Revoluzion Audit

SEAL OF REVOLUZION AUDIT

INTEGRITY

Audit Report

Name	: Spyro
Symbol	: SPYRO
Decimals	: 18
Address	: 0x6D7497751656618Fc38CfB5478994a20F7E235df
Owner	: 0x82b3CeA682daA1276d187Ec35c91c6A76daE2309
Network	: Ethereum
Type	: ERC20
Audited on	: 8 February 2024
Audited Score	: 95%

Table of Contents

Project Overview

Name	Spyro
Symbol	SPYRO
Decimals	18
Total Supply	1,000,000,000,000
Tax	No Tax
Compiler Version	v0.8.23+commit.f704f362
Optimization	Yes with 1000 runs
License Type	MIT
Explorer Link	https://etherscan.io/address/0x6d7497751656618fc38cfb5478994a20f7e235df
Create Tx	https://etherscan.io/tx/0x3205029107157734dd0f1fffcd32e333aff9193e7fc383c737528f337abfff52
Creator	0x842BAAbD96f88a5Fbe5389Ffd8DE768fB286b533
Featured Wallet	N/A

Project Description

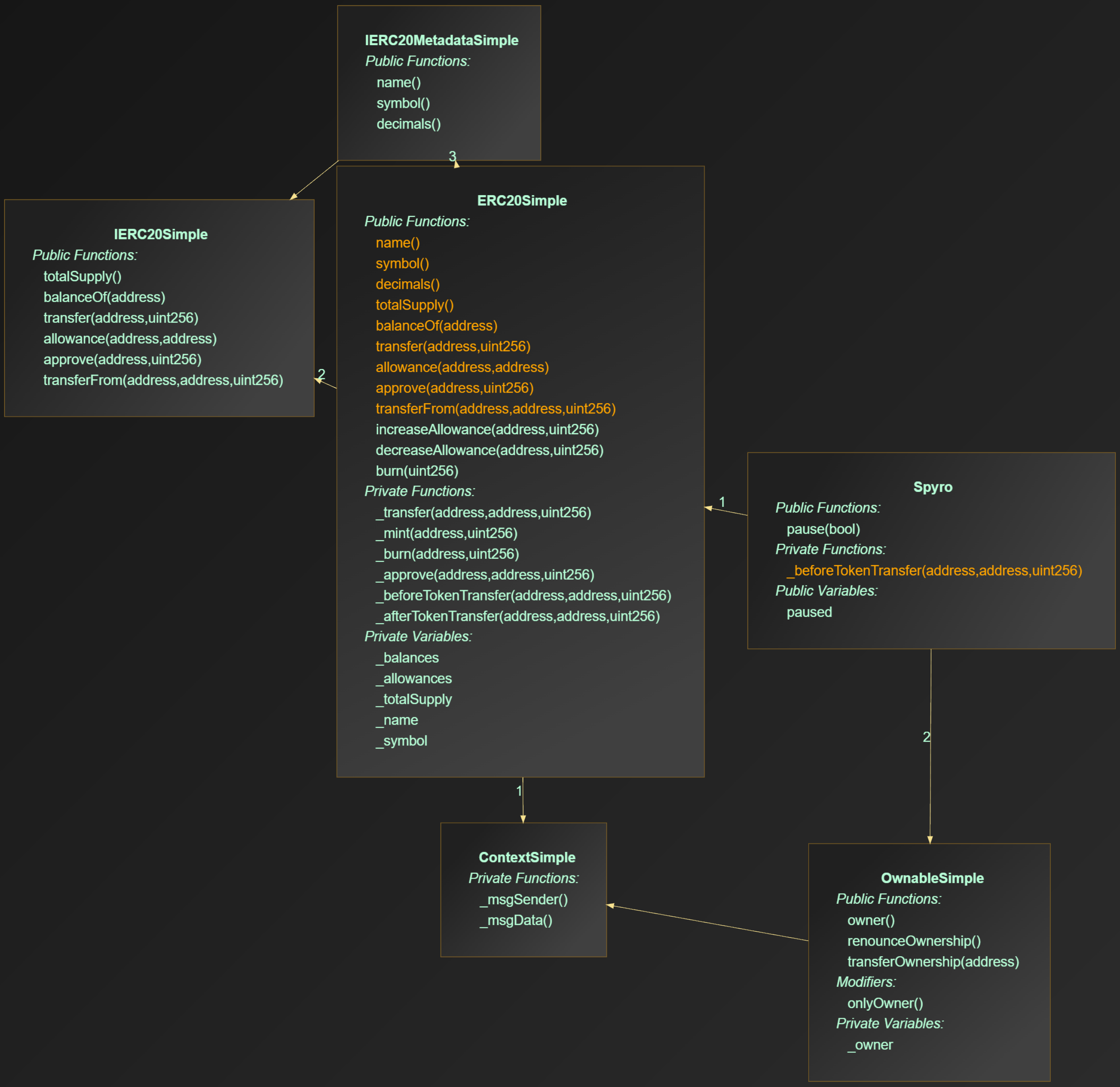
According to their website

No project info provided.

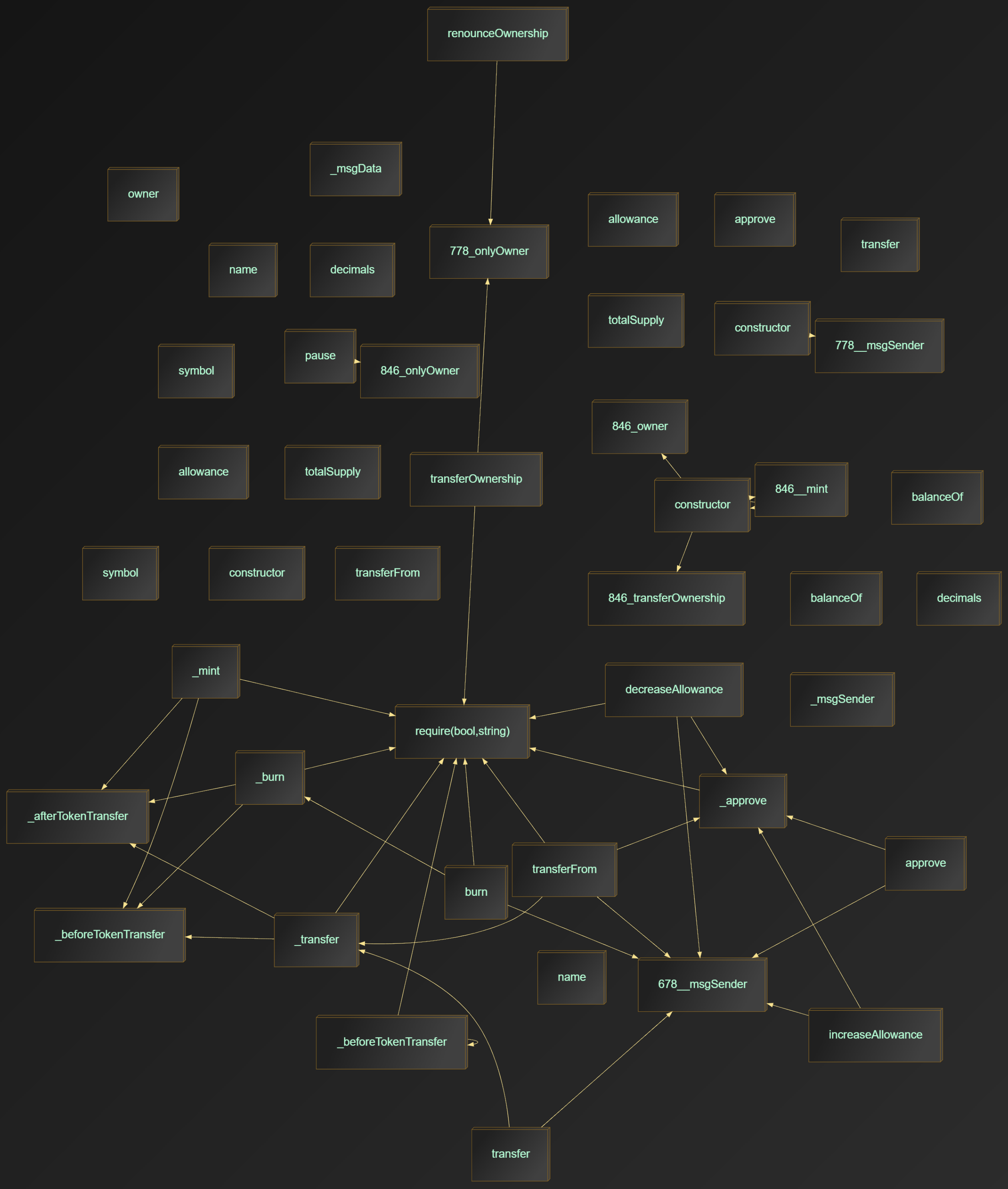
Release Date	: TBA
Category	: DeFi

Contract Functions Interaction

Inheritance Graph



Call Graph (All)



Audit Overview

Threat Level

When conducting audit on smart contract(s), we first look for known vulnerabilities and issues within the code because any exploitation on such vulnerabilities and issues by malicious actors could potentially result in serious financial damage to the projects. All the issues and vulnerabilities will be categorized into the categories as provided below.

Critical

This category provides issues and vulnerabilities that are critical to the performance/functionality of the smart contract and should be fixed by project creator before moving to a live environment.

Medium

This category provides issues and vulnerabilities that are not that critical to the performance/functionality of the smart contract but is recommended to be fixed by project creator before moving to a live environment.

Minor

This category provides issues and vulnerabilities that are minor to the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment.

Informational

This category provides issues and vulnerability that have insignificant effect on the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment. However, fixing them can further improve the efficacy or security for features with a risk-free factor.

Notable Information

- Contract Owner cannot stop or pause transactions.
- Contract Owner cannot transfer tokens from specific address.
- Contract Owner cannot mint new tokens after deploying smart contract.
- Contract Owner cannot burn tokens from specific wallet.
- Contract Owner cannot blacklist wallet.
- There are no compiler warnings when compiling the smart contracts.
- Contract is using safe Zeppelin modules.
- Contract is a standard ERC20 token without any buy, sell or transfer tax and there is no max txn or max wallet limit.
- Project owner should be aware that upon deployment of the smart contract, the ownership of the smart contract will directly be transferred to 0x82b3CeA682daA1276d187Ec35c91c6A76daE2309.
- Project owner should be aware that upon deployment of the smart contract, the smart contract ownership will be transferred first before the token will be minted, hence the token for the initial supply will be minted to 0x82b3CeA682daA1276d187Ec35c91c6A76daE2309.
- Smart contract owner need to remember not to set the current state for paused when initiating pause function since the function to change the state does not have a restriction to prevent such action which is just a waste of gas for the owner.
- Users should be aware that the ownership of the smart contract has been renounced at this transaction: https://etherscan.io/tx/0x17f34110a7d707b56e086d3f81e7222cd68ff3644354f45dee79d7717adb7aea

Bugs and Optimizations Detection

This table is based on the result obtained from running the smart contract through Slither's Solidity static analysis.

What it detects	Impact	Confidence	Status
Storage abiencoderv2 array	High	High	Passed
transferFrom uses arbitrary from	High	High	Passed
Modifying storage array by value	High	High	Passed
The order of parameters in a shift instruction is incorrect.	High	High	Passed
Multiple constructor schemes	High	High	Passed
Contract's name reused	High	High	Passed
Detected unprotected variables	High	High	Passed
Public mappings with nested variables	High	High	Passed
Right-To-Left-Override control character is used	High	High	Passed
State variables shadowing	High	High	Passed
Functions allowing anyone to destruct the contract	High	High	Passed
Uninitialized state variables	High	High	Passed
Uninitialized storage variables	High	High	Passed

Unprotected upgradeable contract	High	High	Passed
transferFrom uses arbitrary from with permit	High	Medium	Passed
Functions that send Ether to arbitrary destinations	High	Medium	Passed
Tainted array length assignment	High	Medium	Passed
Controlled delegatecall destination	High	Medium	Passed
Payable functions using delegatecall inside a loop	High	Medium	Passed
msg.value inside a loop	High	Medium	Passed
Reentrancy vulnerabilities (theft of ethers)	High	Medium	Passed
Signed storage integer array compiler bug	High	Medium	Passed
Unchecked tokens transfer	High	Medium	Passed
Weak PRNG	High	Medium	Passed
Detects ERC20 tokens that have a function whose signature collides with EIP-2612's DOMAIN_SEPARATOR()	Medium	High	Passed
Detect dangerous enum conversion	Medium	High	Passed
Incorrect ERC20 interfaces	Medium	High	Passed
Incorrect ERC721 interfaces	Medium	High	Passed
Dangerous strict equalities	Medium	High	Passed

Contracts that lock ether	Medium	High	Passed
Deletion on mapping containing a structure	Medium	High	Passed
State variables shadowing from abstract contracts	Medium	High	Passed
Tautology or contradiction	Medium	High	Passed
Unused write	Medium	High	Passed
Misuse of Boolean constant	Medium	Medium	Passed
Constant functions using assembly code	Medium	Medium	Passed
Constant functions changing the state	Medium	Medium	Passed
Imprecise arithmetic operations order	Medium	Medium	Passed
Reentrancy vulnerabilities (no theft of ethers)	Medium	Medium	Passed
Reused base constructor	Medium	Medium	Passed
Dangerous usage of tx.origin	Medium	Medium	Passed
Unchecked low-level calls	Medium	Medium	Passed
Unchecked send	Medium	Medium	Passed
Uninitialized local variables	Medium	Medium	Passed
Unused return values	Medium	Medium	Passed
Modifiers that can return the default value	Low	High	Passed

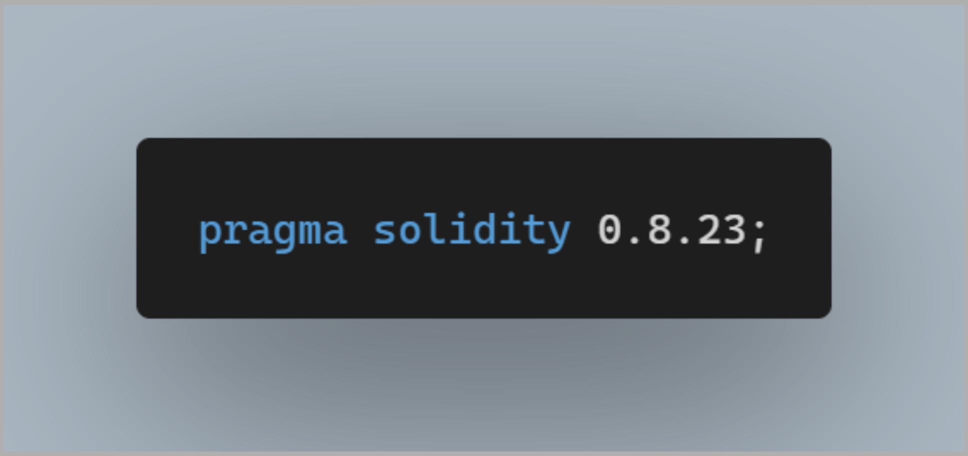
Built-in symbol shadowing	Low	High	Passed
Local variables shadowing	Low	High	Passed
Uninitialized function pointer calls in constructors	Low	High	Passed
Local variables used prior their declaration	Low	High	Passed
Constructor called not implemented	Low	High	Passed
Multiple calls in a loop	Low	Medium	Passed
Missing Events Access Control	Low	Medium	Passed
Missing Events Arithmetic	Low	Medium	Passed
Dangerous unary expressions	Low	Medium	Passed
Missing Zero Address Validation	Low	Medium	Passed
Benign reentrancy vulnerabilities	Low	Medium	Passed
Reentrancy vulnerabilities leading to out-of-order Events	Low	Medium	Passed
Dangerous usage of block.timestamp	Low	Medium	Passed
Assembly usage	Informational	High	Passed
Assert state change	Informational	High	Passed
Comparison to boolean constant	Informational	High	Passed
Deprecated Solidity Standards	Informational	High	Passed
Un-indexed ERC20 event parameters	Informational	High	Passed

Function initializing state variables	Informational	High	Passed
Low level calls	Informational	High	Passed
Missing inheritance	Informational	High	Passed
Conformity to Solidity naming conventions	Informational	High	Passed
If different pragma directives are used	Informational	High	Passed
Redundant statements	Informational	High	Moderated
Incorrect Solidity version	Informational	High	Moderated
Unimplemented functions	Informational	High	Passed
Unused state variables	Informational	High	Passed
Costly operations in a loop	Informational	Medium	Passed
Functions that are not used	Informational	Medium	Moderated
Reentrancy vulnerabilities through send and transfer	Informational	Medium	Passed
Variable names are too similar	Informational	Medium	Passed
Conformance to numeric notation best practices	Informational	Medium	Passed
State variables that could be declared constant	Optimization	High	Passed
Public function that could be declared external	Optimization	High	Passed

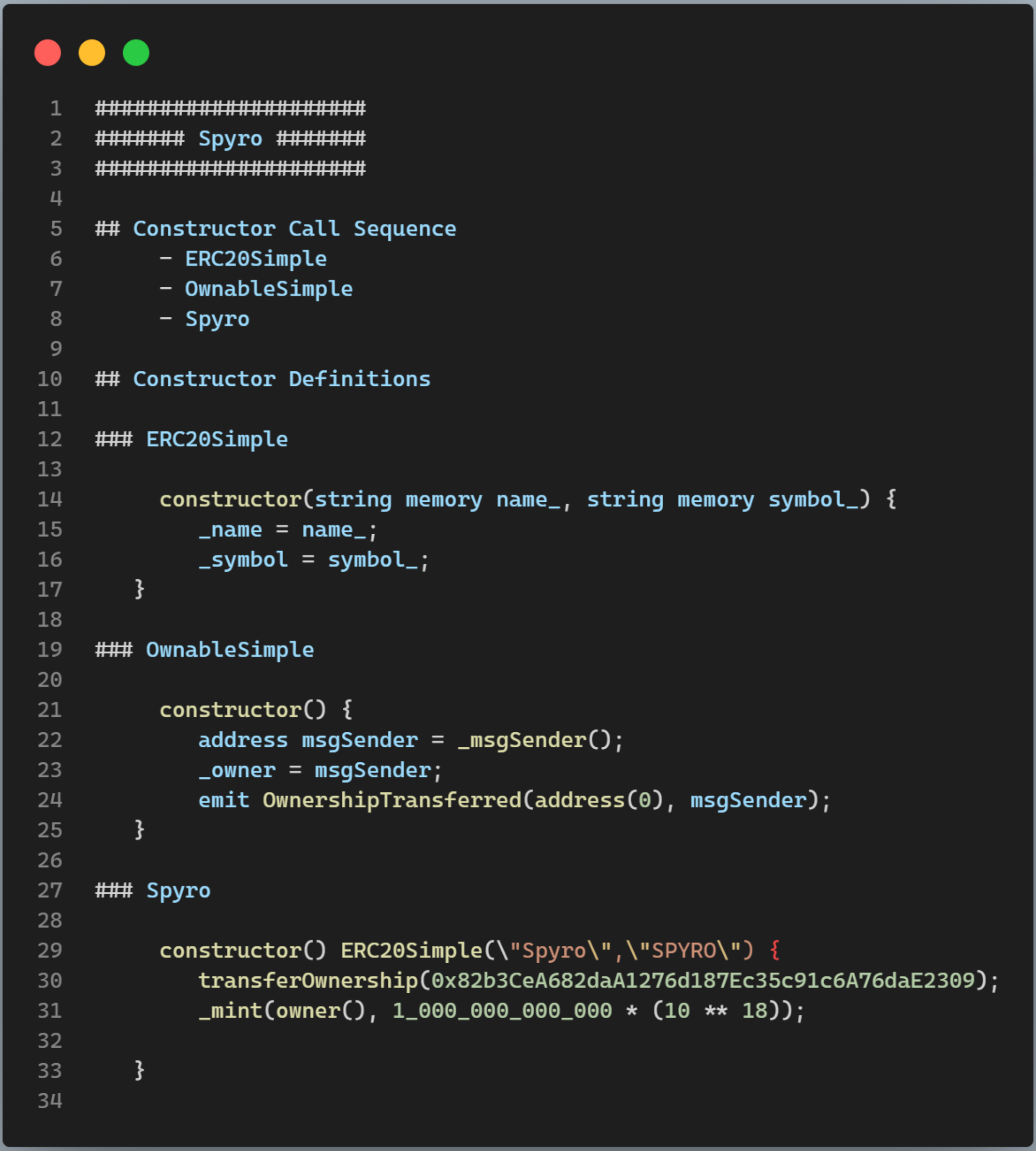
Contract Diagnostic

CODE	SEVERITY	DESCRIPTION
SV	Informational	Incorrect Solidity version

SV — Incorrect Solidity Version

SEVERITY	Informational
LOCATION(S)	L5
 `pragma solidity 0.8.23;`	
DESCRIPTION	The pragma directive in the smart contract specifies the use of version 0.8.23 of the Solidity compiler, which is deemed too recent to be trusted. The concern here is similar to the first issue, indicating that the specified version hasn't been sufficiently vetted by the community and might contain undiscovered bugs.
RECOMMENDATIONS	It is advisable to downgrade the Solidity compiler version to a more established and tested version, such as 0.8.18. This can be done by adjusting the pragma line at the beginning of your Solidity file. For example, change pragma solidity ^0.8.23; to pragma solidity ^0.8.18;. This change will help ensure that the contract is compiled with a version of the compiler that is better understood and has a stronger track record of reliability.
STATUS	**Revoluzion acknowledgement:** Unresolved and should not have any major effect..

Constructor Calls



```
#####################
####### Spyro #######
#####################

## Constructor Call Sequence
     - ERC20Simple
     - OwnableSimple
     - Spyro

## Constructor Definitions

### ERC20Simple

     constructor(string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
   }

### OwnableSimple

     constructor() {
        address msgSender = _msgSender();
        _owner = msgSender;
        emit OwnershipTransferred(address(0), msgSender);
   }

### Spyro

     constructor() ERC20Simple(\"Spyro\",\"SPYRO\") {
        transferOwnership(0x82b3CeA682daA1276d187Ec35c91c6A76daE2309);
        _mint(owner(), 1_000_000_000_000 * (10 ** 18));

   }

```

Disclaimer

This report only shows findings based on our limited project analysis according to the good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall online presence and team transparency details of which are set out in this report. To get a full view of our analysis, it is important for you to read the full report. Under no circumstances did Revoluzion Audit receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Our team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

While we have done our best to conduct thorough analysis to produce this report, it is crucial to note that you should not rely solely on this report and use the content provided in this document as financial advice or a reason to buy any investment. The Our team disclaims any liability against us for the resulting losses based on the you decision made by relying on the content of this report. You must conduct your own independent investigations before making any decisions to protect yourselves from being scammed. We go into more detail on this in the disclaimer clause in the next page — please make sure to read it in full.

Full Disclaimer Clause

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy all copies of this report downloaded and/or printed by you. This report is provided for information purposes only, on a non-reliance basis and does not constitute to any investment advice.

No one shall have any right to rely on the report or its contents, and Revoluzion Audit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (collectively known as Revoluzion) owe no duty of care towards you or any other person, nor do we make any warranty or representation to any person on the accuracy or completeness of the report.

The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Revoluzion hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report.

Except and only to the extent that it is prohibited by law, Revoluzion hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Revoluzion, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts, website, social media, and team.