Warden

WardenSwap Protocol

Security Assessment

May 8th, 2021

Audited By:
Angelos Apostolidis @ CertiK
angelos.apostolidis@certik.org
Reviewed By:
Sheraz Arshad @ CertiK
sheraz.arshad@certik.org

Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are 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 review.

CertiK Reports do 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.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing 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.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Warden - WardenSwap Protocol
Description	WardenSwap decentralized exchange (DEX) prices from multiple pools to find the best price across all pools
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 61bd6cf20ef0297ee61de5ed48869e317760e9a1 2. b3011927b259ddaefbad1599fd43f5a50b6d45b4

Audit Summary

Delivery Date	May 8th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	April 19th, 2021 - May 8th, 2021

Vulnerability Summary

Total Issues	26
● **Total Critical**	0
● **Total Major**	0
● **Total Medium**	2
● **Total Minor**	7
● **Total Informational**	17

Executive Summary

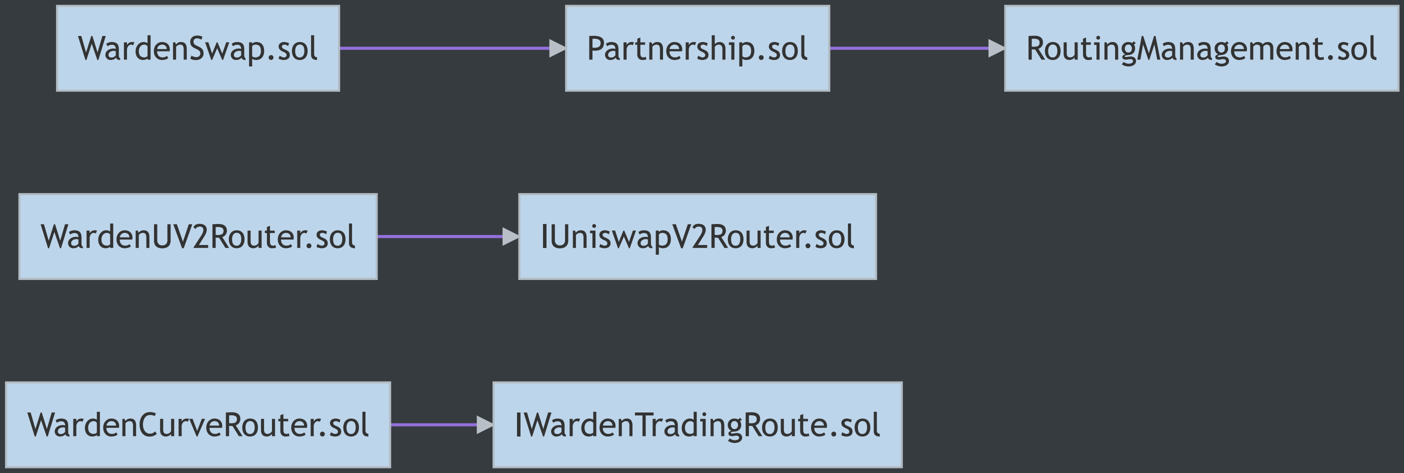
The report represents the results of CertiK's engagement with Warden on the implementation of their WardenSwap decentralized exchange smart contracts.

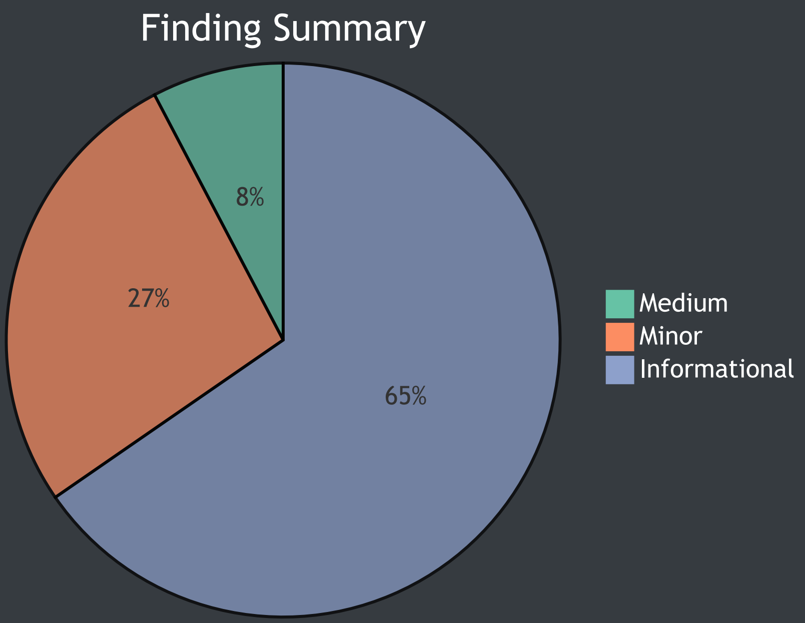
No notable vulnerabilities were identified in the codebase and it makes use of the latest security principles and styleguidelines. There were certain security principles that can optionally be applied to the codebase to fortify thecodebase to a greater extent.

Files In Scope

ID	Contract	Location
PAR	Partnership.sol	contracts/Partnership.sol
RMT	RoutingManagement.sol	contracts/RoutingManagement.sol
WSP	WardenSwap.sol	contracts/WardenSwap.sol
WBR	WardenBestRateQuery.sol	contracts/bestRate/WardenBestRateQuery.sol
IUV	IUniswapV2Router.sol	contracts/interfaces/IUniswapV2Router.sol
IWT	IWardenTradingRoute.sol	contracts/interfaces/IWardenTradingRoute.sol
CSR	CurveSusdRoute.sol	contracts/routes/CurveSusdRoute.sol
SRE	SpartanRoute.sol	contracts/routes/SpartanRoute.sol
SRT	SushiswapRoute.sol	contracts/routes/SushiswapRoute.sol
UVP	UniswapV2PoolToPoolTokenEthTokenRoute.sol	contracts/routes/UniswapV2PoolToPoolTokenEthTokenRoute.sol
UVR	UniswapV2Route.sol	contracts/routes/UniswapV2Route.sol
UVT	UniswapV2TokenEthTokenRoute.sol	contracts/routes/UniswapV2TokenEthTokenRoute.sol
WCR	WardenCurveRouter.sol	contracts/routes/WardenCurveRouter.sol
WUV	WardenUV2Router.sol	contracts/routes/WardenUV2Router.sol

File Dependency Graph





Manual Review Findings

ID	Title	Type	Severity	Resolved
PAR-01M	Inexistent Input Sanitization	Logical Issue	🔵 Minor	✓
PAR-02M	Return Variable Utilization	Gas Optimization	🟢 Informational	✓
RMT-01M	Function Visibility Optimization	Gas Optimization	🟢 Informational	✓
WBR-01M	Return Variable Utilization	Gas Optimization	🟢 Informational	✓
CSR-01M	Potential Underflow	Logical Issue	🟡 Medium	✓
UVP-01M	Ambiguous `payable` Function	Logical Issue	🔵 Minor	⏱
UVT-01M	Ambiguous `payable` Function	Logical Issue	🔵 Minor	⏱
WCR-01M	Potential Underflow	Logical Issue	🟡 Medium	✓
WUV-01M	Redundant `array` Look Up	Gas Optimization	🟢 Informational	✓

Static Analysis Findings

ID	Title	Type	Severity	Resolved
PAR-01S	Declaration Naming Convention	Coding Style	🟢 Informational	✓
RMT-01S	Boolean Comparison	Gas Optimization	🟢 Informational	✓
WSP-01S	External Calls Inside a Loop	Volatile Code	🟢 Informational	◷
WBR-01S	External Calls Inside a Loop	Volatile Code	🟢 Informational	◷
WBR-02S	Declaration Naming Convention	Coding Style	🟢 Informational	✓
CSR-01S	Potential Lock of Ether	Logical Issue	🔵 Minor	◷
CSR-02S	Declaration Naming Convention	Coding Style	🟢 Informational	✓
SRE-01S	Declaration Naming Convention	Coding Style	🟢 Informational	✓
SRT-01S	Declaration Naming Convention	Coding Style	🟢 Informational	✓
UVP-01S	Potential Lock of Ether	Logical Issue	🔵 Minor	◷
UVP-02S	Declaration Naming Convention	Coding Style	🟢 Informational	✓
UVR-01S	Declaration Naming Convention	Coding Style	🟢 Informational	✓
UVT-01S	Potential Lock of Ether	Logical Issue	🔵 Minor	◷
UVT-02S	Declaration Naming Convention	Coding Style	🟢 Informational	✓
WCR-	Potential Lock of Ether	Logical Issue	🔵 Minor	◷

01S				
WCR-02S	Declaration Naming Convention	Coding Style	● Informational	✓
WUV-01S	Declaration Naming Convention	Coding Style	● Informational	✓

PAR-01M: Inexistent Input Sanitization

Type	Severity	Location
Logical Issue	🔵 Minor	Partnership.sol L59-L67

Description:

The linked function allows for no partner name and zero address wallet, as zero fees should be intended.

Recommendation:

We advise to add `require` statements, ensuring that the a new partner will have the correct data.

Alleviation:

The development team opted to consider our references and added `require` statements, ensuring that the a new partner will have a wallet address and a name.

PAR-02M: Return Variable Utilization

Type	Severity	Location
Gas Optimization	● Informational	Partnership.sol L72, L87

Description:

The linked function declarations contain explicitly named `return` variables that are not utilized within the function's code block.

Recommendation:

We advise that the linked variables are either utilized or omitted from the declaration.

Alleviation:

The development team opted to consider our references and omit the named `return` variable from the declaration.

RMT-01M: Function Visibility Optimization

Type	Severity	Location
Gas Optimization	● Informational	RoutingManagement.sol L63

Description:

The linked function is declared as `public`, contains array function arguments and is not invoked in any of the contract's contained within the project's scope.

Recommendation:

We advise that the functions' visibility specifiers are set to `external`, as the array-based arguments have their their data location set to `calldata`, optimizing the gas cost of the function.

Alleviation:

The development team opted to consider our references and changed the linked functions' visibility to `external`, as the array-based arguments have their their data location set to `calldata`.

WBR-01M: Return Variable Utilization

Type	Severity	Location
Gas Optimization	● Informational	WardenBestRateQuery.sol L40, L86

Description:

The linked function declarations contain explicitly named `return` variables that are not utilized within the function's code block.

Recommendation:

We advise that the linked variables are either utilized or omitted from the declaration.

Alleviation:

The development team opted to consider our references and omit the named `return` variable from the declaration.

CSR-01M: Potential Underflow

Type	Severity	Location
Logical Issue	● Medium	CurveSusdRoute.sol L58

Description:

The linked statement can lead to an integer underflow.

Recommendation:

We advise to use the `SafeMath` library for the linked arithmetic operation.

Alleviation:

The development team opted to consider our references and utilized the `SafeMath` library for the linked arithmetic operation.

UVP-01M: Ambiguous `payable` Function

Type	Severity	Location
Logical Issue	● Minor	UniswapV2PoolToPoolTokenEthTokenRoute.sol L37

Description:

The linked function should not be `payable` as it does not deal with sent ETH possibly trapping the ether in contract. The function does not allow trading ETH directly as suggested by the check on L43. Also it does not ensure that the ether transferred is equal to the intended amount.

Recommendation:

We advise to revise the linked function.

Alleviation:

The Warden development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

UVT-01M: Ambiguous `payable` Function

Type	Severity	Location
Logical Issue	● Minor	UniswapV2TokenEthTokenRoute.sol L34

Description:

The linked function should not be `payable` as it does not deal with sent ETH possibly trapping the ether in contract. The function does not allow trading ETH directly as suggested by the check on L40. Also it does not ensure that the ether transferred is equal to the intended amount.

Recommendation:

We advise to revise the linked function.

Alleviation:

The Warden development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

WCR-01M: Potential Underflow

Type	Severity	Location
Logical Issue	● Medium	WardenCurveRouter.sol L46

Description:

The linked statement can lead to an integer underflow.

Recommendation:

We advise to use the `SafeMath` library for the linked arithmetic operation.

Alleviation:

The development team opted to consider our references and utilized the `SafeMath` library for the linked arithmetic operation.

WUV-01M: Redundant `array` Look Up

Type	Severity	Location
Gas Optimization	● Informational	WardenUV2Router.sol L60, L126, L149

Description:

The linked `for` loop conditionals redundantly use the `length` member of the specified `array`.

Recommendation:

We advise to assign the `array` size to a local variable instead.

Alleviation:

The development team opted to consider our references and used a local variable for the `array` size.

PAR-01S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	Partnership.sol L49

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

RMT-01S: Boolean Comparison

Type	Severity	Location
Gas Optimization	● Informational	RoutingManagement.sol L44, L49

Description:

The linked conditionals redundantly compare a boolean variable to a boolean constant.

Recommendation:

We advise to directly utilize the value of the linked variable instead.

Alleviation:

The development team opted to consider our references and directly utilized the value of the linked variable.

WSP-01S: External Calls Inside a Loop

Type	Severity	Location
Volatile Code	● Informational	WardenSwap.sol L398

Description:

The linked statements execute external calls inside a loop, which can lead to a denial-of-service attack.

Recommendation:

We advise to set an upper bound to the linked function input arrays.

Alleviation:

The Warden development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

WBR-01S: External Calls Inside a Loop

Type	Severity	Location
Volatile Code	● Informational	WardenBestRateQuery.sol L65, L111

Description:

The linked statements execute external calls inside a loop, which can lead to a denial-of-service attack.

Recommendation:

We advise to set an upper bound to the linked function input arrays.

Alleviation:

The Warden development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

WBR-02S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	WardenBestRateQuery.sol L23

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

CSR-01S: Potential Lock of Ether

Type	Severity	Location
Logical Issue	● Minor	CurveSusdRoute.sol L28-L61

Description:

The `CurveSusdRoute` contract does not contain a withdraw function to empty the leftover ether in the contract. Also, the contract does not utilize the ether that it withholds.

Recommendation:

We advise to implement a function to withdraw the leftover ether amounts.

Alleviation:

The Warden development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

CSR-02S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	CurveSusdRoute.sol L22-L26

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

SRE-01S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	SpartanRoute.sol L42-L46

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

SRT-01S: Declaration Naming Convention

Type	Severity	Location
Coding Style	🟢 Informational	SushiswapRoute.sol L13-L17

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

UVP-01S: Potential Lock of Ether

Type	Severity	Location
Logical Issue	● Minor	UniswapV2PoolToPoolTokenEthTokenRoute.sol L31-L79

Description:

The `UniswapV2PoolToPoolTokenEthTokenRoute` contract does not contain a withdraw function to empty the leftover ether in the contract.

Recommendation:

We advise to implement a function to withdraw the leftover ether amounts.

Alleviation:

The Warden development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

UVP-02S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	UniswapV2PoolToPoolTokenEthTokenRoute.sol L16, L18, L19

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

UVR-01S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	UniswapV2Route.sol L14, L16, L17

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

UVT-01S: Potential Lock of Ether

Type	Severity	Location
Logical Issue	● Minor	UniswapV2TokenEthTokenRoute.sol L28-L59

Description:

The `UniswapV2TokenEthTokenRoute` contract does not contain a withdraw function to empty the leftover ether in the contract.

Recommendation:

We advise to implement a function to withdraw the leftover ether amounts.

Alleviation:

The Warden development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

UVT-02S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	UniswapV2TokenEthTokenRoute.sol L15, L17, L18

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

WCR-01S: Potential Lock of Ether

Type	Severity	Location
Logical Issue	● Minor	WardenCurveRouter.sol L28-L49

Description:

The `WardenCurveRouter` contract does not contain a withdraw function to empty the leftover ether in the contract.

Recommendation:

We advise to implement a function to withdraw the leftover ether amounts.

Alleviation:

The Warden development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

WCR-02S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	WardenCurveRouter.sol L22-L26

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

WUV-01S: Declaration Naming Convention

Type	Severity	Location
Coding Style	● Informational	WardenUV2Router.sol L16, L18, L19

Description:

The linked declarations do not conform to the Solidity style guide with regards to its naming convention. Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

Recommendation:

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation:

The development team opted to consider our references and changed the name of the linked `constant` variable to an `UPPER_CASE` .

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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 are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.