We recommend updating the affected functions to verify that the *Ether* or *WETH* is still in the balance after the *Ether* wrapping or unwrapping process. Consider the following code snippets for the improved functions.

WardenSwap1_5_L2.sol

```
196 function _tradeStrategiesWithSafeGuard(
197     IERC20      _src,
198     uint256     _srcAmount,
199     IERC20      _dest,
200     uint256     _minDestAmount,
201     uint256[]   memory _subRoutes,
202     IERC20[]    memory _correspondentTokens,
203     address     _receiver,
204     uint256     _learnedId
205 )
206     private
207     returns(uint256 _destAmount)
208 {
209     require(_subRoutes.length - 1 == _correspondentTokens.length, "WardenSwap:
    routes and tokens length mismatched");
```

```
210     {
211         IERC20 adjustedSrc;
212         IERC20 adjustedDest = ETHER_ERC20 == _dest ? IERC20(address(weth)) :
    _dest;
213         address fromAddress;
214 
215         // Wrap ETH
216         if (ETHER_ERC20 == _src) {
217             require(msg.value == _srcAmount, "WardenSwap: Ether source amount
    mismatched");
218 
219             uint256 wethAmountBefore =
    IERC20(address(weth)).balanceOf(address(this));
220             weth.deposit{value: _srcAmount}();
221             uint256 wethAmountAfter =
    IERC20(address(weth)).balanceOf(address(this));
222 
223             // Verify the balance of WETH after wrapping
224             require(wethAmountAfter == wethAmountBefore + _srcAmount,
    "WardenSwap: received unexpected WETH amount");
225 
226             adjustedSrc = IERC20(address(weth));
227             fromAddress = address(this);
228         } else {
229             adjustedSrc = _src;
230             fromAddress = msg.sender;
231         }
232 
233         // Record src/dest asset for later consistency check.
234         uint256 srcAmountBefore = adjustedSrc.balanceOf(fromAddress);
235         uint256 destAmountBefore = adjustedDest.balanceOf(address(this));
236 
237         _destAmount = _tradeStrategies(
238             adjustedSrc,
239             _srcAmount,
240             adjustedDest,
241             _subRoutes,
242             _correspondentTokens,
243             fromAddress
244         );
245 
246         // Sanity check
247         // Recheck if src/dest amount correct
248         require(adjustedSrc.balanceOf(fromAddress) == srcAmountBefore -
    _srcAmount, "WardenSwap: source amount mismatched after trade");
249         require(adjustedDest.balanceOf(address(this)) == destAmountBefore +
    _destAmount, "WardenSwap: destination amount mismatched after trade");
250     }
251 
252 
253     // Unwrap ETH
```

```
254     if (ETHER_ERC20 == _dest) {
255         uint256 etherAmountBefore = address(this).balance;
256         weth.withdraw(_destAmount);
257         uint256 etherAmountAfter = address(this).balance;
258 
259         // Verify the balance of Ether after unwrapping
260         require(etherAmountAfter == etherAmountBefore + _destAmount, "WardenSwap:
    received unexpected Ether amount");
261     }
262 
263     // Collect fee
264     _destAmount = _postTradeAndCollectFee(
265         _src,
266         _dest,
267         _srcAmount,
268         _destAmount,
269         msg.sender,
270         _receiver,
271         false
272     );
273 
274     // Throw exception if destination amount doesn't meet user requirement.
275     require(_destAmount >= _minDestAmount, "WardenSwap: destination amount is too
    low.");
276     if (ETHER_ERC20 == _dest) {
277         (bool success, ) = _receiver.call{value: _destAmount}(""); // Send back
    ether to sender
278         require(success, "WardenSwap: Transfer ether back to caller failed.");
279     } else { // Send back token to sender
280         _dest.safeTransfer(_receiver, _destAmount);
281     }
282 
283     uint256 learnedId = _learnedId;
284     if (0 == _learnedId) {
285         learnedId = cosmicBrain.train(_subRoutes, _correspondentTokens);
286     }
287     cosmicBrain.trainTradingPair(
288         _src,
289         _dest,
290         _srcAmount,
291         _destAmount,
292         learnedId
293     );
294 
295     emit Trade(address(_src), _srcAmount, address(_dest), _destAmount,
    msg.sender, _receiver, 0 != _learnedId, false);
296 }
```

Listing 1.5 The improved *_tradeStrategiesWithSafeGuard* function

WardenSwap1_5_L2.sol

```
409 function _splitTradesWithSafeGuard(
410     uint256[] memory  _learnedIds,
411     uint256[] memory  _volumns,
412     IERC20              _src,
413     uint256             _totalSrcAmount,
414     IERC20              _dest
415 )
416     private
417     returns(uint256 _destAmount)
418 {
419     IERC20 adjustedSrc;
420     IERC20 adjustedDest = ETHER_ERC20 == _dest ? IERC20(address(weth)) : _dest;
421     address fromAddress;
422
423     // Wrap ETH
424     if (ETHER_ERC20 == _src) {
425         require(msg.value == _totalSrcAmount, "WardenSwap: Ether source amount
    mismatched");
426
427         uint256 wethAmountBefore =
    IERC20(address(weth)).balanceOf(address(this));
428         weth.deposit{value: _totalSrcAmount}();
429         uint256 wethAmountAfter = IERC20(address(weth)).balanceOf(address(this));
430
431         // Verify the balance of WETH after wrapping
432         require(wethAmountAfter == wethAmountBefore + _totalSrcAmount,
    "WardenSwap: received unexpected WETH amount");
433
434         adjustedSrc = IERC20(address(weth));
435         fromAddress = address(this);
436     } else {
437         adjustedSrc = _src;
438         fromAddress = msg.sender;
439     }
440
441     // Record src/dest asset for later consistency check.
442     uint256 srcAmountBefore = adjustedSrc.balanceOf(fromAddress);
443     uint256 destAmountBefore = adjustedDest.balanceOf(address(this));
444
445     _destAmount = _split2(
446         _learnedIds,
447         _volumns,
448         adjustedSrc,
449         _totalSrcAmount,
450         adjustedDest,
451         fromAddress
452     );
453
454     // Sanity check
```

```
455     // Recheck if src/dest amount correct
456     require(adjustedSrc.balanceOf(fromAddress) == srcAmountBefore -
    _totalSrcAmount, "WardenSwap: source amount mismatched after trade");
457     require(adjustedDest.balanceOf(address(this)) == destAmountBefore +
    _destAmount, "WardenSwap: destination amount mismatched after trade");
458
459
460     // Unwrap ETH
461     if (ETHER_ERC20 == _dest) {
462         uint256 etherAmountBefore = address(this).balance;
463         weth.withdraw(_destAmount);
464         uint256 etherAmountAfter = address(this).balance;
465
466         // Verify the balance of Ether after unwrapping
467         require(etherAmountAfter == etherAmountBefore + _destAmount, "WardenSwap:
    received unexpected Ether amount");
468     }
469 }
```

Listing 1.6 The improved *_splitTradesWithSafeGuard* function

WardenSwap1_5_L2.sol

```
524 function tradeEthToWeth(
525     address     _receiver
526 )
527     external
528     payable
529     nonReentrant
530     returns(uint256 _destAmount)
531 {
532     uint256 wethAmountBefore = IERC20(address(weth)).balanceOf(address(this));
533     weth.deposit{value: msg.value}();
534     uint256 wethAmountAfter = IERC20(address(weth)).balanceOf(address(this));
535
536     // Verify the balance of WETH after wrapping
537     require(wethAmountAfter == wethAmountBefore + msg.value, "WardenSwap:
    received unexpected WETH amount");
538
539     IERC20(address(weth)).safeTransfer(_receiver, msg.value);
540     _destAmount = msg.value;
541     emit Trade(address(ETHER_ERC20), msg.value, address(weth), _destAmount,
    msg.sender, _receiver, false, false);
542 }
```

Listing 1.7 The improved *tradeEthToWeth* function

WardenSwap1_5_L2.sol

```
544 function tradeWethToEth(
545     uint256     _srcAmount,
546     address     _receiver
547 )
548     external
549     nonReentrant
550     returns(uint256 _destAmount)
551 {
552     IERC20(address(weth)).safeTransferFrom(msg.sender, address(this),
    _srcAmount);
553
554     uint256 etherAmountBefore = address(this).balance;
555     weth.withdraw(_srcAmount);
556     uint256 etherAmountAfter = address(this).balance;
557
558     // Verify the balance of Ether after unwrapping
559     require(etherAmountAfter == etherAmountBefore + _srcAmount, "WardenSwap:
    received unexpected Ether amount");
560
561     (bool success, ) = _receiver.call{value: _srcAmount}(""); // Send back ether
    to sender
562     require(success, "WardenSwap: Transfer ether back to caller failed.");
563     _destAmount = _srcAmount;
564     emit Trade(address(weth), _srcAmount, address(ETHER_ERC20), _destAmount,
    msg.sender, _receiver, false, false);
565 }
```

Listing 1.8 The improved *tradeWethToEth* function

Reassessment

The developer opted to remediate this issue by changing the visibility of the state variable *weth* from *private* to *public* (line no. 20 in Listing 1.9) instead of modifying the affected functions to trade for minimal gas use.

The public visibility of the *weth* enables a user to inspect the legitimacy of the *WETH* contract. Since the *weth* variable can be assigned only once in the constructor (line no. 78 in Listing 1.10), if the *weth* is initialized correctly during the smart contract deployment, the *weth* cannot be updated later.

WardenSwap1_5_L2.sol

```
...(SNIP)...
13 contract WardenSwap1_5_Aegis is Ownable, ReentrancyGuard {
14     using SafeERC20 for IERC20;
15
16     IWardenCosmoCore public cosmoCore;
17     IWardenCosmicBrain public cosmicBrain;
18     IWardenPostTrade public postTrade;
19
20     IWETH public immutable weth;
21     IERC20 private constant ETHER_ERC20 =
IERC20(0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE);
...(SNIP)...
```

Listing 1.9 The public state variable *weth*

WardenSwap1_5_L2.sol

```
69 constructor(
70     IWardenCosmoCore _cosmoCore,
71     IWardenCosmicBrain _cosmicBrain,
72     IWardenPostTrade _postTrade,
73     IWETH _weth
74 ) {
75     cosmoCore = _cosmoCore;
76     cosmicBrain = _cosmicBrain;
77     postTrade = _postTrade;
78     weth = _weth;
79
80     emit UpdatedWardenCosmoCore(_cosmoCore);
81     emit UpdatedWardenCosmicBrain(_cosmicBrain);
82     emit UpdatedWardenPostTrade(_postTrade);
83 }
```

Listing 1.10 The contract constructor is the only place that can assign the *weth* variable

No. 2	Potential Lock of Ether		
Risk	Low	Likelihood	Low
		Impact	Medium
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*receive() L: 583*		

Detailed Issue

The WardenSwap1_5_Aegis contract implements the *receive* function in line no. 583. The *receive* function receives the *Ethers* (native ETHs) from the *WETH* contract (after the best rate swapping) in case a user wants to receive the *Ether* as the destination token.

However, the *receive* function also receives *Ethers* from EOA (Externally Owned Account) wallets, resulting in the lock of *Ethers* by mistake. Even though the contract has implemented the *collectRemainingEther* function to enable the platform developer to withdraw the locked *Ethers*, the user mistake can be avoided by receiving only the *Ethers* from the *WETH* contract.

The code snippet below shows the associated *receive* function.

WardenSwap1_5_L2.sol

```
583  receive() external payable {}
```

Listing 2.1 The *receive* function

Recommendations

We advise enforcing receiving only the *Ethers* from the *WETH* contract by changing the *receive* function as follows.

WardenSwap1_5_L2.sol

```
583  receive() external payable {
584      require(msg.sender == address(weth), "WardenSwap: Receive Ether only from
     WETH");
585  }
```

Listing 2.2 The improved *receive* function

If the *receive* function is implemented according to our advice, the *collectRemainingEther* function (line no's. 572 - 580) can be removed since there will be no *Ether* withholding in the WardenSwap1_5_Aegis contract anymore.

Note that there is still a case that the *Ether* can be enforced to deposit to the WardenSwap1_5_Aegis contract by using the *selfdestruct* instruction. However, we consider that is a special case that would not mistakenly happen by an end-user.

WardenSwap1_5_L2.sol

```
572 function collectRemainingEther(
573     uint256 _amount
574 )
575     external
576     onlyOwner
577 {
578     (bool success, ) = msg.sender.call{value: _amount}(""); // Send back ether to
    sender
579     require(success, "WardenSwap: Transfer ether back to caller failed.");
580 }
```

Listing 2.3 The *collectRemainingEther* function that can be removed

Reassessment

The developer updated the *receive* function according to our advice to enforce receiving only the *Ethers* from the *WETH* contract.

No. 3	Recommended Updating The *PostTrade* Contract With Extra Care		
Risk	Low	Likelihood	Low
		Impact	Medium
Functionality is in use	In use	Status	Acknowledged
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*_postTradeAndCollectFee(IERC20, IERC20, uint256, uint256, address, address, bool) L: 585 - 615*		

Detailed Issue

The *_postTradeAndCollectFee* function (line no's. 585 - 615) is responsible for collecting a trading fee based on the source and destination token amounts and the user's current WAD tokens holding.

Specifically, the function will call the *postTradeAndFee* function of the *PostTrade* contract (line no's. 598 - 606) to calculate the trading fee. Then, the trading fee will be deducted from the destination token (line no's. 608 - 612) by the *_collectFee* function (line no's. 617 - 631).

In other words, the *_postTradeAndCollectFee* function will collect a trading fee according to the result from the *postTradeAndFee* function of the *PostTrade* contract. And, the trading fee calculation algorithm may be subject to change in the future based on Warden Protocol's tokenomics changes.

WardenSwap1_5_L2.sol

```
585 function _postTradeAndCollectFee(
586     IERC20      _src,
587     IERC20      _dest,
588     uint256     _srcAmount,
589     uint256     _destAmount,
590     address     _trader,
591     address     _receiver,
592     bool        _isSplit
593 )
594     private
595     returns (uint256 _newDestAmount)
596 {
597     // Collect fee
598     (uint256 fee, address feeWallet) = postTrade.postTradeAndFee(
599         _src,
600         _dest,
601         _srcAmount,
602         _destAmount,
603         _trader,
604         _receiver,
```

```
605             _isSplit
606         );
607         if (fee > 0) {
608             _collectFee(
609                 _dest,
610                 fee,
611                 feeWallet
612             );
613         }
614         return _destAmount - fee;
615     }
```

Listing 3.1 The *_postTradeAndCollectFee* function

Recommendations

Since the trading fee calculation algorithm may be subject to change in the future based on Warden Protocol's tokenomics changes, the platform developer has to exercise extra care when updating the *PostTrade* contract (new contract deployment), such as using the unit and integration testings, to ensure that the *postTradeAndFee* function will calculate the accurate trading fee.

Reassessment

The developer considered and acknowledged our recommendation to update the *PostTrade* contract with extra care.

No. 4	Unchecking Duplication of Trading Route On The *addTradingRoute* Function		
Risk	Low	Likelihood	Medium
		Impact	Low
Functionality is in use	In use	Status	Acknowledged
Associated Files	*libraries/WardenCosmoCore.sol*		
Locations	*addTradingRoute(string, IWardenTradingRoute) L: 27 - 40*		

Detailed Issue

The *addTradingRoute* function of the *WardenCosmoCore* contract allows the platform developer to add new trading routes (line no's. 27 - 40). However, the function lacked verifying the duplication of the trading route being added, as shown below.

WardenCosmoCore.sol

```
27  function addTradingRoute(
28      string calldata _name,
29      IWardenTradingRoute _routingAddress
30  )
31      external
32      onlyOwner
33  {
34      _tradingRoutes.push(Route({
35          name: _name,
36          enable: true,
37          route: _routingAddress
38      }));
39      emit AddedTradingRoute(msg.sender, _name, _routingAddress,
    _tradingRoutes.length - 1);
40  }
```

Listing 4.1 The *addTradingRoute* function

The duplicated trading routes may interfere with the process of best rate querying, such as slowing down a query or pushing a dispensable load to the best rate query engine.

Recommendations

We recommend detecting whether the given trading route is duplicated with an existing route. The trading route can be added only when there is no duplication. Consider the following code snippet.

WardenCosmoCore.sol

```
27  function checkTradingRouteDuplicate(IWardenTradingRoute _routingAddress) public
28  {
29      uint256 length = _tradingRoutes.length;
30      for (uint256 rid = 0; rid < length; rid++) {
31          require(_tradingRoutes[rid].route != _routingAddress, "Duplicate trading
    route");
32      }
33  }
34
35  function addTradingRoute(
36      string calldata _name,
37      IWardenTradingRoute _routingAddress
38  )
39      external
40      onlyOwner
41  {
42      checkTradingRouteDuplicate(_routingAddress);
43      _tradingRoutes.push(Route({
44          name: _name,
45          enable: true,
46          route: _routingAddress
47      }));
48      emit AddedTradingRoute(msg.sender, _name, _routingAddress,
    _tradingRoutes.length - 1);
49  }
```

Listing 4.2 The improved *addTradingRoute* function

Reassessment

The developer acknowledged the issue but decided to remain the original code to preserve the minimal gas use. The developer also took note of adding new trading routes with care.

No. 5	Unchecking Duplication of Trading Route On The *updateTradingRoute* Function		
Risk	Low	Likelihood	Medium
		Impact	Low
Functionality is in use	In use	Status	Acknowledged
Associated Files	*libraries/WardenCosmoCore.sol*		
Locations	*updateTradingRoute(uint256, string, IWardenTradingRoute) L: 48 - 59*		

Detailed Issue

The *updateTradingRoute* function of the *WardenCosmoCore* contract allows the platform developer to update an existing trading route (line no's. 48 - 59). However, the function lacked verifying the duplication of the trading route being updated, as shown below.

WardenCosmoCore.sol

```
48  function updateTradingRoute(
49      uint256 _index,
50      string calldata _name,
51      IWardenTradingRoute _route
52  )
53      external
54      onlyOwner
55  {
56      _tradingRoutes[_index].name = _name;
57      _tradingRoutes[_index].route = _route;
58      emit UpdatedTradingRoute(msg.sender, _name, _route, _index);
59  }
```

Listing 5.1 The *updateTradingRoute* function

The duplicated trading routes may interfere with the process of best rate querying, such as slowing down a query or pushing a dispensable load to the best rate query engine.

Recommendations

We recommend detecting whether the given trading route is duplicated with an existing route. The trading route can be updated only when there is no duplication. Consider the following code snippet.

WardenCosmoCore.sol

```
48  function checkTradingRouteDuplicate(IWardenTradingRoute _routingAddress) public
49  {
50      uint256 length = _tradingRoutes.length;
51      for (uint256 rid = 0; rid < length; rid++) {
52          require(_tradingRoutes[rid].route != _routingAddress, "Duplicate trading
    route");
53      }
54  }
55
56  function updateTradingRoute(
57      uint256 _index,
58      string calldata _name,
59      IWardenTradingRoute _route
60  )
61      external
62      onlyOwner
63  {
64      checkTradingRouteDuplicate(_route);
65      _tradingRoutes[_index].name = _name;
66      _tradingRoutes[_index].route = _route;
67      emit UpdatedTradingRoute(msg.sender, _name, _route, _index);
68  }
```

Listing 5.2 The improved *updateTradingRoute* function

Reassessment

The developer acknowledged the issue but decided to remain the original code to preserve the minimal gas use. The developer also took note of updating trading routes with care.

No. 6	The Split Volume May Be Inconsistent With The Actual Amount		
Risk	Low	Likelihood	Medium
		Impact	Low
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*_split2(uint256[],uint256[],address,uint256,address,address) L:380-406* *decodeLearnedIdsAndVolumns(bytes,uint256) L:339-346*		

Detailed Issue

The relevant split trading functions namely `splitTrades`, `splitTradesC1`, `splitTradesC2`, and `splitTradesC3` are consequently call the internal functions namely `_split2` and `decodeLearnedIdsAndVolumns`, allowing a user to split volumes of a trading token by percentages. Therefore, the split volumes should be accumulated to 100. However, the above-mentioned internal functions do not check the sum of percentage volumes which may lead to an accounting issue when the percentage of split volumes and the actual token amount is unmatched.

For example, in the `splitTrades` function, the `_volumns` variable is an array containing the split percentages of the `_totalSourceAmount` variable in which the `_learnedIds` variable can have 2 or more elements. There is a requirement that the length of the `_learnedIds` must be equal to the length of the `_volumns`. However, the function does not check that the sum of all percentage elements inside the `_volumns` variable should not exceed 100.

WardenSwap1_5_L2.sol

```
471 function splitTrades(
472     uint256[] memory  _learnedIds,
473     uint256[] memory  _volumns,
474     IERC20              _src,
475     uint256             _totalSrcAmount,
476     IERC20              _dest,
477     uint256             _minDestAmount,
478     address             _receiver
479 )
480     public
481     payable
482     nonReentrant
483     returns(uint256 _destAmount)
484 {
485     require(_learnedIds.length > 0, "WardenSwap: learnedIds can not be empty");
486     require(_learnedIds.length == _volumns.length, "WardenSwap: learnedIds and
    volumns lengths mismatched");
    ...(SNIP)...
```

Listing 6.1 Lengths of the *_volumns* and *learnedIds* variables are checked, but the sum of the *_volumns* variable is not checked

In the *_split2* function, the *_volumns* will be used to split the *_totalSrcAmount* by its percentages to actual split amounts. If the sum of all split volumes is more than 100, the actual amount and the percentage volumes may be inconsistent.

For example, given *[70, 40]* is the *_volumns* of the *_learnedIds[1, 2]* with 1000 *_totalSrcAmount*. The first learned ID would split 70% of the *_totalSrcAmount*, that is 700. The second learned ID uses the remaining *_totalSrcAmount*, that is 300 which is inconsistent with the specified 40%.

WardenSwap1_5_L2.sol

```
366 function _split2(
367     uint256[]   memory _learnedIds,
368     uint256[]   memory _volumns,
369     IERC20      _src,
370     uint256     _totalSrcAmount,
371     IERC20      _dest,
372     address     _fromAddress
373 )
374     private
375     returns (
376         uint256 _destAmount
377     )
378 {
379     // Trade with routes
380     uint256 amountRemain = _totalSrcAmount;
381     for (uint i = 0; i < _learnedIds.length; i++) {
```

```
382             uint256 amountForThisRound;
383             if (i == _learnedIds.length - 1) {
384                 amountForThisRound = amountRemain;
385             } else {
386                 amountForThisRound = _totalSrcAmount * _volumns[i] / 100;
387                 amountRemain = amountRemain - amountForThisRound;
388             }
    ...(SNIP)...
```

Listing 6.2 Amount is splitted by percentage volumes on the *_split2* function

Recommendations

The *amountForThisRound* variable should be calculated to be consistent with each split percentage from the *_volumns* variable.

Reassessment

The *splitTrades* function was fixed by checking that the *learnedIds* length must be equal to the *volumes* length - 1.

WardenSwap1_5_L2.sol

```
481 function splitTrades(
482     uint256[] memory  _learnedIds,
483     uint256[] memory  _volumes,
484     IERC20              _src,
485     uint256             _totalSrcAmount,
486     IERC20              _dest,
487     uint256             _minDestAmount,
488     address             _receiver
489 )
490     public
491     payable
492     nonReentrant
493     returns(uint256 _destAmount)
494 {
495     require(_learnedIds.length > 0, "WardenSwap: learnedIds can not be empty");
496     require(_learnedIds.length == _volumes.length - 1, "WardenSwap: learnedIds
    and volumes lengths mismatched");
    ...(SNIP)...
```

Listing 6.3 The fixed *splitTrades* function

No. 7	Lack of Some Edge Case In Input Validation		
Risk	Low	Likelihood	Low
		Impact	Medium
Functionality is in use	In use	Status	Fixed
Associated Files	*tools/WardenDataSerialize.sol*		
Locations	*tradeStrategiesSerialize(address, uint256, address, uint256, uint256[], address[])* *L: 88 - 148*		

Detailed Issue

On the *tradeStrategiesSerialize* function (line no's. 88 - 148), we found that the function lacked validating the following edge case:

_subRoutes.length - 1 == _correspondentTokens.length

The function could not detect the case when the *_correspondentTokens.length* is more than *_subRoutes.length*. The code snippet below shows the *tradeStrategiesSerialize* function.

WardenDataSerialize.sol

```
 88 function tradeStrategiesSerialize(
 89    address     _src,
 90    uint256     _srcAmount,
 91    address     _dest,
 92    uint256     _minDestAmount,
 93    uint256[]   calldata _subRoutes,
 94    address[]   calldata _correspondentTokens
 95 )
 96    external
 97    view
 98    returns(
 99        bytes memory _data
100    )
101 {
102    require(_srcAmount <= type(uint96).max,
    "WardenDataSerialize:tradeStrategiesSerialize _srcAmount is too large, uint96
    support only.");
103    require(_minDestAmount <= type(uint96).max,
    "WardenDataSerialize:tradeStrategiesSerialize _minDestAmount is too large,
    uint96 support only.");
104
105    // tokenLookup
```

```
106     uint256 srcIndex = addressTable.lookup(_src);
107     uint256 destIndex = addressTable.lookup(_dest);
108 
109     require(srcIndex <= type(uint24).max,
    "WardenDataSerialize:tradeStrategiesSerialize srcIndex is too large, uint24
    support only.");
110     require(destIndex <= type(uint24).max,
    "WardenDataSerialize:tradeStrategiesSerialize destIndex is too large, uint24
    support only.");
111 
112     _data = abi.encodePacked(
113         uint24(srcIndex),
114         uint24(destIndex),
115         uint96(_srcAmount),
116         uint96(_minDestAmount)
117     );
118 
119     require(_subRoutes.length < 64, "WardenDataSerialize:tradeStrategiesSerialize
    _subRoutes.length is too large, uint6 support only.");
120     uint8 routeLength = uint8(_subRoutes.length);
121     // token instruction: 2 (24-bit)
122     uint8 instructions = 2 << 6;
123     instructions += routeLength;
124 
125     _data = abi.encodePacked(
126         _data,
127         instructions
128     );
129 
130     for (uint256 i = 0; i < routeLength; i++) {
131         require(_subRoutes[i] <= type(uint16).max,
    "WardenDataSerialize:tradeStrategiesSerialize _subRoutes[i] is too large, uint16
    support only.");
132         _data = abi.encodePacked(
133             _data,
134             uint16(_subRoutes[i])
135         );
136     }
137 
138     for (uint256 i = 0; i < routeLength - 1; i++) {
139         address tokenAddress = address(_correspondentTokens[i]);
140         uint256 tokenId = addressTable.lookup(tokenAddress);
141         require(tokenId <= type(uint24).max,
    "WardenDataSerialize:tradeStrategiesSerialize tokenId is too large, uint24
    support only.");
142 
143         _data = abi.encodePacked(
144             _data,
145             uint24(tokenId)
146         );
147     }
```

```
148 }
```

Listing 7.1 The *tradeStrategiesSerialize* function

Recommendations

We recommend adding the lacking validation logic using the *require* statement as follows.

WardenDataSerialize.sol

```
88  function tradeStrategiesSerialize(
89      address     _src,
90      uint256     _srcAmount,
91      address     _dest,
92      uint256     _minDestAmount,
93      uint256[]   calldata _subRoutes,
94      address[]   calldata _correspondentTokens
95  )
96      external
97      view
98      returns(
99          bytes memory _data
100     )
101 {
102     require(_srcAmount <= type(uint96).max,
    "WardenDataSerialize:tradeStrategiesSerialize _srcAmount is too large, uint96
    support only.");
103     require(_minDestAmount <= type(uint96).max,
    "WardenDataSerialize:tradeStrategiesSerialize _minDestAmount is too large,
    uint96 support only.");
104
105     // tokenLookup
106     uint256 srcIndex = addressTable.lookup(_src);
107     uint256 destIndex = addressTable.lookup(_dest);
108
109     require(srcIndex <= type(uint24).max,
    "WardenDataSerialize:tradeStrategiesSerialize srcIndex is too large, uint24
    support only.");
110     require(destIndex <= type(uint24).max,
    "WardenDataSerialize:tradeStrategiesSerialize destIndex is too large, uint24
    support only.");
111
112     _data = abi.encodePacked(
113         uint24(srcIndex),
114         uint24(destIndex),
115         uint96(_srcAmount),
116         uint96(_minDestAmount)
117     );
118
```

```
119     require(_subRoutes.length - 1 == _correspondentTokens.length,
    "WardenDataSerialize:tradeStrategiesSerialize routes and tokens length
    mismatched");
120     require(_subRoutes.length < 64, "WardenDataSerialize:tradeStrategiesSerialize
    _subRoutes.length is too large, uint6 support only.");
121     uint8 routeLength = uint8(_subRoutes.length);
122     // token instruction: 2 (24-bit)
123     uint8 instructions = 2 << 6;
124     instructions += routeLength;
125
126     _data = abi.encodePacked(
127         _data,
128         instructions
129     );
130
131     for (uint256 i = 0; i < routeLength; i++) {
132         require(_subRoutes[i] <= type(uint16).max,
    "WardenDataSerialize:tradeStrategiesSerialize _subRoutes[i] is too large, uint16
    support only.");
133         _data = abi.encodePacked(
134             _data,
135             uint16(_subRoutes[i])
136         );
137     }
138
139     for (uint256 i = 0; i < routeLength - 1; i++) {
140         address tokenAddress = address(_correspondentTokens[i]);
141         uint256 tokenId = addressTable.lookup(tokenAddress);
142         require(tokenId <= type(uint24).max,
    "WardenDataSerialize:tradeStrategiesSerialize tokenId is too large, uint24
    support only.");
143
144         _data = abi.encodePacked(
145             _data,
146             uint24(tokenId)
147         );
148     }
149 }
```

Listing 7.2 The improved *tradeStrategiesSerialize* function

Reassessment

The developer added the recommended *require* statement to check the corresponding lengths of *_subRoutes* and *_correspondentTokens*.

WardenDataSerialize.sol

```
77  function tradeStrategiesSerialize(
78      address     _src,
79      uint256     _srcAmount,
80      address     _dest,
81      uint256     _minDestAmount,
82      uint256[]   calldata _subRoutes,
83      address[]   calldata _correspondentTokens
84  )
85      external
86      view
87      returns(
88          bytes memory _data
89      )
90  {
    ...(SNIP)...
108     require(_subRoutes.length - 1 == _correspondentTokens.length,
    "WardenDataSerialize:tradeStrategiesSerialize routes and tokens length
    mismatched");
109     require(_subRoutes.length < 64, "WardenDataSerialize:tradeStrategiesSerialize
    _subRoutes.length is too large, uint6 support only.");
    ...(SNIP)...
```

Listing 7.3 The fixed *tradeStrategiesSerialize* function

No. 8	The Compiler May Be Susceptible To The Publicly Disclosed Bugs		
Risk	Low	Likelihood	Low
		Impact	Medium
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol* *interface/IWardenCosmicBrainForL2.sol* *interface/IWardenCosmoCore0_8.sol* *interface/IWardenPostTrade.sol* *libraries/IWETH.sol* *libraries/IWardenTradingRoute0_8.sol* *libraries/WardenCosmoCore.sol* *libraries/WardenDataDeserialize.sol* *library/arbitrum/IArbAddressTable.sol* *library/byte/BytesLib.sol* *tools/WardenDataSerialize.sol*		
Locations	*WardenSwap1_5_L2.sol L: 2* *IWardenCosmicBrainForL2.sol L: 2* *IWardenCosmoCore0_8.sol L: 3* *IWardenPostTrade.sol L: 2* *IWETH.sol L: 3* *IWardenTradingRoute0_8.sol L: 3* *WardenCosmoCore.sol L: 3* *WardenDataDeserialize.sol L: 10* *IArbAddressTable.sol L: 2* *BytesLib.sol L: 5* *WardenDataSerialize.sol L: 10*		

Detailed Issue

The WardenSwap smart contracts use an outdated Solidity compiler version which may be susceptible to publicly disclosed vulnerabilities. The compiler version currently used by the WardenSwap is v0.8.0, which contains the list of known bugs as the following link:

`https://docs.soliditylang.org/en/v0.8.0/bugs.html`

The known bugs may not directly lead to the vulnerability, but it may increase an opportunity to trigger some attacks further.

An example of the Solidity code that does not use the latest patch version (v0.8.8) is shown below.

WardenCosmoCore.sol

```
1  //SPDX-License-Identifier: MIT
2
3  pragma solidity ^0.8.0;
4
5  import
   "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.2.0/contracts/ac
   cess/Ownable.sol";
6  import "../interface/IWardenCosmoCore0_8.sol";
7
8  contract WardenCosmoCore is Ownable, IWardenCosmoCore {
```

Listing 8.1 An example of the Solidity code that does not use the latest patch version (v0.8.8)

Recommendations

We recommend using the latest patch version, v0.8.8, that fixes all known bugs.

Reassessment

The developer fixed this issue by specifying the latest patch version, v0.8.8.

No. 9	The Compiler Is Not Locked To A Specific Version		
Risk	Low	Likelihood	Low
		Impact	Medium
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol* *interface/IWardenCosmicBrainForL2.sol* *interface/IWardenCosmoCore0_8.sol* *interface/IWardenPostTrade.sol* *libraries/IWETH.sol* *libraries/IWardenTradingRoute0_8.sol* *libraries/WardenCosmoCore.sol* *libraries/WardenDataDeserialize.sol* *library/arbitrum/IArbAddressTable.sol* *library/byte/BytesLib.sol* *tools/WardenDataSerialize.sol*		
Locations	*WardenSwap1_5_L2.sol L: 2* *IWardenCosmicBrainForL2.sol L: 2* *IWardenCosmoCore0_8.sol L: 3* *IWardenPostTrade.sol L: 2* *IWETH.sol L: 3* *IWardenTradingRoute0_8.sol L: 3* *WardenCosmoCore.sol L: 3* *WardenDataDeserialize.sol L: 10* *IArbAddressTable.sol L: 2* *BytesLib.sol L: 5* *WardenDataSerialize.sol L: 10*		

Detailed Issue

The WardenSwap smart contracts should be deployed with the compiler version used in the development and testing process.

The compiler version that is not strictly locked via the `pragma` statement may make the contracts incompatible against unforeseen circumstances.

An example of the Solidity code that is not locked to a specific version (e.g., using `>=` or `^` directive) is shown below.

WardenCosmoCore.sol

```
1  //SPDX-License-Identifier: MIT
2
3  pragma solidity ^0.8.0;
4
5  import
   "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.2.0/contracts/access/Ownable.sol";
6  import "../interface/IWardenCosmoCore0_8.sol";
7
8  contract WardenCosmoCore is Ownable, IWardenCosmoCore {
```

Listing 9.1 An example of the Solidity code that is not locked to a specific version

Recommendations

We recommend locking the *pragma* version like the example code snippet below.

```
pragma solidity 0.8.8;
// or
pragma solidity =0.8.8;
contract SemVerFloatingPragmaFixed {
}
```

Reference: https://swcregistry.io/docs/SWC-103

Reassessment

The developer fixed this issue by locking the *pragma* version to the latest patch version, v0.8.8.

No. 10	Transparency Improvement For The *collectRemainingToken* Function		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*collectRemainingToken(IERC20, uint256) L: 561 - 569*		

Detailed Issue

The *collectRemainingToken* function lets the platform developer collect the leftover ERC20 tokens from the WardenSwap1_5_Aegis contract. This function is helpful in case a user mistakenly transfers ERC20 tokens to the contract.

However, the function does not implement an event emission after the token collecting, affecting transparency and traceability. The *collectRemainingToken* function is shown below.

WardenSwap1_5_L2.sol

```
561 function collectRemainingToken(
562     IERC20  _token,
563     uint256 _amount
564 )
565     external
566     onlyOwner
567 {
568     _token.safeTransfer(msg.sender, _amount);
569 }
```

Listing 10.1 The *collectRemainingToken* function

Recommendations

We recommend emitting the event *CollectedRemainingToken* on the *collectRemainingToken* function to improve transparency and traceability. See the improved function below.

WardenSwap1_5_L2.sol

```
561 function collectRemainingToken(
562     IERC20  _token,
563     uint256 _amount
564 )
565     external
566     onlyOwner
567 {
568     _token.safeTransfer(msg.sender, _amount);
569     emit CollectedRemainingToken(address(_token), _amount, msg.sender);
570 }
```

Listing 10.2 The improved *collectRemainingToken* function

Reassessment

The developer emitted the *CollectedRemainingToken* event as per our recommendation.

WardenSwap1_5_L2.sol

```
571 function collectRemainingToken(
572     IERC20  _token,
573     uint256 _amount
574 )
575     external
576     onlyOwner
577 {
578     _token.safeTransfer(msg.sender, _amount);
579     emit CollectedRemainingToken(address(_token), _amount);
580 }
```

Listing 10.3 The fixed *collectRemainingToken* function

No. 11	Transparency Improvement For The `collectRemainingEther` Function		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*collectRemainingEther(uint256) L: 572 - 580*		

Detailed Issue

The `collectRemainingEther` function lets the platform developer collect the leftover Ether (native ETH) from the WardenSwap1_5_Aegis contract. This function is helpful in case a user mistakenly transfers Ethers to the contract.

However, the function does not implement an event emission after the coin collecting, affecting transparency and traceability. The `collectRemainingEther` function is shown below.

WardenSwap1_5_L2.sol

```
572 function collectRemainingEther(
573     uint256 _amount
574 )
575     external
576     onlyOwner
577 {
578     (bool success, ) = msg.sender.call{value: _amount}(""); // Send back ether to
    sender
579     require(success, "WardenSwap: Transfer ether back to caller failed.");
580 }
```

Listing 11.1 The `collectRemainingEther` function

Recommendations

We recommend emitting the event *CollectedRemainingEther* on the *collectRemainingEther* function to improve transparency and traceability. See the improved function below.

WardenSwap1_5_L2.sol

```
572 function collectRemainingEther(
573     uint256 _amount
574 )
575     external
576     onlyOwner
577 {
578     (bool success, ) = msg.sender.call{value: _amount}(""); // Send back ether to
    sender
579     require(success, "WardenSwap: Transfer ether back to caller failed.");
580     emit CollectedRemainingEther(_amount, msg.sender);
581 }
```

Listing 11.2 The improved *collectRemainingEther* function

Reassessment

The developer emitted the *CollectedRemainingEther* event as per our recommendation.

WardenSwap1_5_L2.sol

```
583 function collectRemainingEther(
584     uint256 _amount
585 )
586     external
587     onlyOwner
588 {
589     (bool success, ) = msg.sender.call{value: _amount}(""); // Send back ether to
    receiver
590     require(success, "WardenSwap: Transfer ether back to receiver failed.");
591     emit CollectedRemainingEther(_amount);
592 }
```

Listing 11.3 The fixed *collectRemainingEther* function

No. 12	Gas Optimization and Readability Improvement On The *_split2* Function		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*_split2(uint256[], uint256[], IERC20, uint256, IERC20, address)* *L: 381, 383, 387, and 396*		

Detailed Issue

We found that the source code of the *_split2* function can be improved to optimize gas usage and enhance code readability. The associated line numbers that can be improved include 381, 383, 387, and 396. The following shows the *_split2* function.

WardenSwap1_5_L2.sol

```
366 function _split2(
367     uint256[]   memory _learnedIds,
368     uint256[]   memory _volumns,
369     IERC20      _src,
370     uint256     _totalSrcAmount,
371     IERC20      _dest,
372     address     _fromAddress
373 )
374     private
375     returns (
376         uint256 _destAmount
377     )
378 {
379     // Trade with routes
380     uint256 amountRemain = _totalSrcAmount;
381     for (uint i = 0; i < _learnedIds.length; i++) {
382         uint256 amountForThisRound;
383         if (i == _learnedIds.length - 1) {
384             amountForThisRound = amountRemain;
385         } else {
386             amountForThisRound = _totalSrcAmount * _volumns[i] / 100;
387             amountRemain = amountRemain - amountForThisRound;
388         }
389
390         bytes32 learnedHash = cosmicBrain.learnedHashes(_learnedIds[i]);
391         (
```

```
392             uint256[]   memory subRoutes,
393             IERC20[]    memory correspondentTokens
394         ) = cosmicBrain.fetchRoutesAndTokens(learnedHash);
395
396         _destAmount = _destAmount +
397             _tradeStrategies(
398                 _src,
399                 amountForThisRound,
400                 _dest,
401                 subRoutes,
402                 correspondentTokens,
403                 _fromAddress
404             )
405         ;
406     }
407 }
```

Listing 12.1 The *_split2* function that can be improved

Recommendations

We recommend changing the associated source code to optimize gas usage and enhance code readability as follows.

WardenSwap1_5_L2.sol

```
366 function _split2(
367     uint256[]   memory _learnedIds,
368     uint256[]   memory _volumns,
369     IERC20      _src,
370     uint256     _totalSrcAmount,
371     IERC20      _dest,
372     address     _fromAddress
373 )
374     private
375     returns (
376         uint256 _destAmount
377     )
378 {
379     // Trade with routes
380     uint256 amountRemain = _totalSrcAmount;
381     uint256 learnedIdsLenght = _learnedIds.length;
382     for (uint i = 0; i < learnedIdsLenght; i++) {
383         uint256 amountForThisRound;
384         if (i == learnedIdsLenght - 1) {
385             amountForThisRound = amountRemain;
386         } else {
387             amountForThisRound = _totalSrcAmount * _volumns[i] / 100;
388             amountRemain -= amountForThisRound;
```

```
389         }
390
391         bytes32 learnedHash = cosmicBrain.learnedHashes(_learnedIds[i]);
392         (
393             uint256[]   memory subRoutes,
394             IERC20[]    memory correspondentTokens
395         ) = cosmicBrain.fetchRoutesAndTokens(learnedHash);
396
397         _destAmount += _tradeStrategies(
398                 _src,
399                 amountForThisRound,
400                 _dest,
401                 subRoutes,
402                 correspondentTokens,
403                 _fromAddress
404             )
405         ;
406     }
407 }
```

Listing 12.2 The optimized *_split2* function

Reassessment

The developer updated the associated function according to our recommendation. The developer also renamed the function from *_split2* to *_loopSplit* to describe its functionality better.

WardenSwap1_5_L2.sol

```
375 function _loopSplit(
376     uint256[]   memory _learnedIds,
377     uint256[]   memory _volumes,
378     IERC20      _src,
379     uint256     _totalSrcAmount,
380     IERC20      _dest,
381     address     _fromAddress
382 )
383     private
384     returns (
385         uint256 _destAmount
386     )
387 {
388     // Trade with routes
389     uint256 amountRemain = _totalSrcAmount;
390     uint256 learnedIdsLenght = _learnedIds.length;
391     for (uint i = 0; i < learnedIdsLenght; i++) {
392         uint256 amountForThisRound;
393         if (i == learnedIdsLenght - 1) {
394             amountForThisRound = amountRemain;
```

```
395         } else {
396             amountForThisRound = _totalSrcAmount * _volumes[i] / 100;
397             amountRemain -= amountForThisRound;
398         }
399 
400         bytes32 learnedHash = cosmicBrain.learnedHashes(_learnedIds[i]);
401         (
402             uint256[]   memory subRoutes,
403             IERC20[]    memory correspondentTokens
404         ) = cosmicBrain.fetchRoutesAndTokens(learnedHash);
405 
406         _destAmount +=
407             _tradeStrategies(
408                 _src,
409                 amountForThisRound,
410                 _dest,
411                 subRoutes,
412                 correspondentTokens,
413                 _fromAddress
414             )
415         ;
416     }
417 }
```

Listing 12.3 The fixed *_loopSplit* function

No. 13	Inconsistent Comments/Error Messages With The Code		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol* *libraries/WardenCosmoCore.sol*		
Locations	*WardenSwap1_5_L2.sol L: 266, 267, 268, 510, 511, 512, 554, 555, 578, 579, and 625* *WardenCosmoCore.sol L: 25, 64, and 80*		

Detailed Issue

We found comments and error messages inconsistent with the source code, leading to misunderstandings among users.

The associated comments and error messages are in the following line numbers in WardenSwap1_5_L2.sol: 266, 267, 268, 510, 511, 512, 554, 555, 578, 579, and 625, and in the following line numbers in WardenCosmoCore.sol: 25, 64, and 80. The following shows an example of an inconsistent comment.

WardenSwap1_5_L2.sol

```
617 function _collectFee(
618     IERC20  _token,
619     uint256 _fee,
620     address _feeWallet
621 )
622     private
623 {
624     if (ETHER_ERC20 == _token) {
625         (bool success, ) = payable(_feeWallet).call{value: _fee}(""); // Send
    back ether to sender
626         require(success, "Transfer fee of ether failed.");
627     } else {
628         _token.safeTransfer(_feeWallet, _fee);
629     }
630     emit CollectFee(_token, _feeWallet, _fee);
631 }
```

Listing 13.1 An example of an inconsistent comment

Recommendations

We recommend changing the associated comments and error messages to reflect how the source code is doing.

Reassessment

The developer changed all associated comments and error messages to describe the source code better.

WardenSwap1_5_L2.sol

```
631 function _collectFee(
632     IERC20  _token,
633     uint256 _fee,
634     address _feeWallet
635 )
636     private
637 {
638     if (ETHER_ERC20 == _token) {
639         (bool success, ) = payable(_feeWallet).call{value: _fee}(""); // Send
    ether to fee collector
640         require(success, "Transfer fee of ether failed.");
641     } else {
642         _token.safeTransfer(_feeWallet, _fee); // Send token to fee collector
643     }
644     emit CollectFee(_token, _feeWallet, _fee);
645 }
```

Listing 13.2 An example of updated comments

No. 14	Recommended Explicit Trading Route Validation Checks		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Acknowledged
Associated Files	*libraries/WardenCosmoCore.sol*		
Locations	*WardenCosmoCore.sol L: 12, 17, 56, 73, 89, 104, and 121*		

Detailed Issue

The implementation of the *WardenCosmoCore* contract relies on the implicit compiler-generated bound-checks of the *_tradingRoutes* array to ensure that the routing index is within the array range *[0, _tradingRoutes.length - 1]*.

The associated checks include the following line numbers 12, 17, 56, 73, 89, 104, and 121. The code snippet below is an example of the implicit bound check.

WardenCosmoCore.sol

```
48  function updateTradingRoute(
49      uint256 _index,
50      string calldata _name,
51      IWardenTradingRoute _route
52  )
53      external
54      onlyOwner
55  {
56      _tradingRoutes[_index].name = _name;
57      _tradingRoutes[_index].route = _route;
58      emit UpdatedTradingRoute(msg.sender, _name, _route, _index);
59  }
```

Listing 14.1 The *updateTradingRoute* function that uses the implicit bound check

The explicit sanity checks may be required to enable the contract to handle a revert transaction with a proper error message for a better debugging solution.

Recommendations

We recommend implementing a *validateTradingRoute* modifier and attaching the modifier to the associated functions. Consider the following example code snippet.

WardenCosmoCore.sol

```
48  modifier validateTradingRoute(uint256 _index) {
49      require (_index < _tradingRoutes.length, "Trading route index out of bounds")
    ;
50      _;
51  }
52
53  function updateTradingRoute(
54      uint256 _index,
55      string calldata _name,
56      IWardenTradingRoute _route
57  )
58      external
59      onlyOwner
60      validateTradingRoute(_index)
61  {
62      _tradingRoutes[_index].name = _name;
63      _tradingRoutes[_index].route = _route;
64      emit UpdatedTradingRoute(msg.sender, _name, _route, _index);
65  }
```

Listing 14.2 The improved *updateTradingRoute* function that uses the explicit sanity check

With the explicit sanity checks, the contract can handle a revert transaction with a proper error message, improving a transaction debugging solution.

Reassessment

The developer acknowledged this issue but decided to remain the original code to preserve the minimal gas use.

No. 15	Unchecked Call Return Value		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Acknowledged
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*_tradeStrategiesWithSafeGuard(address,uint256,address,uint256,uint256[],address[],address,uint256) L:276*		

Detailed Issue

The *_tradeStrategiesWithSafeGuard* function does not check the return value when performing an external call which may lead to unexpected behaviors of the application logic.

The external functions will return the data and result. The result indicates whether the call fails or succeeds which is represented by the following boolean value:

- 0x0 -> false -> fails
- 0x1 -> true -> succeeds

If the return value is not checked, the remaining code would be continuously executed even if the external call is accidentally or deliberately failed, this may lead to unexpected behaviors of the subsequent logic.

On the *IWardenCosmicBrainForL2* interface, the *trainTradingPair* function returns the boolean value, *_isAlreadyLearned*.

IWardenCosmicBrainForL2.sol

```
14  function trainTradingPair(
15      IERC20      _src,
16      IERC20      _dest,
17      uint256     _srcAmount,
18      uint256     _destAmount,
19      uint256     _learnedId
20  )
21      external
22      returns (bool _isAlreadyLearned);
```

Listing 15.1 The return value of the *trainTradingPair* function

The *trainTradingPair* function is called by the *_tradeStrategiesWithSafeGuard* function but the return value is not checked.

WardenSwap1_5_L2.sol

```
196 function _tradeStrategiesWithSafeGuard(
197     IERC20      _src,
198     uint256     _srcAmount,
199     IERC20      _dest,
200     uint256     _minDestAmount,
201     uint256[]   memory _subRoutes,
202     IERC20[]    memory _correspondentTokens,
203     address     _receiver,
204     uint256     _learnedId
205 )
    ...(SNIP)...
276     cosmicBrain.trainTradingPair(
277         _src,
278         _dest,
279         _srcAmount,
280         _destAmount,
281         learnedId
282     );
    ...(SNIP)..
```

Listing 15.2 The *trainTradingPair* function is called, but the return value is not checked

Recommendations

The return value of the external call should be checked and handled. The function needs to decide and clarify the expected value returned by the callee. The unexpected return value should be handled depending on the application logic.

Reassessment

The developer acknowledged this issue but decided to remain the original code.

No. 16	Gas Optimization On Redundant Code		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Acknowledged
Associated Files	*libraries/WardenDataDeserialize.sol*		
Locations	*decodeSubRoutesAndCorrespondentTokens(bytes,uint256) L:278-279* *lookupSrcDestReceiverAddresses(bytes,uint256) L:439* *_decodeSrcMinAmountsLearnedId(bytes,uint256,bool) L:186* *decodeLearnedIdsAndVolumns(bytes,uint256) L:329* *lookupSrcDestReceiverAddresses(bytes,uint256) L:439*		

Detailed Issue

The excessive code might consume more gas. The examples of redundant behaviours are as follows:

- The value is recomputed even if it is previously computed.
- The comparison of booleans.
- The unreachable code.

In line 275, 6 bits of the `instructions` variable are removed by right shifting. Therefore, the 6 most significant bits after bit shifting will only be 0. Therefore, the AND (&) instruction is unnecessary to zero out the unwanted bits.

To illustratrate, suppose the 8 bits of the `instructions` variable is 0xFF. When the code in line 275 is executed, the `instructions` will be 0000 0011 which is equal to 0x03. In line 277, the function tries to zero out the 6 most significant bits to obtain only 2 least significant bits and assign it to the variable `tokenInstruction`. However, the 6 most significant bits are already zeroed since the right shifting. Therefore, zeroing out the unwanted bits by 0x03 is unnecessary.

In addition, in line 278, the `instructions` variable is shifted to the right to remove the 2 least significant bits from the `instructions` itself. But, the `instructions` variable is not used anymore after this operation. The code in line 278, therefore, is unnecessary.

WardenDataDeserialize.sol

```
234 function decodeSubRoutesAndCorrespondentTokens(
235     bytes memory _data,
236     uint256 _cursor
237 )
238     public
239     view
240     returns (
241         uint256[]   memory _subRoutes, // 16-bit, 65,536 possible
242         IERC20[]    memory _correspondentTokens,
243         uint256             _newCursor
244     )
245 {
    ...(SNIP)...
271     uint8 instructions = _data.toUint8(_cursor);
272     _cursor += 1;
273
274     uint256 routeLength = instructions & 0x3F;
275     instructions = instructions >> 6;
276
277     uint8 tokenInstruction = instructions & 0x03;
278     instructions = instructions >> 2;
    ...(SNIP)...
```

Listing 16.1 An example code being redundant when decompressing the variable *instructions*

Recommendations

Remove the redundant code for saving gas.

In the example case, consider removing AND (&) operation from the last instruction extraction (line no. 277) since the prior right shift (line no. 275) already removes unwanted bits. The code could be simplified to

uint8 tokenInstruction = instructions;

and remove the unnecessary right shift operation in line 278.

Reassessment

The developer acknowledged this issue but decided to remain the original code.

No. 17	Duplicate Function Implementation		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Fixed
Associated Files	*tools/WardenDataSerialize.sol*		
Locations	*toBytes32(bytes, uint256) L: 26 - 35*		

Detailed Issue

The implementation of the *toBytes32* function (line no's. 26 - 35) is duplicated with the function in the imported *BytesLib* library. The following shows the duplicated function.

WardenDataSerialize.sol

```
26 function toBytes32(bytes memory _bytes, uint256 _start) internal pure returns
   (bytes32) {
27    require(_bytes.length >= _start + 32, "toBytes32_outOfBounds");
28    bytes32 tempBytes32;
29
30    assembly {
31        tempBytes32 := mload(add(add(_bytes, 0x20), _start))
32    }
33
34    return tempBytes32;
35 }
```

Listing 17.1 The duplicated *toBytes32* function

The *toBytes32* function is used in the *tradeWithLearnedSerialize* function (line no. 84) to convert the *tradeWithLearned* data in *bytes* to *bytes32*, as shown below.

WardenDataSerialize.sol

```
49 function tradeWithLearnedSerialize(
50    address    _src,
51    uint256    _srcAmount,
52    address    _dest,
53    uint256    _minDestAmount,
54    uint256    _learnedId
55 )
56    external
```

```
57      view
58      returns(
59          bytes32 _compressedData
60      )
61  {
62      require(_srcAmount <= type(uint96).max,
    "WardenDataSerialize:tradeWithLearnedSerialize _srcAmount is too large, uint96
    support only.");
63      require(_minDestAmount <= type(uint96).max,
    "WardenDataSerialize:tradeWithLearnedSerialize _minDestAmount is too large,
    uint96 support only.");
64      require(_learnedId <= type(uint16).max,
    "WardenDataSerialize:tradeWithLearnedSerialize _learnedId is too large, uint16
    support only.");
65
66      // tokenLookup
67      uint256 srcIndex = addressTable.lookup(_src);
68      uint256 destIndex = addressTable.lookup(_dest);
69
70      require(srcIndex <= type(uint24).max,
    "WardenDataSerialize:tradeWithLearnedSerialize srcIndex is too large, uint24
    support only.");
71      require(destIndex <= type(uint24).max,
    "WardenDataSerialize:tradeWithLearnedSerialize destIndex is too large, uint24
    support only.");
72
73      bytes memory _bytes = abi.encodePacked(
74          uint24(srcIndex),
75          uint24(destIndex),
76          uint96(_srcAmount),
77          uint96(_minDestAmount),
78          uint16(_learnedId)
79      );
80
81      require(_bytes.length == 32, "WardenDataSerialize:toBytes32 length is not
    32");
82
83
84      return toBytes32(_bytes, 0);
85  }
```

Listing 17.2 The *tradeWithLearnedSerialize* function

Recommendations

The mentioned *toBytes32* function can be removed, and we can use the function in the imported *BytesLib* library instead, as shown below.

WardenDataSerialize.sol

```
49  function tradeWithLearnedSerialize(
50      address    _src,
51      uint256   _srcAmount,
52      address    _dest,
53      uint256   _minDestAmount,
54      uint256   _learnedId
55  )
56      external
57      view
58      returns(
59          bytes32 _compressedData
60      )
61  {
62      require(_srcAmount <= type(uint96).max,
    "WardenDataSerialize:tradeWithLearnedSerialize _srcAmount is too large, uint96
    support only.");
63      require(_minDestAmount <= type(uint96).max,
    "WardenDataSerialize:tradeWithLearnedSerialize _minDestAmount is too large,
    uint96 support only.");
64      require(_learnedId <= type(uint16).max,
    "WardenDataSerialize:tradeWithLearnedSerialize _learnedId is too large, uint16
    support only.");
65
66      // tokenLookup
67      uint256 srcIndex = addressTable.lookup(_src);
68      uint256 destIndex = addressTable.lookup(_dest);
69
70      require(srcIndex <= type(uint24).max,
    "WardenDataSerialize:tradeWithLearnedSerialize srcIndex is too large, uint24
    support only.");
71      require(destIndex <= type(uint24).max,
    "WardenDataSerialize:tradeWithLearnedSerialize destIndex is too large, uint24
    support only.");
72
73      bytes memory _bytes = abi.encodePacked(
74          uint24(srcIndex),
75          uint24(destIndex),
76          uint96(_srcAmount),
77          uint96(_minDestAmount),
78          uint16(_learnedId)
79      );
80
81      require(_bytes.length == 32, "WardenDataSerialize:toBytes32 length is not
```

```
    32");
82
83      return _bytes.toBytes32(0);
84  }
```

Listing 17.3 The improved *tradeWithLearnedSerialize* function

Using the standard *BytesLib* library function will benefit an aspect of code maintenance and bug fixes.

Reassessment

The developer updated the associated source code as per our recommendation.

No. 18	Generic Typographic Error		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	***For example:*** *_split2(uint256[],uint256[],address,uint256,address,address) L:368* *_splitTradesWithSafeGuard(uint256[],uint256[],address,uint256,address) L:411*		

Detailed Issue

Some variables and comments contain typos which may lead to the name confusion, or slightly increase the time when debugging or maintaining the source code.

For example, the `_volumns` or `_volumn` is misspelled in several locations.

WardenSwap1_5_L2.sol

```
409 function _splitTradesWithSafeGuard(
410     uint256[] memory  _learnedIds,
411     uint256[] memory  _volumns,
412     IERC20              _src,
413     uint256             _totalSrcAmount,
414     IERC20              _dest
415 )
416     private
417     returns(uint256 _destAmount)
418 {
    ...(SNIP)...
439      _destAmount = _split2(
440         _learnedIds,
441         _volumns,
442         adjustedSrc,
443         _totalSrcAmount,
444         adjustedDest,
445         fromAddress
446     );
```

Listing 18.1 An example variable that is misspelled

Recommendations

Review the variables and comments and revise the incorrect or typographical words. In the example case, consider renaming the variable from "*_volumns*" to "*_volumes*" instead.

Reassessment

The misspelled variables were renamed to "volume" or "volumes".

WardenSwap1_5_L2.sol

```
419 function _splitTradesWithSafeGuard(
420     uint256[] memory  _learnedIds,
421     uint256[] memory  _volumes,
422     IERC20              _src,
423     uint256             _totalSrcAmount,
424     IERC20              _dest
425 )
426     private
427     returns(uint256 _destAmount)
428 {
    ...(SNIP)...
449     _destAmount = _loopSplit(
450         _learnedIds,
451         _volumes,
452         adjustedSrc,
453         _totalSrcAmount,
454         adjustedDest,
455         fromAddress
456     );
```

Listing 18.2 An example variable that was renamed

No. 19	Misleading Function Name		
Risk	Informational	Likelihood	Low
		Impact	Low
Functionality is in use	In use	Status	Fixed
Associated Files	*WardenSwap1_5_L2.sol*		
Locations	*_split2(uint256[], uint256[], IERC20, uint256, IERC20, address) L: 366 - 407*		

Detailed Issue

The *_split2* function splits trading into multiple trading routes according to the length of the *_learnedIds* variable (line no. 381). The function actually supports multiple trading routes, which can be more than two routes. At this point, it comes to our attention that the function name, *_split2*, is misleading. The misleading function name may hinder the source code maintenance process as well as the understanding of the source code.

WardenSwap1_5_L2.sol

```
366 function _split2(
367     uint256[]   memory _learnedIds,
368     uint256[]   memory _volumns,
369     IERC20      _src,
370     uint256     _totalSrcAmount,
371     IERC20      _dest,
372     address     _fromAddress
373 )
374     private
375     returns (
376         uint256 _destAmount
377     )
378 {
379     // Trade with routes
380     uint256 amountRemain = _totalSrcAmount;
381     for (uint i = 0; i < _learnedIds.length; i++) {
382         uint256 amountForThisRound;
383         if (i == _learnedIds.length - 1) {
384             amountForThisRound = amountRemain;
385         } else {
386             amountForThisRound = _totalSrcAmount * _volumns[i] / 100;
387             amountRemain = amountRemain - amountForThisRound;
388         }
389
390         bytes32 learnedHash = cosmicBrain.learnedHashes(_learnedIds[i]);
```

```
391         (
392             uint256[]   memory subRoutes,
393             IERC20[]    memory correspondentTokens
394         ) = cosmicBrain.fetchRoutesAndTokens(learnedHash);
395 
396         _destAmount = _destAmount +
397             _tradeStrategies(
398                 _src,
399                 amountForThisRound,
400                 _dest,
401                 subRoutes,
402                 correspondentTokens,
403                 _fromAddress
404             )
405         ;
406     }
407 }
```

Listing 19.1 The *_split2* function

Recommendations

We recommend renaming the *_split2* function to reflect its functionality to improve the source code maintenance process as well as the understanding of the source code.

Reassessment

The *_split2* function was renamed to *_loopSplit* to describe its functionality better.

WardenSwap1_5_L2.sol

```
375 function _loopSplit(
376     uint256[]   memory _learnedIds,
377     uint256[]   memory _volumes,
378     IERC20      _src,
379     uint256     _totalSrcAmount,
380     IERC20      _dest,
381     address     _fromAddress
382 )
383     private
384     returns (
385         uint256 _destAmount
386     )
387 {
    ...(SNIP)...
```

Listing 19.2 The renamed function, *_loopSplit*

Appendix

About Us

Founded in 2020, Valix Consulting is a blockchain and smart contract security firm offering a wide range of cybersecurity consulting services such as blockchain and smart contract security consulting, smart contract security review, and smart contract security audit.

Our team members are passionate cybersecurity professionals and researchers in areas of private and public blockchain technology, smart contract, and decentralized application (DApp).

We provide a service for assessing and certifying the security of smart contracts. Our service also includes recommendations on smart contracts' security and gas optimization to bring the most benefit to users and platform creators.

Contact Information


info@valix.io


https://www.facebook.com/ValixConsulting


https://twitter.com/ValixConsulting


https://medium.com/valixconsulting

References

Title	Link
OWASP Risk Rating Methodology	https://owasp.org/www-community/OWASP_Risk_Rating_Methodology
Smart Contract Weakness Classification and Test Cases	https://swcregistry.io/