Smart Contract
Security Audit
V1

Sirius DAO Smart Contract

14/5/2022

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

Table of Contents

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Name**: Sirius DAO
- **Ticker**: SRS
- **Website**: https://siriusdao.finance/
- **Twitter**: https://twitter.com/sirius_dao
- **Telegram**: https://t.me/siriusdao
- **Medium**: https://medium.com/@Siriusdao
- **Discord**: https://discord.com/invite/siriusdao
- **Instagram**: https://www.instagram.com/siriusdao/
- **Platform**: Binance Smart Chain Network
- **Contract Address**: 0xa61b3beabD81464FC19830597F2F769f4eebD6d1
- **Code:**

https://bscscan.com/address/0xa61b3beabd81464fc19830597f2f769f4eebd6d1#code

Contracts address deployed to test net (BSC)

Sirius (SRS)Token contract on BSC test net to test every function by the auditor.

https://testnet.bscscan.com/address/0xdd8bef05207b9517dfd31ed2b6688ee1dc3e3eda

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.

Well Secured	
Secured	✓
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 1 medium, 2 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

Sirius.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Sirius.sol	**a8638c6009c618c7a35f1891f7582d43de081d2bd211c61069c4586bbc87f396**	0x3e181764f67360f6cc89cf267cec0e53682f2399

- Contract: Sirius
- Inherit: ERC20, ERC20Burnable, Pausable, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
nonces	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
totalSuppy	✓	Read / public	**Passed**
DOMAIN_SEPARATOR	✓	Read / public	**Passed**
vault	✓	Read / public	**Passed**
PERMIT_TYPEHASH	✓	Read / public	**Passed**

decreaseAllowance	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
mint	✓	Write / public	**Passed**
burn	✓	Write / public	**Passed**
burnFrom	✓	Write / public	**Passed**
approve	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
setVault	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
permit	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Design Logic.	**Passed**
6	Timestamp dependence.	**Passed with Notes**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed with Notes**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

No Critical severity vulnerabilities were found

High:

No High severity vulnerabilities were found

Medium:

#Centralization Risks

Description

The vault has the authority to :

- Can mint new tokens. this represents a risk for the users because in that case their funds will be lower if the vault mint more SRS.

```
function mint(address account_, uint256 amount_) external onlyVault() {
        _mint(account_, amount_);
    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before mint more tokens to give them time if they want to do anything.

P.S: This issue is common to the majority of Some Token's smart contracts.

Status: Acknowledged by the Auditee

Low:

Functions that do not have a function visibility

Description

Functions that do not have a function visibility type specified are public by default. This can lead to a vulnerability if a developer forgot to set the visibility and a malicious user is able to make unauthorized or unintended state changes.

```
  function permit(
        address owner,
        address spender,
        uint256 amount,
        uint256 deadline,
```

```
        uint8 v,
        bytes32 r,
        bytes32 s
    ) public virtual override {
        require(block.timestamp <= deadline, "Permit: expired deadline");
        bytes32 hashStruct =
            keccak256(abi.encode(PERMIT_TYPEHASH, owner, spender, amount,
_nonces[owner].current(), deadline));
        bytes32 _hash = keccak256(abi.encodePacked(uint16(0x1901),
DOMAIN_SEPARATOR, hashStruct));
        address signer = ecrecover(_hash, v, r, s);
        require(signer != address(0) && signer == owner, "ERC20Permit: Invalid
signature");
        _nonces[owner].increment();
        _approve(owner, spender, amount);}
```

Remediation

The team should Make this constructor with clear visibility of at least the initial supply of the token.

Status: Acknowledged by the Auditee because it ERC20 Permit library.

#Use of block.timestamp for comparisons

Description

The value of block.timestamp can be manipulated by the miner. And conditions with strict equality is difficult to achieve -block.timestamp

Remediation

Avoid use of block.timestamp

Status: Acknowledged

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Compiler version is old

Description

The compiler being used was released a year – a year and half ago. It's recommended to use more recent compiler version, there can be benefits like reduction in bytecode size etc.

Status: Acknowledged

Automatic Testing

1- Check for security

a8638c6009c618c7a35f1891f7582d43de081d2bd211c61069c4586bbc87f396

File: Sirius.sol | Language: solidity | Size: 12897 bytes | Date: 2022-05-12T12:11:22.324Z

Critical	High	Medium	Low	Note
0	0	0	0	0

2- SOLIDITY STATIC ANALYSIS

SOLIDITY STATIC ANALYSIS

Select all | Autorun | Run

Security

Select Security

- **Transaction origin:** 'tx.origin' used
- **Check-effects-interaction:** Potential reentrancy bugs
- **Inline assembly:** Inline assembly used
- **Block timestamp:** Can be influenced by miners
- **Low level calls:** Should only be used by experienced devs
- **Block hash:** Can be influenced by miners
- **Selfdestruct:** Contracts using destructed contract can be broken

Gas & Economy

Select Gas & Economy

- **Gas costs:** Too high gas requirement of functions
- **This on local calls:** Invocation of local functions via 'this'
- **Delete dynamic array:** Use require/assert to ensure complete deletion
- **For loop over dynamic array:** Iterations depend on dynamic array's size
- **Ether transfer in loop:** Transferring Ether in a for/while/do-while loop

SOLIDITY STATIC ANALYSIS

ERC

Select ERC

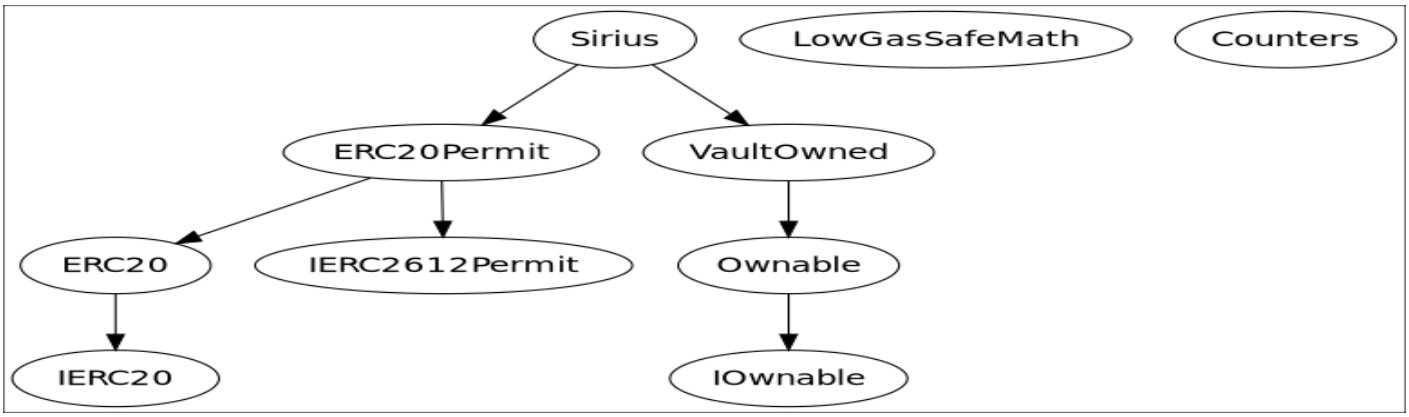
- **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

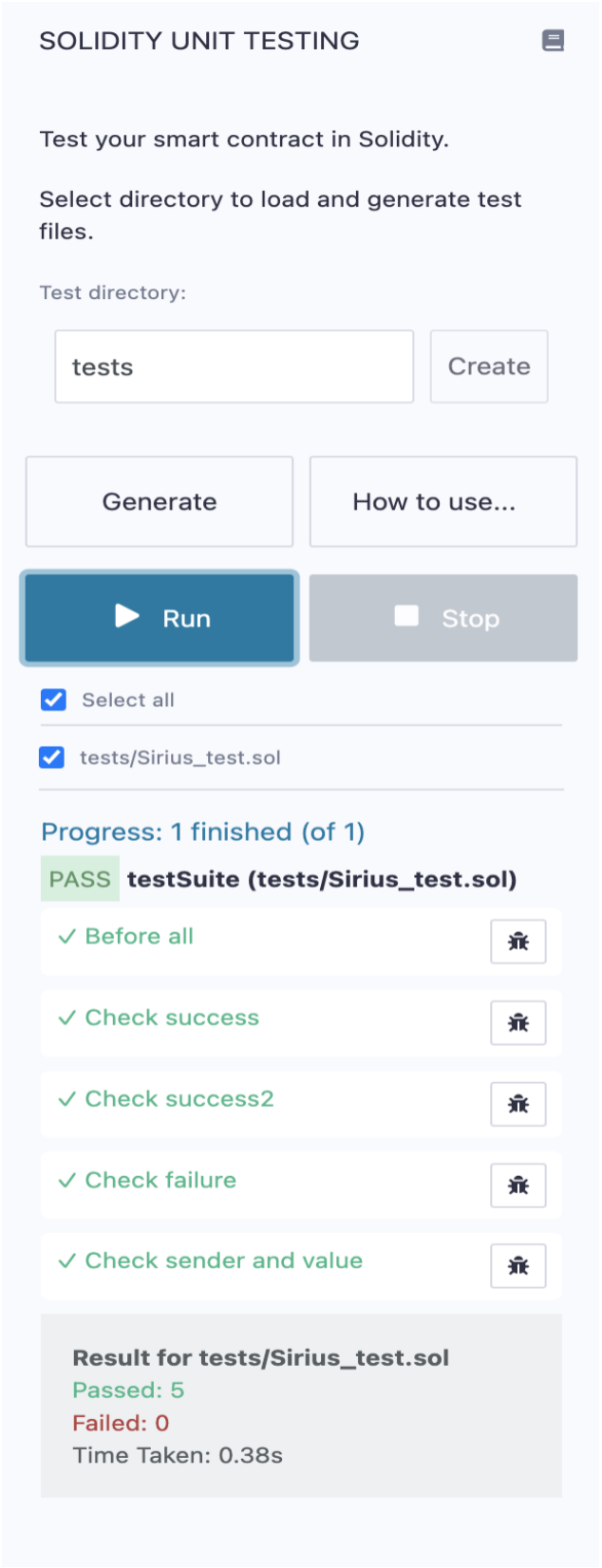
Select Miscellaneous

- **Constant/View/Pure functions:** Potentially constant/view/pure functions
- **Similar variable names:** Variable names are too similar
- **No return:** Function with 'returns' not returning
- **Guard conditions:** Ensure appropriate use of require/assert
- **Result not used:** The result of an operation not used
- **String length:** Bytes length != String length
- **Delete from dynamic array:** 'delete' leaves a gap in array
- **Data truncated:** Division on int/uint values truncates the result

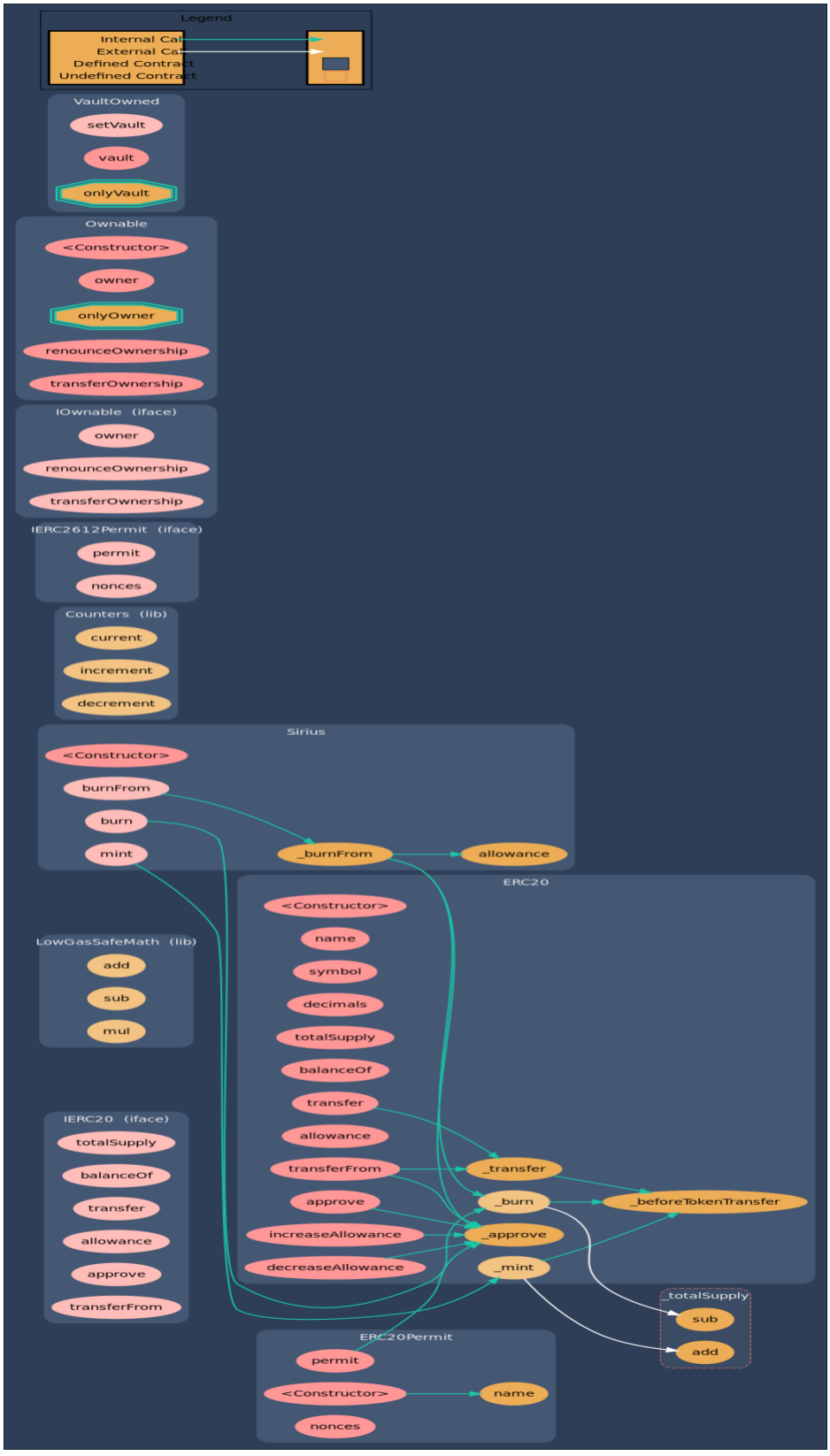
3- Inheritance graph



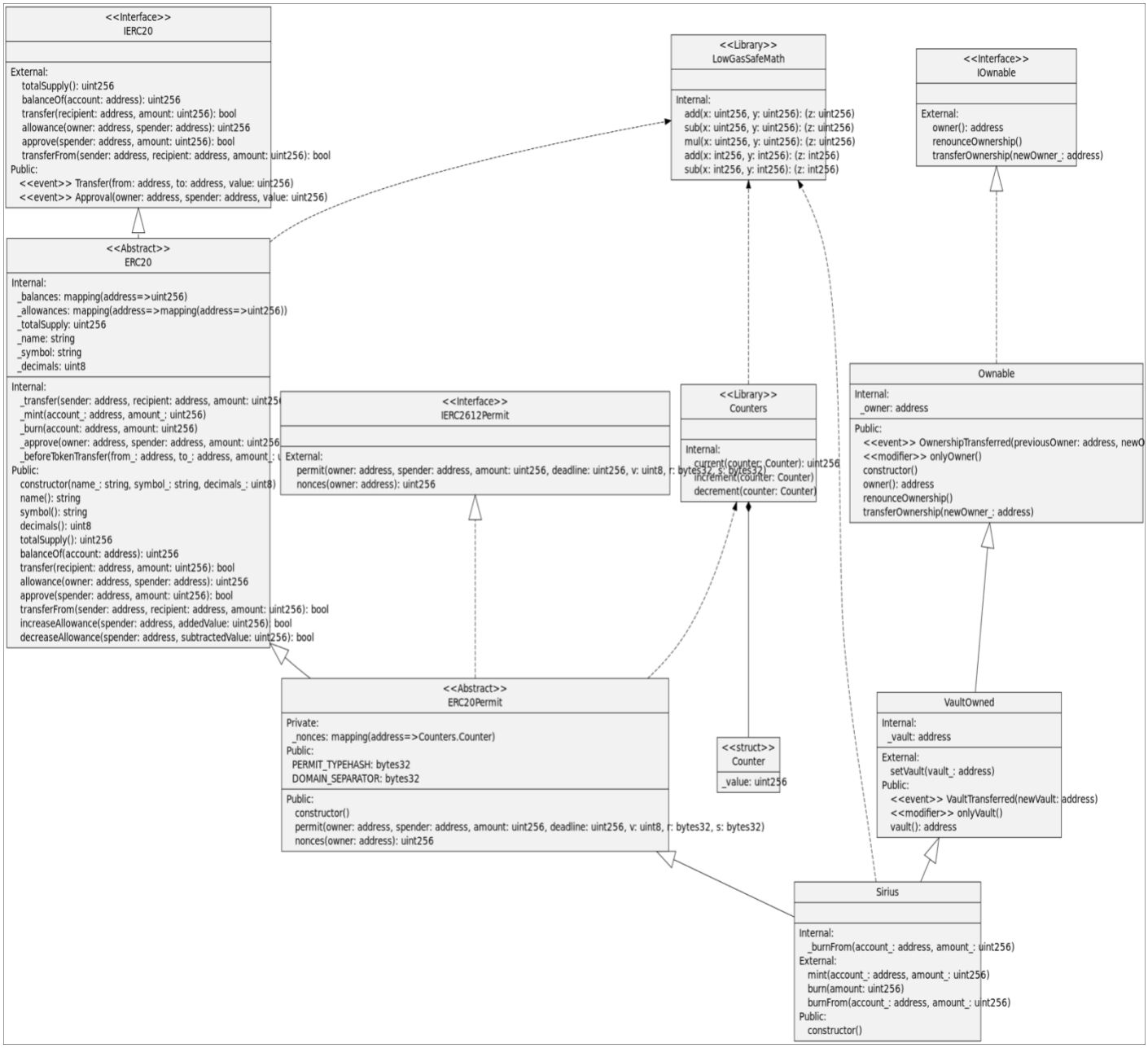
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
VaultOwned
setVault
vault
onlyVault
Ownable
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
IOwnable (iface)
owner
renounceOwnership
transferOwnership
IERC2612Permit (iface)
permit
nonces
Counters (lib)
current
increment
decrement
Sirius
<Constructor>
burnFrom
burn
mint
_burnFrom
allowance
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
approve
increaseAllowance
decreaseAllowance
_transfer
_burn
_approve
_mint
_beforeTokenTransfer
LowGasSafeMath (lib)
add
sub
mul
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
_totalSupply
sub
add
ERC20Permit
permit
<Constructor>
nonces
name

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a5f3c23b  =>  add(int256,int256)
adefc37b  =>  sub(int256,int256)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
ad04a8d1  =>  current(Counter)
e2bee435  =>  increment(Counter)
854ec98e  =>  decrement(Counter)
d505accf  =>
permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
7ecebe00  =>  nonces(address)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
6817031b  =>  setVault(address)
fbfa77cf  =>  vault()
40c10f19  =>  mint(address,uint256)
42966c68  =>  burn(uint256)
79cc6790  =>  burnFrom(address,uint256)
a22b35ce  =>  _burnFrom(address,uint256)
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Sirius.sol	8987aac7ee485f43bcbb2601da0ad75b73e53271

Contracts Description Table

Contract	Type	Bases				
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**		
IERC20	Interface					
└	totalSupply	External ❗		NO❗		
└	balanceOf	External ❗		NO❗		
└	transfer	External ❗	🛑	NO❗		
└	allowance	External ❗		NO❗		
└	approve	External ❗	🛑	NO❗		
└	transferFrom	External ❗	🛑	NO❗		
LowGasSafeMath	Library					
└	add	Internal 🔒				
└	sub	Internal 🔒				
└	mul	Internal 🔒				
└	add	Internal 🔒				
└	sub	Internal 🔒				
ERC20	Implementation	IERC20				
└	<Constructor>	Public ❗	🛑	NO❗		
└	name	Public ❗		NO❗		
└	symbol	Public ❗		NO❗		
└	decimals	Public ❗		NO❗		
└	totalSupply	Public ❗		NO❗		
└	balanceOf	Public ❗		NO❗		
└	transfer	Public ❗	🛑	NO❗		
└	allowance	Public ❗		NO❗		
└	approve	Public ❗	🛑	NO❗		
└	transferFrom	Public ❗	🛑	NO❗		
└	increaseAllowance	Public ❗	🛑	NO❗		
└	decreaseAllowance	Public ❗	🛑	NO❗		

└	_transfer	Internal 🔒	🛑		
└	_mint	Internal 🔒	🛑		
└	_burn	Internal 🔒	🛑		
└	_approve	Internal 🔒	🛑		
└	_beforeTokenTransfer	Internal 🔒	🛑		
Counters	Library				
└	current	Internal 🔒			
└	increment	Internal 🔒	🛑		
└	decrement	Internal 🔒	🛑		
IERC2612Permit	Interface				
└	permit	External ❗	🛑	NO❗	
└	nonces	External ❗		NO❗	
ERC20Permit	Implementation	ERC20, IERC2612Permit			
└	<Constructor>	Public ❗	🛑	NO❗	
└	permit	Public ❗	🛑	NO❗	
└	nonces	Public ❗		NO❗	
IOwnable	Interface				
└	owner	External ❗		NO❗	
└	renounceOwnership	External ❗	🛑	NO❗	
└	transferOwnership	External ❗	🛑	NO❗	
Ownable	Implementation	IOwnable			
└	<Constructor>	Public ❗	🛑	NO❗	
└	owner	Public ❗		NO❗	
└	renounceOwnership	Public ❗	🛑	onlyOwner	
└	transferOwnership	Public ❗	🛑	onlyOwner	
VaultOwned	Implementation	Ownable			
└	setVault	External ❗	🛑	onlyOwner	
└	vault	Public ❗		NO❗	
Sirius	Implementation	ERC20Permit, VaultOwned			
└	<Constructor>	Public ❗	🛑	ERC20	
└	mint	External ❗	🛑	onlyVault	
└	burn	External ❗	🛑	NO❗	
└	burnFrom	External ❗	🛑	NO❗	
└	_burnFrom	Internal 🔒	🛑		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts are written systematically. Team found critical issues. So, it is no need to redeploy the contract.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “ Secured”.

✓ No volatile code.
✓ Not many high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

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 any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico 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 Saferico 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, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, 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 alone. No applications or operations were reviewed for security. No product code has been reviewed.