H-X

Report on Security Assessment of Milkomeda Smart Contracts for OccamX

Version 1.0

Revision History

Contributor	Date	Comment
Bogdan Barchuk, CEH; Vladimir Buldyzhov, CISSP	15 Feb 2022	Version 1.0

Introduction

By request of OccamX (Customer, Company), and according to Purchase Order dated 2 Feb 2022, H-X Technologies (H-X, Provider or pentesters) has delivered the professional information security services, namely, security assessment of the Customer's smart contracts (target object).

After reviewing the implementation of Milkomeda-OCCAMX's smart contracts, this audit report has been prepared to discover potential issues and vulnerabilities of their source code. We have outlined our approach to evaluate the potential security risks. Advice to further improve the quality of security or performance was also given in the report.

What is a Smart Contract Audit

Smart Contracts are the crux of all Ethereum DApps and Token Sales. Smart Contracts are essentially programs designed to execute automatically and enforce a set of rules autonomously.

Smart Contracts are completely unchangeable once deployed on the blockchain — a quality that makes smart contracts uniquely reliable and trustworthy, but also dangerous objects.

Coding for the blockchain is a relatively new field, without many security standards, documentation, or best practices. It is also the ultimate test of defensive software engineering. Smart contracts can end up controlling tens of millions of dollars, making them a target for attackers.

Audit of Smart Contracts is focused on finding logic flaws and security vulnerabilities, especially, which could let an attacker to misuse the Smart Contract, to violate customer's business requirements or cause any other harm to the customer or its clients or partners. The goal of the audit is to model and verify the target object compromise, sensitive information theft, weak conditions or other ways or prerequisites for realization of fraud or security incidents. To achieve this goal, tools and techniques very similar to those that an attacker would use are typically required.

The audit was carried out externally from the auditors' premise. The best practices Solidity Style Guide and Ethereum Smart Contract Security Best Practices were used. Automated and manual techniques were used to verify and evaluate the security of the target object.

Audit Summary

According to the assessment, the Customer's Solidity smart contract is well secured.

In general, the code is well commented. Commenting provides extensive documentation of functions, return variables, and so on, and helps auditors quickly understand the flow of code logic.

According to the findings, we recommend eliminating all high-, medium- and low-level findings, additionally reviewing informative findings as well. This will help to ensure confidentiality, integrity and availability within in place.

Based on the number of findings, Medium-level vulnerabilities are likely possible to compromise confidential data and system integrity, however, this can be less serious security breaches. Low-level vulnerabilities could be a potential security threat for the system. Based on the number of informative related findings, this indicates the violation of code best practices.

Recommendations

Auditors recommend to mitigate all vulnerabilities described in this report.

The nature of information threats considers uncertainty of penetration paths that may be used by an attacker. In addition, the set of known technical vulnerabilities of the libraries, components and the hosting environment is constantly increasing. Therefore, the results of this audit cannot guarantee uncovering all possible compromise or penetration ways and security problems, and only show most weak points in the security of the target object.

Besides smart contract audits, to enhance Customer’s security effectively and to reduce Customer’s business risks, other appropriate security management processes and security solutions should be designed and implemented. These security measures include but are not limited by: secure development lifecycle, regular security audits by an independent party, security event monitoring, incident response.

Using internationally recognized standards and best practices such as ISO 27000, PCI DSS, NIST is recommended. We can help in implementation of these processes and solutions.

Methodology

To evaluate the potential vulnerabilities or issues, we go through a checklist of well-known smart-contract-related security issues, using automatic verification tools and manual review. For any discovered issue, we might test it on our private network to reproduce the issue to prove our findings. In this audit, we considered the following important features of the code.

- Compliance of the code with the requirements of the SWC registry.
- Implementation of protocol standards.
- Whether the code conforms to coding best practices.

Manual audit

- Manual code analysis for security vulnerabilities.
- Evaluation of the overall structure, complexity, and quality of the project.
- Checking SWC registry errors in the code.
- Analysis of data security in the network.
- Failover analysis to check the operation of the smart contract in case of errors and vulnerabilities.

Automated analysis

- Scanning the project's codebase with Mythx, Slither, Echidna, Manticore, and others.
- Manual check of all problems found by the tools.
- Manual security testing according to SWC-Registry.

Project Scope

The scope of the project is a smart contract and its dependencies. We have checked this smart contract for well-known and more specific vulnerabilities, for example:

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- ERC20, ERC2612 and ERC3156 API violation
- Malicious libraries
- Compiler version not fixed

- Unchecked external call
- Unchecked math

During the audit, 31 files were analyzed. The following files have been scanned using the tools, checked and tested manually.

File	keccak256
CalHash.sol	9fa458d528035823a07418ef5b821db6b13343fe36242b930b5d420eb84ce30a
Collector.sol	cd38529d83dc9ed37daa611cccbb9c45c4c1c3b257eb4029e4c1b45d7b92fcc1
ERC20Permit.sol	62b5a199c63dbc51bd50b16dde9f0b1ed0eb38049b5fc7fd525f8633b5627294
Factory.sol	cb3ceeff088373ed5c396253de64fe65d1fad4aafdf3fc013a64746eff98d6c8
Ownable.sol	90ca4f10c4c097b6f5b993f7708ca25fc41af33ce09dea64b807ef6954b76df4
Pair.sol	113cfb99beb44e94883b0d9558f4b68c9908b63ec6b37dafcc5c0f7fe47882a1
ProtocolToken.sol	b64e616c8e04f10baa9b062233e2c810ea6d798da0d9e61e1520cbb10c4d08f7
Router01.sol	6e4819b1312bad28fc964523cb938609276dc91d0f6ba1be30f4779e7f037135
Router02.sol	4b7a0c837e0c5c2dcf5bb8458be34de7158c2aa90c9f24a01469230620f90f9b
WADA10.sol	7966590645ce2e29408005919e7a0ac7d7026fb39088ee1cc37ff5c5456071d7
ICallee.sol	594ca01e838483aac649f9f0c6e45f086ed1e58808e0cd1d5e6e34464f2683bc
IERC20.sol	458d9d1f245b1c3573040db74cde5467e7214cba6089bb0b591c42f3607aa626
IERC20Burnable.sol	6f7a79d7b39728a13fd394562212cec8fe0cd14d8d1dc4bd826fdd15b9fe725e
IERC20Permit.sol	3aef2a1027c03d8a6f60404f05b57582db4fd3e7def38ee16cd1cc3f682fce16
IERC2612.sol	cc3b8ca26e390a7b95b4236065be6ed157aaa28ecd778e8e7236fdea52db5dad
IERC3156FlashBorrower.sol	7a0afdaceaade0d31c37a09e0ceef82ea430bd5c415a93b7c69795f0042b2be6
IERC3156FlashLender.sol	2078ed64867f14159d069260bdee7f695a59c853993ad1736eac4555a283401c
IFactory.sol	cbb934ca2ecd73931cdb2d4bdb8e6f43e091f716428056c4b334e1da42794563
IPair.sol	656dda086115e6385e67c545aa7335987b798e55b34b0146e3fc243f0ca615ed

IRouter01.sol	2146f69864f4f7015cd4b198b7d20f9b108bcb4347aff6d4bb874200c91c3afb
IRouter02.sol	7cec5d60090704cc64a14d1ab9e49a56b94707e75a578e1d2b159382f4e2a800
IWADA.sol	9610e8b6ce9079e6331e2869475eccfe8281db1c9cb38d096fa2659e200c5a24
IWADA10.sol	883fa61759646133f2811457c48be963d9dac1c6a4199163fe09893b86ff2338
IExchange.sol	5ab52a6c52d122f8592f89111c95a98043a018c9a5e3989200897161f0aa73f5
Library.sol	5a1a834bbebe2528313076e21a5fa5f3937ad808ed9101d1c079b9653fab0915
Math.sol	cb9d40d4c5260996c6f4e16aa21dd6d4984515e930c758d9abab022788ba8ac7
OracleLibrary.sol	edb88dbffd46b09bdb1509bc9863019d27dbc56f61090327d1653b40459ed9f6
SafeERC20.sol	2592e2d2e7f1151f14d198051e66c9d8250278b681d09ca168f34629c25fcccb
SafeMath.sol	0ae5e0d6e8821c6b5045294ecf8f22e5bc63cba9b40bd229e02bbee978ac2b69
TransferHelper.sol	1b35418d4193b4f2db7be259301960c2064212d23610f3c649b97bca614d1cce
UQ112x112.sol	f837fd7ea62928e310c07a8e00b630d810fa8ac8e964bb67066fe5fca1c0f0da

Used dependencies

- @openzeppelin/contracts/math/SafeMath.sol
- @openzeppelin/contracts/token/ERC20/ERC20Capped.sol
- @openzeppelin/contracts/token/ERC20/ERC20Burnable.sol
- @openzeppelin/contracts/access/Ownable.sol

The Severity Level of the Issues

Severity	Description
Critical	The problem will result in loss of assets or manipulation of data.
High	The problem will seriously affect the correctness of the business model.
Medium	The problem is still important to solve, but impractical to use.
Low	The problem is mainly related to outdated, unused code snippets.
Informational	This issue is mainly related to code style, informational statements, and is not required to be fixed.

Findings and Risk Levels

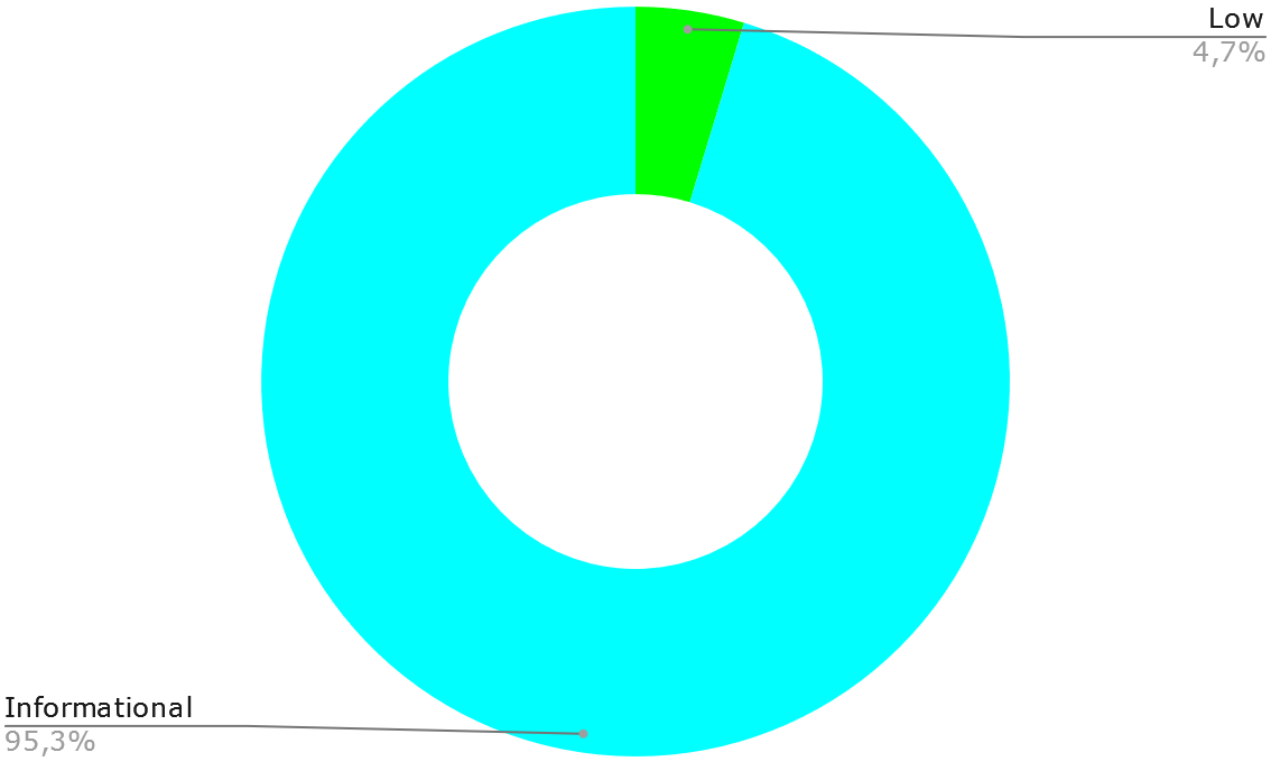
Findings	Risk level
Extra gas consumption	Low
Wrong compiler version	Low
Wrong compiler version	Low
Missing zero address validation	Informational
Compiler version not fixed	Informational
Keyword order	Informational
Missing zero address validation	Informational
Missing zero address validation	Informational
Low-Level Calls	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Too many digits	Informational

Different pragma directives are used	Informational
Dead-Code	Informational
Low-level Calls	Informational
Low-level Calls	Informational
Low-level Calls	Informational
Low-level Calls	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Variable Names too Similar	Informational
Different pragma directives are used	Informational
Dead-code	Informational
Dead-code	Informational
Dead-code	Informational
Dead-code	Informational
Assembly	Informational
Assembly	Informational
Assembly	Informational
Pragma	Informational
Low-level-calls	Informational
Low-level-calls	Informational
Low-level-calls	Informational
Low-level-calls	Informational

Low-level-calls	Informational
Assembly	Informational
Low-level-calls	Informational
Assembly	Informational
Dead-code	Informational
Dead-code	Informational
Dead-code	Informational
Low-level calls	Informational
Low-level calls	Informational
Low-level calls	Informational
Low-level calls	Informational
Low-level calls	Informational
Different pragma directives are used	Informational
Dead-code	Informational
Low-level calls	Informational
Low-level calls	Informational
Low-level calls	Informational
Low-level calls	Informational
Variable names too similar	Informational
Variable names too similar	Informational
Variable names too similar	Informational
Variable names too similar	Informational
Variable names too similar	Informational
Variable names too similar	Informational
Variable names too similar	Informational
Variable names too similar	Informational
Variable names too similar	Informational

Diagram of the Findings

Critical	High	Medium	Low	Informational
0	0	0	3	61



Visualization of Smart Contract Logics

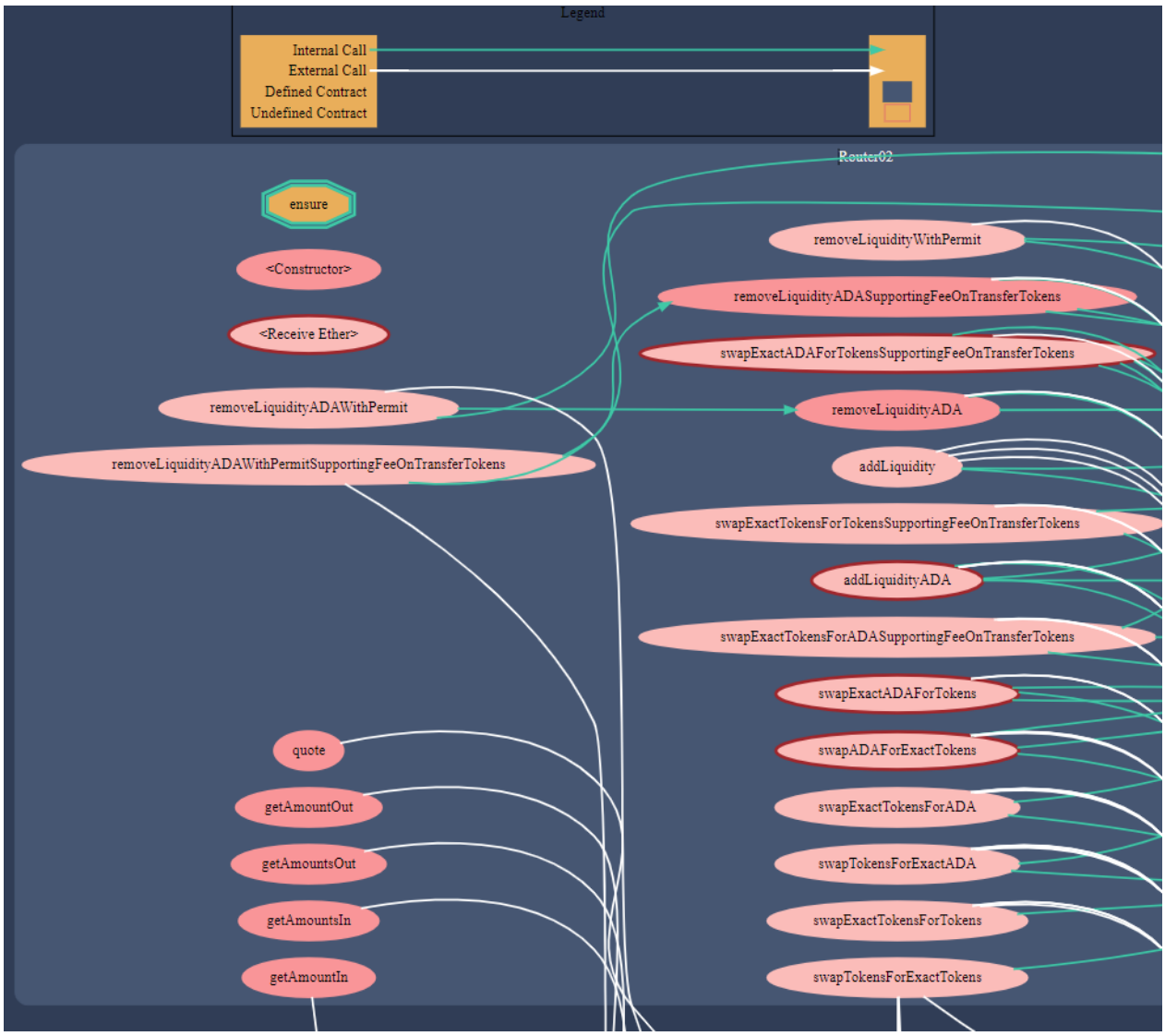


Figure 1. The graph of the WADA contract structure.

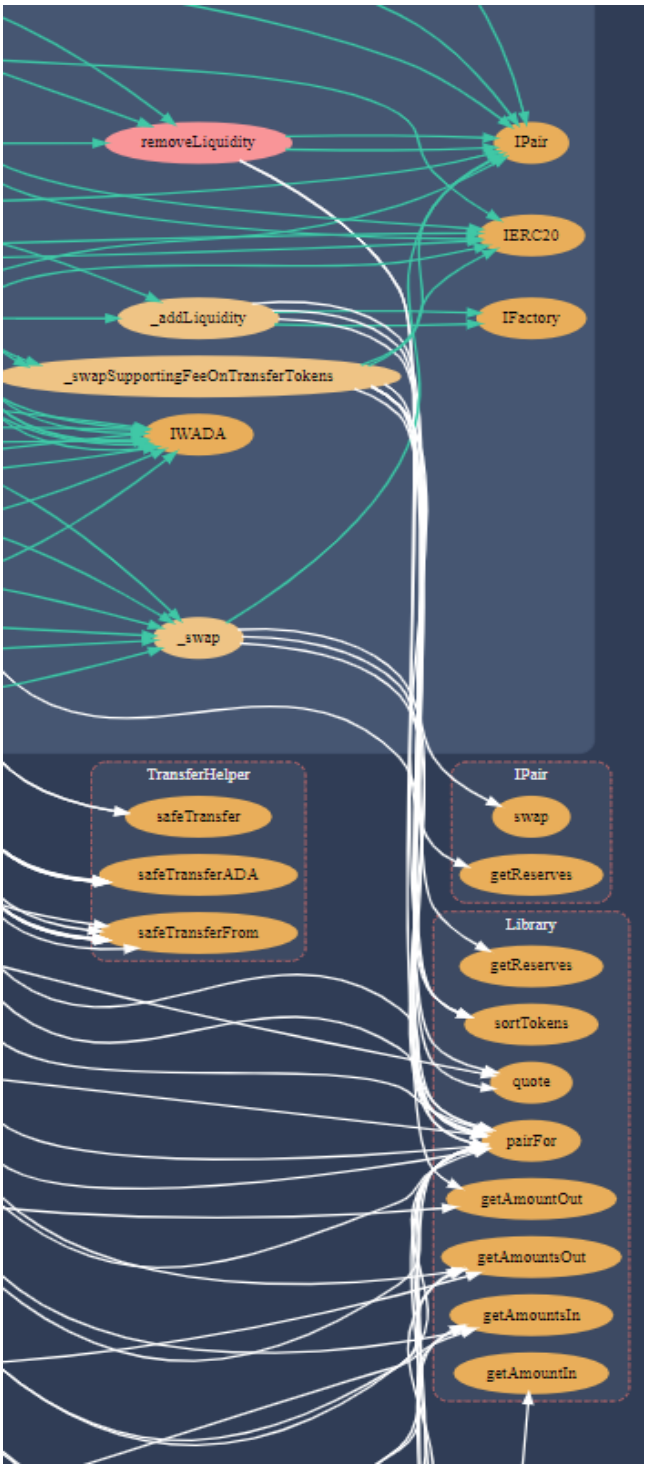


Figure 2. The graph of the Router02 contract structure.

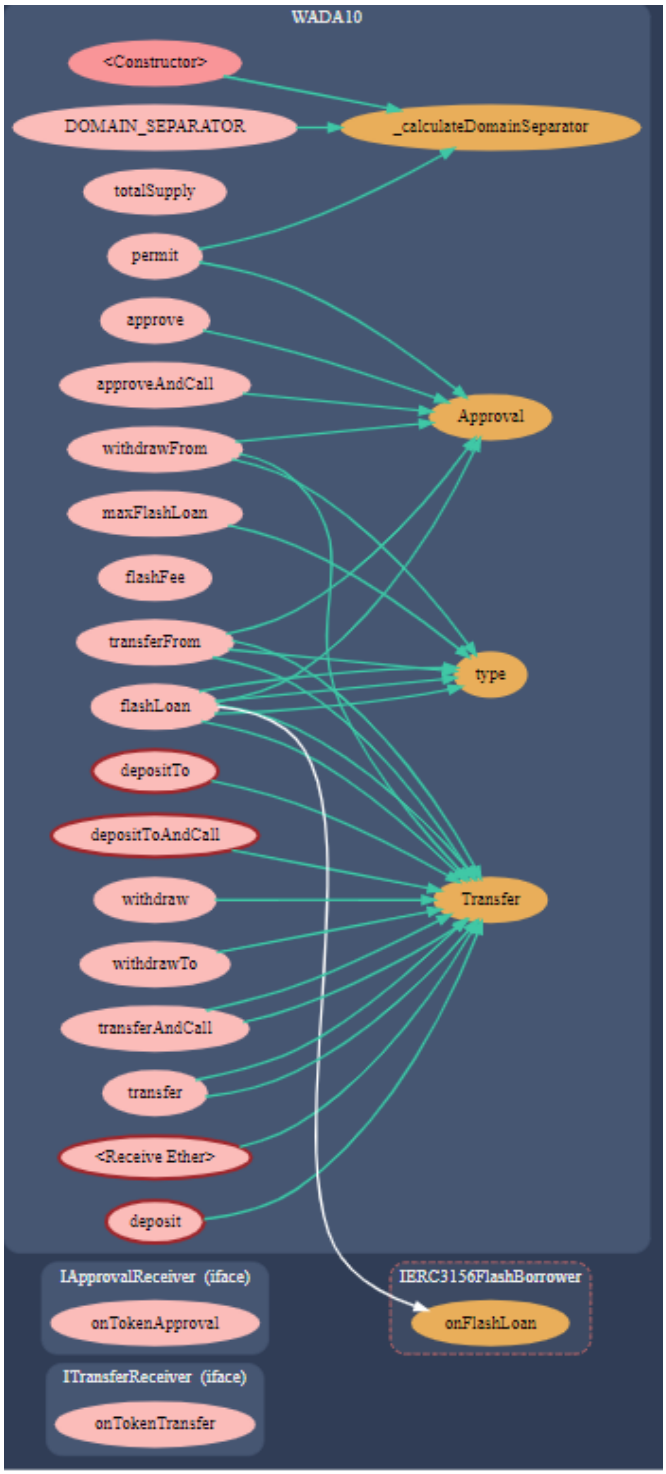


Figure 3. The graph of the Router02 contract structure.

Findings Overview

SWC ID	Bug class	Pass
SWC-100	Function Default Visibility	OK
SWC-101	Integer Overflow and Underflow	OK
SWC-102	Outdated Compiler Version	FAIL
SWC-103	Floating Pragma	FAIL
SWC-104	Unchecked Call Return Value	OK
SWC-105	Unprotected Ether Withdrawal	OK
SWC-106	Unprotected SELFDESTRUCT Instruction	OK
SWC-107	Reentrancy	OK
SWC-108	State Variable Default Visibility	OK
SWC-109	Uninitialized Storage Pointer	OK
SWC-110	Assert Violation	OK
SWC-111	Use of Deprecated Solidity Functions	OK
SWC-112	Delegatecall to Untrusted Callee	OK
SWC-113	DoS with Failed Call	OK
SWC-114	Transaction Order Dependence	OK
SWC-115	Authorization through tx.origin	OK
SWC-116	Timestamp Dependence	OK
SWC-118	Incorrect Constructor Name	OK
SWC-119	Shadowing State Variables	OK
SWC-120	Weak Sources of Randomness	OK
SWC-123	Requirement Violation	OK
SWC-124	Write to Arbitrary Storage Location	OK
SWC-127	Arbitrary Jump	OK
SWC-128	Gas Exhaustion	OK
SWC-129	Typographical Error	OK
SWC-130	Right-To-Left-Override control character	OK
SWC-131	Presence of unused variables	OK
SWC-132	Unexpected Ether balance	OK
SWC-133	Hash Collisions With Multiple Variable Length Arguments	OK
SWC-134	Message call with hardcoded gas amount	OK
SWC-135	Code With No Effects	OK
SWC-136	Unencrypted Private Data On-Chain	OK

Results from Manual analysis

Extra gas consumption

Description:

Both **immutable** and **constant** are keywords that can be used on state variables to restrict modifications to their state. The difference is that **constant** variables can never be changed after compilation, while **immutable** variables can be set within the constructor. State variables can be declared as **constant** or **immutable**. In both cases, the *CALLBACK_SUCCESS* and *PERMIT_TYPEHASH* variables cannot be modified after the contract has been constructed. For **constant** variables, the value has to be fixed at compile-time, the compiler does not reserve a storage slot for these variables, and every occurrence is replaced by the respective value, while for **immutable**, it can still be assigned at construction time. This makes using a **constant** variable cheaper than using an **immutable** variable.

Risk: **Low**

Location:

- ./contracts/WADA10.sol ***#22-23***

Code section:

```
pragma solidity ^0.6.0;

...

contract WADA10 is IWADA10 {

  ...

      bytes32 public immutable CALLBACK_SUCCESS = keccak256("ERC3156FlashBorrower.onFlashLoan");

      bytes32 public immutable PERMIT_TYPEHASH = keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)");

  ...

}
```

Recommendation on Improvement:

Change the keyword for declaring a state variable from **immutable** to **constant**.

```
contract WADA10 is IWADA10 {
  ...
    bytes32 public constant CALLBACK_SUCCESS = keccak256("ERC3156FlashBorrower.onFlashLoan");
    bytes32 public constant PERMIT_TYPEHASH = keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)");
  ...
}
```

Wrong compiler version

Description:

The specified compiler versions contain invalid compilers. The WADA10 smart contract uses a language feature introduced in version 0.6.5. State variables can be marked **immutable** which causes them to be read-only, but assignable in the constructor. The value will be stored directly in the code.

However, the specified version allows the use of lower versions of the compiler, which results in an error.

https://github.com/ethereum/solidity/releases/tag/v0.6.5

Risk: Low

Location:

- ./contracts/WADA10.sol ***#22-25***

Code section:

```
pragma solidity ^0.6.0;
...
contract WADA10 is IWADA10 {
  ...
```

```
        bytes32 public immutable CALLBACK_SUCCESS = keccak256("ERC3156FlashBorrower.onFlashLoan");

        bytes32 public immutable PERMIT_TYPEHASH = keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)");

        uint256 public immutable deploymentChainId;

        bytes32 private immutable _DOMAIN_SEPARATOR;

    ...
}
```

Recommendation on Improvement:

Change compiler version to 0.6.12.

```
pragma solidity 0.6.12;
```

Wrong compiler version

Description:

The specified compiler versions contain invalid compilers. The WADA10 smart contract uses a language feature introduced in version 0.6.8. Implemented **type(T).min** and **type(T).max** for every integer type T that returns the smallest and largest value representable by the type.

However, the specified version allows the use of lower versions of the compiler, which results in an error.

https://github.com/ethereum/solidity/releases/tag/v0.6.8

Risk: Low

Location:

- ./contracts/WADA10.sol ***#110***
- ./contracts/WADA10.sol ***#132***
- ./contracts/WADA10.sol ***#134***
- ./contracts/WADA10.sol ***#147***

- ./contracts/WADA10.sol ***#207***
- ./contracts/WADA10.sol ***#328***

Code section:

```
pragma solidity ^0.6.0;
...
contract WADA10 is IWADA10 {
  ...
        return token == address(this) ? type(uint112).max - flashMinted : 0; // Can't underflow
  ...
        require(value <= type(uint112).max, "WADA: individual loan limit exceeded");
        flashMinted = flashMinted + value;
        require(flashMinted <= type(uint112).max, "WADA: total loan limit exceeded");
  ...
        if (allowed != type(uint256).max) {
  ...
        if (allowed != type(uint256).max) {
  ...
   if (allowed != type(uint256).max) {
  ...
}
```

Recommendation on Improvement:

Change compiler version to 0.6.12.

Compiler version not fixed

Description:

Future compiler versions may handle certain language constructions in a way the developer did not foresee.

Risk: Informational

Location:

- CalHash.sol
- Collector.sol
- ERC20Permit.sol
- Factory.sol
- Ownable.sol
- Pair.sol
- ProtocolToken.sol
- Router01.sol
- Router02.sol
- WADA10.sol
- interfaces\ICallee.sol
- interfaces\IERC20.sol
- interfaces\IERC20Burnable.sol
- interfaces\IERC20Permit.sol
- interfaces\IERC2612.sol
- interfaces\IERC3156FlashBorrower.sol
- interfaces\IERC3156FlashLender.sol
- interfaces\IFactory.sol
- interfaces\IPair.sol
- interfaces\IRouter01.sol
- interfaces\IRouter02.sol
- interfaces\IWADA.sol
- interfaces\IWADA10.sol
- interfaces\V1\IExchange.sol
- libraries\Library.sol
- libraries\Math.sol

- libraries\OracleLibrary.sol
- libraries\SafeERC20.sol
- libraries\SafeMath.sol
- libraries\TransferHelper.sol
- libraries\UQ112x112.sol

Code section:

```
pragma solidity ^0.6.0;
pragma solidity >=0.6.0;
pragma solidity ^0.7.0;
pragma solidity >=0.6.0 <=0.8.0;
```

Recommendation on Improvement:

Specify the exact compiler version (ex: pragma solidity x.y.z;).

```
pragma solidity 0.6.12;
pragma solidity 0.6.12;
pragma solidity 0.7.6;
pragma solidity 0.7.6;
```

Keyword order

Description:

The wrong order can be confusing for developers and code reviewers.

Risk: Informational

Location:

- ./contracts/WADA10.sol ***#18-20***

Code section:

```
contract WADA10 is IWADA10 {
        string public override constant name = "Wrapped ADA v10";
        string public override constant symbol = "WADA10";
        uint8  public override constant decimals = 18;
   ...
}
```

Recommendation on Improvement:

Swap override and constant keywords.

```
contract WADA10 is IWADA10 {
        string public constant override name = "Wrapped ADA v10";
        string public constant override symbol = "WADA10";
        uint8  public constant override decimals = 18;
   ...
}
```

Missing zero address validation

Description:

Missing zero address validation.

Risk: Informational

Location:

- ./contracts/Router01.sol ***#21-22***

Code section:

```
contract Router01 is IRouter01 {
  ...
  constructor(address _factory, address _WADA) public {
        factory = _factory;
        WADA = _WADA;
  }
  ...
}
```

Recommendation on Improvement:

It is recommended to validate parameters **_factor**, and **_WADA** to be a non-zero value before the assignment.

Missing zero address validation

Description:

Missing zero address validation.

Risk: Informational

Location:

- ./contracts/Router02.sol ***#24-25***

Code section:

```
contract Router01 is IRouter02 {
  ...
  constructor(address _factory, address _WADA) public {
        factory = _factory;
        WADA = _WADA;
  }
  ...
}
```

Recommendation on Improvement:

It is recommended to validate parameters **_factor**, and **_WADA** to be a non-zero value before the assignment.

Results from Semi-Automatic Scans

We also used a self-developed Solidity static analysis framework which runs a suite of vulnerability detectors, shows visual information about contract details, and provides an API to write custom analyses quickly. Slither enables developers to find vulnerabilities, enhance their code comprehension, and promptly prototype custom analyses. Each solidity file and project together has been analyzed. We got a report with a few warnings and errors.

Missing zero address validation

Description:

Missing zero address validation.

Impact: **Medium**

Confidence: Medium

Location:

- ./contracts/WADA10.sol

Details:

Reentrancy in WADA10.flashLoan(IERC3156FlashBorrower,address,uint256,bytes) (contracts/WADA10.sol#130-162):

External calls:

- require(bool,string)(receiver.onFlashLoan(msg.sender,address(this),value,0,data) == CALLBACK_SUCCESS,WADA: flash loan failed) (contracts/WADA10.sol#140-143)

State variables written after the call(s):

- balanceOf[address(receiver)] = balance - value (contracts/WADA10.sol#157)

- flashMinted = flashMinted - value (contracts/WADA10.sol#160)

Low-Level Calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/Pair.sol

Details:

Low level call in Pair._safeTransfer(address,address,uint256) (contracts/Pair.sol#44-47):

- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/Pair.sol#45)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Pair.sol

Details:

Variable Pair.swap(uint256,uint256,address,bytes).balance0Adjusted (contracts/Pair.sol#180) is too similar to Pair.swap(uint256,uint256,address,bytes).balance1Adjusted (contracts/Pair.sol#181)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Pair.sol

Details:

Variable Pair.price0CumulativeLast (contracts/Pair.sol#26) is too similar to Pair.price1CumulativeLast (contracts/Pair.sol#27)

Too many digits

Description:

Literals with many digits are difficult to read and review.

Impact: Informational

Confidence: Medium

Location:

- contracts/CalHash.sol

Details:

CalHash.getInitHash() (contracts/CalHash.sol#5-8) uses literals with too many digits:

- bytecode = type()(Pair).creationCode (contracts/CalHash.sol#6)

Different pragma directives are used

Description:

Detects whether different Solidity versions are used.

Impact: Informational

Confidence: High

Location:

- contracts/Router01.sol

Details:

Different versions of Solidity is used:

- Version used: ['>=0.6.0', '^0.6.0']
- ^0.6.0 (contracts/Router01.sol#1)
- >=0.6.0 (contracts/interfaces/IERC20.sol#1)
- ^0.6.0 (contracts/interfaces/IFactory.sol#1)
- ^0.6.0 (contracts/interfaces/IPair.sol#1)
- ^0.6.0 (contracts/interfaces/IRouter01.sol#1)
- ^0.6.0 (contracts/interfaces/IWADA.sol#1)
- ^0.6.0 (contracts/libraries/Library.sol#1)
- ^0.6.0 (contracts/libraries/SafeMath.sol#1)
- >=0.6.0 (contracts/libraries/TransferHelper.sol#1)

Dead-Code

Description:

Functions that are not used.

Impact: Informational

Confidence: Medium

Location:

- contracts/libraries/TransferHelper.sol

Details:

TransferHelper.safeApprove(address,address,uint256) (contracts/libraries/TransferHelper.sol#5-9) is never used and should be removed

Low-level Calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/TransferHelper.sol

Details:

Low level call in TransferHelper.safeApprove(address,address,uint256) (contracts/libraries/TransferHelper.sol#5-9):

- (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (contracts/libraries/TransferHelper.sol#7)

Low-level Calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/TransferHelper.sol

Details:

Low level call in TransferHelper.safeTransfer(address,address,uint256) (contracts/libraries/TransferHelper.sol#11-15):

- (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (contracts/libraries/TransferHelper.sol#13)

Low-level Calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/TransferHelper.sol

Details:

Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (contracts/libraries/TransferHelper.sol#17-21):

- (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (contracts/libraries/TransferHelper.sol#19)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Low-level Calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/TransferHelper.sol

Details:

Low level call in TransferHelper.safeTransferADA(address,uint256) (contracts/libraries/TransferHelper.sol#23-26):

- (success) = to.call{value: value}(new bytes(0)) (contracts/libraries/TransferHelper.sol#24)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router01.sol

Details:

Variable Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/Router01.sol#33) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/IRouter01.sol#11)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/interfaces/IRouter01.sol

Details:

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/interfaces/IRouter01.sol#10) is too similar to Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Router01.sol#62)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router01.sol

Details:

Variable Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Router01.sol#61) is too similar to Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Router01.sol#62)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router01.sol

Details:

Variable Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/Router01.sol#33) is too similar to Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/Router01.sol#34)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router01.sol

Details:

Variable Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/Router01.sol#33) is too similar to Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Router01.sol#62)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/interfaces/IRouter01.sol

Details:

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/interfaces/IRouter01.sol#10) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/IRouter01.sol#11)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router01.sol

Details:

Variable Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Router01.sol#61) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/IRouter01.sol#11)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router01.sol

Details:

Variable Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Router01.sol#61) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/IRouter01.sol#11)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router01.sol

Details:

Variable Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Router01.sol#61) is too similar to Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/Router01.sol#34)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/interfaces/IRouter01.sol

Details:

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/interfaces/IRouter01.sol#10) is too similar to Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/Router01.sol#34)

Variable Names too Similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router01.sol

Details:

Variable Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (contracts/Router01.sol#51) is too similar to Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (contracts/Router01.sol#46)

Different pragma directives are used

Description:

Detect whether different Solidity versions are used.

Impact: Informational

Confidence: High

Location:

- contracts/Collector.sol

Details:

Different versions of Solidity is used:

- Version used: ['>=0.6.0', '^0.6.0']
- ^0.6.0 (contracts/Collector.sol#4)
- ^0.6.0 (contracts/Ownable.sol#5)
- >=0.6.0 (contracts/interfaces/IERC20.sol#1)
- >=0.6.0 (contracts/interfaces/IERC20Burnable.sol#1)
- >=0.6.0 (contracts/interfaces/IERC20Permit.sol#1)
- ^0.6.0 (contracts/interfaces/IFactory.sol#1)
- ^0.6.0 (contracts/interfaces/IPair.sol#1)
- ^0.6.0 (contracts/libraries/SafeERC20.sol#2)
- ^0.6.0 (contracts/libraries/SafeMath.sol#1)

Dead-code

Description:

Functions that are not used.

Impact: Informational

Confidence: Medium

Location:

- contracts/libraries/SafeERC20.sol

Details:

SafeERC20.safeName(IERC20) (contracts/libraries/SafeERC20.sol#12-15) is never used and should be removed

Dead-code

Description:

Functions that are not used.

Impact: Informational

Confidence: Medium

Location:

- contracts/libraries/SafeERC20.sol

Details:

SafeERC20.safeSymbol(IERC20) (contracts/libraries/SafeERC20.sol#7-10) is never used and should be removed

Dead-code

Description:

Functions that are not used.

Impact: Informational

Confidence: Medium

Location:

- contracts/libraries/SafeERC20.sol

Details:

SafeERC20.safeTransferFrom(IERC20,address,uint256) (contracts/libraries/SafeERC20.sol#27-30) is never used and should be removed

Dead-code

Description:

Functions that are not used.

Impact: Informational

Confidence: Medium

Location:

- contracts/libraries/SafeMath.sol

Details:

SafeMath.sub(uint256,uint256) (contracts/libraries/SafeMath.sol#10-12) is never used and should be removed

Assembly

Description:

Functions declared as constant/pure/view using assembly code.

constant/pure/view was not enforced prior to Solidity 0.5. Starting from Solidity 0.5, a call to a constant/pure/view function uses the STATICCALL opcode, which reverts in case of state modification.

Impact: Informational

Confidence: High

Location:

contracts/WADA10.sol

WADA10.constructor() (contracts/WADA10.sol#40-45) uses assembly

- INLINE ASM (contracts/WADA10.sol#42)

Assembly

Description:

Functions declared as constant/pure/view using assembly code.

constant/pure/view was not enforced prior to Solidity 0.5. Starting from Solidity 0.5, a call to a constant/pure/view function uses the STATICCALL opcode, which reverts in case of state modification.

Impact: Informational

Confidence: High

Location:

- contracts/WADA10.sol

WADA10.DOMAIN_SEPARATOR() (contracts/WADA10.sol#61-65) uses assembly

- INLINE ASM (contracts/WADA10.sol#63)

Assembly

Description:

Functions declared as constant/pure/view using assembly code.

constant/pure/view was not enforced prior to Solidity 0.5. Starting from Solidity 0.5, a call to a constant/pure/view function uses the STATICCALL opcode, which reverts in case of state modification.

Impact: Informational

Confidence: High

Location:

- contracts/WADA10.sol

WADA10.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/WADA10.sol#257-284) uses assembly

- INLINE ASM (contracts/WADA10.sol#261)

Pragma

Description:

Detects out-of-range enum conversion (solc < 0.4.5).

Impact: Informational

Confidence: High

Location:

- contracts/WADA10.sol

Different versions of Solidity is used:

- Version used: ['>=0.6.0', '>=0.6.0<=0.8.0', '^0.6.0']
- ^0.6.0 (contracts/WADA10.sol#3)
- >=0.6.0 (contracts/interfaces/IERC20.sol#1)
- ^0.6.0 (contracts/interfaces/IERC2612.sol#3)

- >=0.6.0<=0.8.0 (contracts/interfaces/IERC3156FlashBorrower.sol#2)
- >=0.6.0<=0.8.0 (contracts/interfaces/IERC3156FlashLender.sol#2)
- ^0.6.0 (contracts/interfaces/IWADA10.sol#4)

Low-level-calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/WADA10.sol

Low level call in WADA10.withdraw(uint256) (contracts/WADA10.sol#168-178):

- (success) = msg.sender.call{value: value}() (contracts/WADA10.sol#176)

Low-level-calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

contracts/WADA10.sol

Low level call in WADA10.withdrawTo(address,uint256) (contracts/WADA10.sol#184-194):

- (success) = to.call{value: value}() (contracts/WADA10.sol#192)

Low-level-calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

-contracts/WADA10.sol

Low level call in WADA10.withdrawFrom(address,address,uint256) (contracts/WADA10.sol#203-224):

- (success) = to.call{value: value}() (contracts/WADA10.sol#222)

Low-level-calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

contracts/WADA10.sol

Low level call in WADA10.transfer(address,uint256) (contracts/WADA10.sol#292-312):

- (success) = msg.sender.call{value: value}() (contracts/WADA10.sol#307)

Low-level-calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

contracts/WADA10.sol

Low level call in WADA10.transferFrom(address,address,uint256) (contracts/WADA10.sol#324-355):

- (success) = msg.sender.call{value: value}() (contracts/WADA10.sol#350)

Assembly

Description:

Functions declared as constant/pure/view using assembly code.

constant/pure/view was not enforced prior to Solidity 0.5. Starting from Solidity 0.5, a call to a constant/pure/view function uses the STATICCALL opcode, which reverts in case of state modification.

Impact: Informational

Confidence: High

Location:

contracts/ERC20Permit.sol

ERC20Permit.constructor() (contracts/ERC20Permit.sol#24-38) uses assembly

- INLINE ASM (contracts/ERC20Permit.sol#26-28)

Low-level-calls

Description:

Impact: Informational

Confidence: High

Location:

- contracts/WADA10.sol

Low level call in WADA10.transferAndCall(address,uint256,bytes) (contracts/WADA10.sol#364-384):

- (success) = msg.sender.call{value: value}() (contracts/WADA10.sol#379)

Assembly

Description:

Impact: Informational

Confidence: High

Location:

contracts/ERC20Permit.sol

ERC20Permit.constructor() (contracts/ERC20Permit.sol#24-38) uses assembly

- INLINE ASM (contracts/ERC20Permit.sol#26-28)

Dead-code

Description:

Functions that are not used.

Exploit Scenario:

```
contract Contract{
  function dead_code() internal() {}
```

```
}
```

Impact: Informational

Confidence: Medium

Location:

- contracts/ERC20Permit.sol

ERC20Permit._burn(address,uint256) (contracts/ERC20Permit.sol#46-50) is never used and should be removed

Dead-code

Description:

Functions that are not used.

Exploit Scenario:

```
contract Contract{
  function dead_code() internal() {}
}
```

Impact: Informational

Confidence: Medium

Location:

contracts/ERC20Permit.sol

ERC20Permit._mint(address,uint256) (contracts/ERC20Permit.sol#40-44) is never used and should be removed

Dead-code

Description:

Functions that are not used.

Exploit Scenario:

```
contract Contract{
   function dead_code() internal() {}
}
```

Impact: Informational

Confidence: Medium

Location:

contracts/libraries/SafeMath.sol

SafeMath.mul(uint256,uint256) (contracts/libraries/SafeMath.sol#14-16) is never used and should be removed.

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/SafeERC20.sol

Details:

Low level call in SafeERC20.safeSymbol(IERC20) (contracts/libraries/SafeERC20.sol#7-10):

- (success,data) = address(token).staticcall(abi.encodeWithSelector(0x95d89b41)) (contracts/libraries/SafeERC20.sol#8)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/SafeERC20.sol

Details:

Low level call in SafeERC20.safeName(IERC20) (contracts/libraries/SafeERC20.sol#12-15):

- (success,data) = address(token).staticcall(abi.encodeWithSelector(0x06fdde03)) (contracts/libraries/SafeERC20.sol#13)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/SafeERC20.sol

Details:

Low level call in SafeERC20.safeDecimals(IERC20) (contracts/libraries/SafeERC20.sol#17-20):

- (success,data) = address(token).staticcall(abi.encodeWithSelector(0x313ce567)) (contracts/libraries/SafeERC20.sol#18)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/SafeERC20.sol

Details:

Low level call in SafeERC20.safeTransfer(IERC20,address,uint256) (contracts/libraries/SafeERC20.sol#22-25):

- (success,data) = address(token).call(abi.encodeWithSelector(0xa9059cbb,to,amount)) (contracts/libraries/SafeERC20.sol#23)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/SafeERC20.sol

Details:

Low level call in SafeERC20.safeTransferFrom(IERC20,address,uint256) (contracts/libraries/SafeERC20.sol#27-30):

- (success,data) = address(token).call(abi.encodeWithSelector(0x23b872dd,from,address(this),amount)) (contracts/libraries/SafeERC20.sol#28)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Different pragma directives are used

Description:

Detect whether different Solidity versions are used.

Impact: Informational

Confidence: High

Location:

- contracts/Router02.sol

Details:

Different versions of Solidity is used:

- Version used: ['>=0.6.0', '^0.6.0']
- ^0.6.0 (contracts/Router02.sol#1)
- >=0.6.0 (contracts/interfaces/IERC20.sol#1)
- ^0.6.0 (contracts/interfaces/IFactory.sol#1)
- ^0.6.0 (contracts/interfaces/IPair.sol#1)
- ^0.6.0 (contracts/interfaces/IRouter01.sol#1)
- ^0.6.0 (contracts/interfaces/IRouter02.sol#1)
- ^0.6.0 (contracts/interfaces/IWADA.sol#1)
- ^0.6.0 (contracts/libraries/Library.sol#1)
- ^0.6.0 (contracts/libraries/SafeMath.sol#1)
- >=0.6.0 (contracts/libraries/TransferHelper.sol#1)

Recommendations: Use one Solidity version.

Dead-code

Description:

Functions that are not used.

```
Exploit Scenario:
contract Contract{
   function dead_code() internal() {}
}
```

Impact: Informational

Confidence: Medium

Location:

- contracts/libraries/TransferHelper.sol

Details:

TransferHelper.safeApprove(address,address,uint256) (contracts/libraries/TransferHelper.sol#5-9) is never used and should be removed

Recommendations: Remove unused functions.

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/TransferHelper.sol

Details:

Low level call in TransferHelper.safeApprove(address,address,uint256) (contracts/libraries/TransferHelper.sol#5-9):

- (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (contracts/libraries/TransferHelper.sol#7)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/TransferHelper.sol

Details:

Low level call in TransferHelper.safeTransfer(address,address,uint256) (contracts/libraries/TransferHelper.sol#11-15):

- (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (contracts/libraries/TransferHelper.sol#13)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/TransferHelper.sol

Details:

Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (contracts/libraries/TransferHelper.sol#17-21):

- (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (contracts/libraries/TransferHelper.sol#19)

Low-level calls

Description:

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Impact: Informational

Confidence: High

Location:

- contracts/libraries/TransferHelper.sol

Details:

Low level call in TransferHelper.safeTransferADA(address,uint256) (contracts/libraries/TransferHelper.sol#23-26):

- (success) = to.call{value: value}(new bytes(0)) (contracts/libraries/TransferHelper.sol#24)

Recommendations: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Variable names too similar

Description:

Detect variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/interfaces/IRouter01.sol

Details:

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/interfaces/IRouter01.sol#10) is too similar to

Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/Router02.sol#37)

Recommendations: Prevent variables from having similar names.

Variable names too similar

Description:

Detect variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router02.sol

Details:

Variable Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Router02.sol#65) is too similar to Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Router02.sol#66)

Recommendations: Prevent variables from having similar names.

Variable names too similar

Description:

Detect variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router02.sol

Details:

Variable Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/Router02.sol#36) is too similar to Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Router02.sol#66)

Recommendations: Prevent variables from having similar names.

Variable names too similar

Description:

Detects variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router02.sol

Details:

Variable Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/Router02.sol#36) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/IRouter01.sol#11)

Variable names too similar

Description:

Detect variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router02.sol

Details:

Variable Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/Router02.sol#36) is too similar to

Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/Router02.sol#37)

Recommendations: Prevent variables from having similar names.

Variable names too similar

Description:

Detect variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/interfaces/IRouter01.sol

Details:

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/interfaces/IRouter01.sol#10) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/IRouter01.sol#11)

Recommendations: Prevent variables from having similar names.

Variable names too similar

Description:

Detect variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/interfaces/IRouter01.sol

Details:

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/interfaces/IRouter01.sol#10) is too similar to Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Router02.sol#66)

Recommendations: Prevent variables from having similar names.

Variable names too similar

Description:

Detect variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router02.sol

Details:

Variable Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Router02.sol#65) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/IRouter01.sol#11)

Recommendations: Prevent variables from having similar names.

Variable names too similar

Description:

Detect variables with names that are too similar.

Impact: Informational

Confidence: Medium

Location:

- contracts/Router02.sol

Details:

Variable Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Router02.sol#65) is too similar to Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/Router02.sol#37)

Recommendations: Prevent variables from having similar names.

Some additional Popular Automatic static security analysis tools.

Some third-party tools used to do analysis did not give any real results and were ignored.

contracts/WADA10.sol

Severity Low (1) Medium (0) High (0)

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	WADA10.sol	L: 3 C: 0

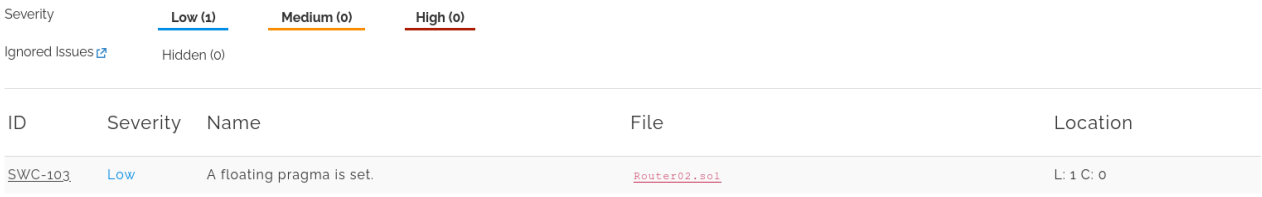
contracts/Router01.sol

Severity Low (2) Medium (0) High (0)

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	Router01.sol	L: 1 C: 0
SWC-110	Low	An assertion violation was triggered.	Router01.sol	L: 26 C: 8

contracts/Router02.sol



Conclusion

Auditors conducted the smart contract security audit. Multiple vulnerabilities were found.

The specific goal of the security audit was to identify if an attacker could compromise Customer’s smart contract protection.

The goal of the audit was met. It was determined that it was not possible to fully misuse the smart contract or to violate the Customer's business requirements directly, but the customer should pay attention to the found vulnerabilities and mitigate them.