RUGDOG

SMART CONTRACT SECURITY AUDIT

Final report

Plan: Simple

InstaDEX Finance

January 2024

rugdog.net

the@rugdog.net

✧ CONTENTS

✧ INTRODUCTION

The report has been prepared for InstaDEX Finance.

InstaDEX Finance is a decentralized exchange (DEX). The audit checked vulnerabilities of InstaDEX token.

Name	InstaDEX Finance
Audit date	2024-01-12 - 2024-01-12
Language	Solidity
Network	Binance Smart Chain

✧ CONTRACTS CHECKED

Name	Address
InstaDEX	0xD5a9D3396Da7472551561F0E872E677cA2227a6B

✧ AUDIT PROCESS

The code was audited by the team according to the following order:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual confirmation of all the issues found by the tools

Manual audit

- Thorough manual analysis of smart contracts for security vulnerabilities
- Smart contracts' logic check

ATTACKS CHECKED

Title	Check result
Unencrypted Private Data On-Chain	✓ passed
Code With No Effects	✓ passed
Message call with hardcoded gas amount	✓ passed
Typographical Error	✓ passed
DoS With Block Gas Limit	✓ passed
Presence of unused variables	✓ passed
Incorrect Inheritance Order	✓ passed
Requirement Violation	✓ passed
Weak Sources of Randomness from Chain Attributes	✓ passed
Shadowing State Variables	✓ passed
Incorrect Constructor Name	✓ passed
Block values as a proxy for time	✓ passed
Authorization through tx.origin	✓ passed

DoS with Failed Call	✓ passed
Delegatecall to Untrusted Callee	✓ passed
Use of Deprecated Solidity Functions	✓ passed
Assert Violation	✓ passed
State Variable Default Visibility	✓ passed
Reentrancy	✓ passed
Unprotected SELFDESTRUCT Instruction	✓ passed
Unprotected Ether Withdrawal	✓ passed
Unchecked Call Return Value	✓ passed
Floating Pragma	✓ passed
Outdated Compiler Version	✓ passed
Integer Overflow and Underflow	✓ passed
Function Default Visibility	✓ passed

✧ OVERVIEW OF RELEVANCE LEVELS

High relevance Issues of high relevance may lead to losses of users' funds as well as changes of ownership of a contract or possible issues with the logic of the contract.
High-relevance issues require immediate attention and a response from the team.

Medium relevance While issues of medium relevance don't pose as high a risk as the high-relevance ones do, they can be just as easily exploited by the team or a malicious user, causing a contract failure and damaging the project's reputation in the process. Usually, these issues can be fixed if the contract is redeployed.
Medium-relevance issues require a response from the team.

Low relevance Issues of low relevance don't pose high risks since they can't cause damage to the functionality of the contract. However, it's still recommended to consider fixing them.

ISSUES

High relevance issues

No high relevance issues found

Medium relevance issues

No medium relevance issues found

Low relevance issues

No low relevance issues found

✧ CONCLUSION

InstaDEX Finance InstaDEX contract was audited. No relevance issues were found.

✧ DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without RugDog prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts RugDog to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.
The rights to publish the results of this audit are exclusively retained by RugDog.

✧ AUTOMATED ANALYSIS

```
INFO:Detectors:
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) performs a multiplication on the result of a division:
        - denominator = denominator / twos (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#169)
        - inverse = (3 * denominator) ^ 2 (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#184)
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) performs a multiplication on the result of a division:
        - denominator = denominator / twos (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#169)
        - inverse *= 2 - denominator * inverse (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#188)
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) performs a multiplication on the result of a division:
        - denominator = denominator / twos (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#169)
        - inverse *= 2 - denominator * inverse (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#189)
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) performs a multiplication on the result of a division:
        - denominator = denominator / twos (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#169)
        - inverse *= 2 - denominator * inverse (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#190)
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) performs a multiplication on the result of a division:
        - denominator = denominator / twos (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#169)
        - inverse *= 2 - denominator * inverse (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#191)
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) performs a multiplication on the result of a division:
```

```
        - denominator = denominator / twos (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#169)
        - inverse *= 2 - denominator * inverse (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#192)
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) performs a multiplication on the result of a division:
        - denominator = denominator / twos (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#169)
        - inverse *= 2 - denominator * inverse (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#193)
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) performs a multiplication on the result of a division:
        - prod0 = prod0 / twos (contracts/InstaDEX/@openzeppelin/contracts/utils/
math/Math.sol#172)
        - result = prod0 * inverse (contracts/InstaDEX/@openzeppelin/contracts/utils/
math/Math.sol#199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-
before-multiply
INFO:Detectors:
ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/
InstaDEX/@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol#44-67) uses
timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > deadline (contracts/InstaDEX/@openzeppelin/contracts/
token/ERC20/extensions/ERC20Permit.sol#53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp
INFO:Detectors:
ShortStrings.toString(ShortString) (contracts/InstaDEX/@openzeppelin/contracts/utils/
ShortStrings.sol#63-73) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
ShortStrings.sol#68-71)
StorageSlot.getAddressSlot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/
utils/StorageSlot.sol#59-64) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#61-63)
StorageSlot.getBooleanSlot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/
```

```
utils/StorageSlot.sol#69-74) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#71-73)
StorageSlot.getBytes32Slot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/
utils/StorageSlot.sol#79-84) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#81-83)
StorageSlot.getUint256Slot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/
utils/StorageSlot.sol#89-94) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#91-93)
StorageSlot.getStringSlot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#99-104) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#101-103)
StorageSlot.getStringSlot(string) (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#109-114) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#111-113)
StorageSlot.getBytesSlot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#119-124) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#121-123)
StorageSlot.getBytesSlot(bytes) (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#129-134) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#131-133)
Strings.toString(uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/
Strings.sol#24-44) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
Strings.sol#30-32)
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/
Strings.sol#36-38)
ECDSA.tryRecover(bytes32,bytes) (contracts/InstaDEX/@openzeppelin/contracts/utils/
cryptography/ECDSA.sol#56-73) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/cryptography/
ECDSA.sol#64-68)
```

```
MessageHashUtils.toEthSignedMessageHash(bytes32) (contracts/InstaDEX/@openzeppelin/
contracts/utils/cryptography/MessageHashUtils.sol#30-37) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/cryptography/
MessageHashUtils.sol#32-36)
MessageHashUtils.toTypedDataHash(bytes32,bytes32) (contracts/InstaDEX/@openzeppelin/
contracts/utils/cryptography/MessageHashUtils.sol#76-85) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/cryptography/
MessageHashUtils.sol#78-84)
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) uses assembly
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#130-133)
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#154-161)
        - INLINE ASM (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#167-176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-
usage
INFO:Detectors:
Context._contextSuffixLength() (contracts/InstaDEX/@openzeppelin/contracts/utils/
Context.sol#25-27) is never used and should be removed
Context._msgData() (contracts/InstaDEX/@openzeppelin/contracts/utils/
Context.sol#21-23) is never used and should be removed
ECDSA.recover(bytes32,bytes) (contracts/InstaDEX/@openzeppelin/contracts/utils/
cryptography/ECDSA.sol#89-93) is never used and should be removed
ECDSA.recover(bytes32,bytes32,bytes32) (contracts/InstaDEX/@openzeppelin/contracts/
utils/cryptography/ECDSA.sol#112-116) is never used and should be removed
ECDSA.tryRecover(bytes32,bytes) (contracts/InstaDEX/@openzeppelin/contracts/utils/
cryptography/ECDSA.sol#56-73) is never used and should be removed
ECDSA.tryRecover(bytes32,bytes32,bytes32) (contracts/InstaDEX/@openzeppelin/
contracts/utils/cryptography/ECDSA.sol#100-107) is never used and should be removed
ERC20._burn(address,uint256) (contracts/InstaDEX/@openzeppelin/contracts/token/ERC20/
ERC20.sol#241-246) is never used and should be removed
Math.average(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#96-99) is never used and should be removed
Math.ceilDiv(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#107-115) is never used and should be removed
```

```
Math.log10(uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#321-353) is never used and should be removed
Math.log10(uint256,Math.Rounding) (contracts/InstaDEX/@openzeppelin/contracts/utils/
math/Math.sol#359-364) is never used and should be removed
Math.log2(uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#268-304) is never used and should be removed
Math.log2(uint256,Math.Rounding) (contracts/InstaDEX/@openzeppelin/contracts/utils/
math/Math.sol#310-315) is never used and should be removed
Math.log256(uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#372-396) is never used and should be removed
Math.log256(uint256,Math.Rounding) (contracts/InstaDEX/@openzeppelin/contracts/utils/
math/Math.sol#402-407) is never used and should be removed
Math.max(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#81-83) is never used and should be removed
Math.min(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#88-90) is never used and should be removed
Math.mulDiv(uint256,uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#123-202) is never used and should be removed
Math.mulDiv(uint256,uint256,uint256,Math.Rounding) (contracts/InstaDEX/@openzeppelin/
contracts/utils/math/Math.sol#207-213) is never used and should be removed
Math.sqrt(uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#221-252) is never used and should be removed
Math.sqrt(uint256,Math.Rounding) (contracts/InstaDEX/@openzeppelin/contracts/utils/
math/Math.sol#257-262) is never used and should be removed
Math.tryAdd(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#25-31) is never used and should be removed
Math.tryDiv(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#61-66) is never used and should be removed
Math.tryMod(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#71-76) is never used and should be removed
Math.tryMul(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#46-56) is never used and should be removed
Math.trySub(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#36-41) is never used and should be removed
Math.unsignedRoundsUp(Math.Rounding) (contracts/InstaDEX/@openzeppelin/contracts/
utils/math/Math.sol#412-414) is never used and should be removed
MessageHashUtils.toDataWithIntendedValidatorHash(address,bytes) (contracts/InstaDEX/
```

@openzeppelin/contracts/utils/cryptography/MessageHashUtils.sol#63-65) is never used and should be removed
MessageHashUtils.toEthSignedMessageHash(bytes) (contracts/InstaDEX/@openzeppelin/contracts/utils/cryptography/MessageHashUtils.sol#49-52) is never used and should be removed
MessageHashUtils.toEthSignedMessageHash(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/cryptography/MessageHashUtils.sol#30-37) is never used and should be removed
Nonces._useCheckedNonce(address,uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/Nonces.sol#40-45) is never used and should be removed
ShortStrings.byteLengthWithFallback(ShortString,string) (contracts/InstaDEX/@openzeppelin/contracts/utils/ShortStrings.sol#116-122) is never used and should be removed
SignedMath.abs(int256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/SignedMath.sol#37-42) is never used and should be removed
SignedMath.average(int256,int256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/SignedMath.sol#28-32) is never used and should be removed
SignedMath.max(int256,int256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/SignedMath.sol#13-15) is never used and should be removed
SignedMath.min(int256,int256) (contracts/InstaDEX/@openzeppelin/contracts/utils/math/SignedMath.sol#20-22) is never used and should be removed
StorageSlot.getAddressSlot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/StorageSlot.sol#59-64) is never used and should be removed
StorageSlot.getBooleanSlot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/StorageSlot.sol#69-74) is never used and should be removed
StorageSlot.getBytes32Slot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/StorageSlot.sol#79-84) is never used and should be removed
StorageSlot.getBytesSlot(bytes) (contracts/InstaDEX/@openzeppelin/contracts/utils/StorageSlot.sol#129-134) is never used and should be removed
StorageSlot.getBytesSlot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/StorageSlot.sol#119-124) is never used and should be removed
StorageSlot.getStringSlot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/StorageSlot.sol#99-104) is never used and should be removed
StorageSlot.getUint256Slot(bytes32) (contracts/InstaDEX/@openzeppelin/contracts/utils/StorageSlot.sol#89-94) is never used and should be removed
Strings.equal(string,string) (contracts/InstaDEX/@openzeppelin/contracts/utils/Strings.sol#91-93) is never used and should be removed

```
Strings.toHexString(address) (contracts/InstaDEX/@openzeppelin/contracts/utils/
Strings.sol#84-86) is never used and should be removed
Strings.toHexString(uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/
Strings.sol#56-60) is never used and should be removed
Strings.toHexString(uint256,uint256) (contracts/InstaDEX/@openzeppelin/contracts/
utils/Strings.sol#65-78) is never used and should be removed
Strings.toString(uint256) (contracts/InstaDEX/@openzeppelin/contracts/utils/
Strings.sol#24-44) is never used and should be removed
Strings.toStringSigned(int256) (contracts/InstaDEX/@openzeppelin/contracts/utils/
Strings.sol#49-51) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/interfaces/
IERC5267.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/interfaces/draft-
IERC6093.sol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/token/ERC20/
ERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/token/ERC20/
IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/token/ERC20/
extensions/ERC20Permit.sol#4) necessitates a version too recent to be trusted.
Consider deploying with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/token/ERC20/
extensions/IERC20Metadata.sol#4) necessitates a version too recent to be trusted.
Consider deploying with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/token/ERC20/
extensions/IERC20Permit.sol#4) necessitates a version too recent to be trusted.
Consider deploying with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/
Context.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/
```

```
Nonces.sol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/
ShortStrings.sol#4) necessitates a version too recent to be trusted. Consider
deploying with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/
StorageSlot.sol#5) necessitates a version too recent to be trusted. Consider
deploying with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/
Strings.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/cryptography/
ECDSA.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/cryptography/
EIP712.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/cryptography/
MessageHashUtils.sol#4) necessitates a version too recent to be trusted. Consider
deploying with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
Math.sol#4) necessitates a version too recent to be trusted. Consider deploying with
0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/@openzeppelin/contracts/utils/math/
SignedMath.sol#4) necessitates a version too recent to be trusted. Consider
deploying with 0.8.18.
Pragma version^0.8.20 (contracts/InstaDEX/InstaDEX.sol#4) necessitates a version too
recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.23 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Function EIP712._EIP712Name() (contracts/InstaDEX/@openzeppelin/contracts/utils/
cryptography/EIP712.sol#146-148) is not in mixedCase
Function EIP712._EIP712Version() (contracts/InstaDEX/@openzeppelin/contracts/utils/
cryptography/EIP712.sol#157-159) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-
```

```
to-solidity-naming-conventions
INFO:Detectors:
ShortStrings.slitherConstructorConstantVariables() (contracts/InstaDEX/@openzeppelin/
contracts/utils/ShortStrings.sol#40-123) uses literals with too many digits:
        - FALLBACK_SENTINEL =
0x00000000000000000000000000000000000000000000000000000000000000FF (contracts/
InstaDEX/@openzeppelin/contracts/utils/ShortStrings.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits
```

RUGDOG

WOOF!

rugdog.net

the@rugdog.net