Smart Contract Security Audit V1

Prometheum Prodigy Smart Contract Audit

Sep 28, 2023

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 and Token Information

- **Platform**: Ethereum
- **Name**: Prometheum Prodigy
- **Language** : solidity
- **Contract Address**: 0x1075b82974490c604B76c49fFE91728595767ea1
- **Code Source**: **https://etherscan.io/address/0x1075b82974490c604b76c49ffe91728595767ea1#code**

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well 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, 0 medium, 2 low, 0 very low-level issues and 0 note in all solidity files of the contract

The files:

PrometheumProdigy.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
PrometheumProdigy.sol	8e63dcf5ae9024b5e7615064ea2a7b54cf0f1b984	0x1075b82974490c604B76c49fFE91728595767ea1

- Contract: PrometheumProdigy
- Inherit: Context, IERC20, 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**
decimals	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
getOwner	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
isExcluded	✓	Read / public	**Passed**
reflectionFromToken	✓	Read / public	**Passed**
rOwned	✓	Read / public	**Passed**
rTotal	✓	Read / public	**Passed**
tokenFromReflection	✓	Read / public	**Passed**

totalReflections	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
mint	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
changeDev	✓	Write / public	**Passed**
changeTaxes	✓	Write / public	**Passed**
disableMaxBalance	✓	Write / public	**Passed**
excludeAccount	✓	Write / public	**Passed**
includeAccount	✓	Write / public	**Passed**
flipPause	✓	Write / public	**Passed**
reflect	✓	Write / public	**Passed**
enableMaxBalance	✓	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**
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:

No Medium severity vulnerabilities were found.

Low:

No zero-address validation for some functions

When the owner wants to deploy the smart contract, he need to add 2 addresses the dev address and the mint address, he has to check for the zero address to make, he didn't add the zero address. Otherwise, the mint address will act like the burn address, the same for the dev address .

```
constructor (address _toMint, address _dev) public Ownable() {
        _rOwned[_toMint] = _rTotal;
        dev = _dev;
        excludeAccount(_toMint);
        excludeAccount(_dev);
        emit Transfer(address(0), _toMint, _tTotal);
    }
```

Recommendation

Use the require statement to check for zero addresses.

Status

Acknowledged, the team deploy the contract correctly.

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.20 instead of ^0.8.18). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Remediation

Remove the ^ sign to lock the pragma version.

Status: Acknowledged.

Very Low:

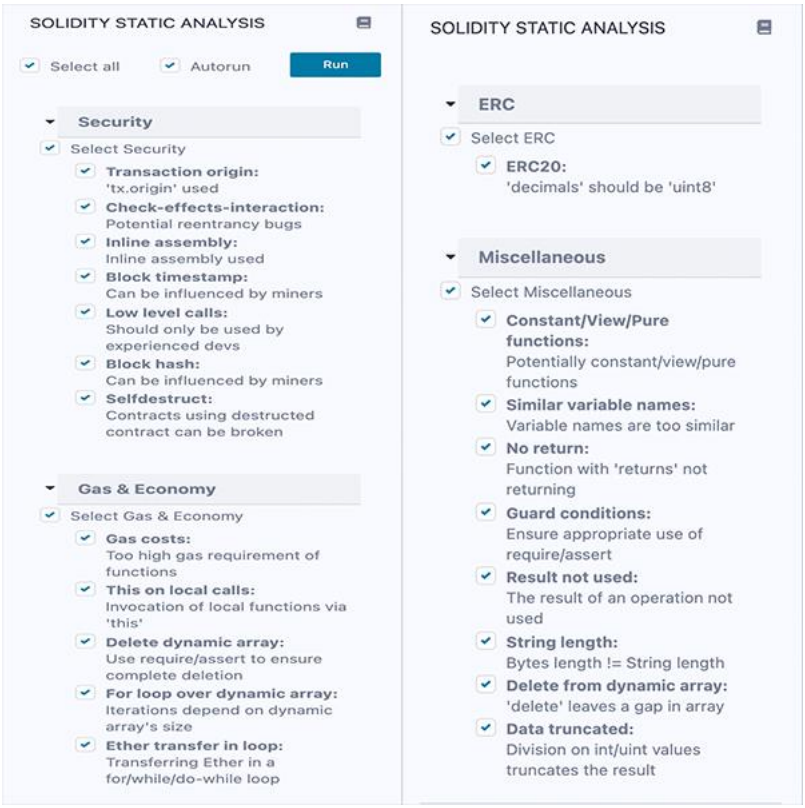
No Very Low severity vulnerabilities were found.

Notes:

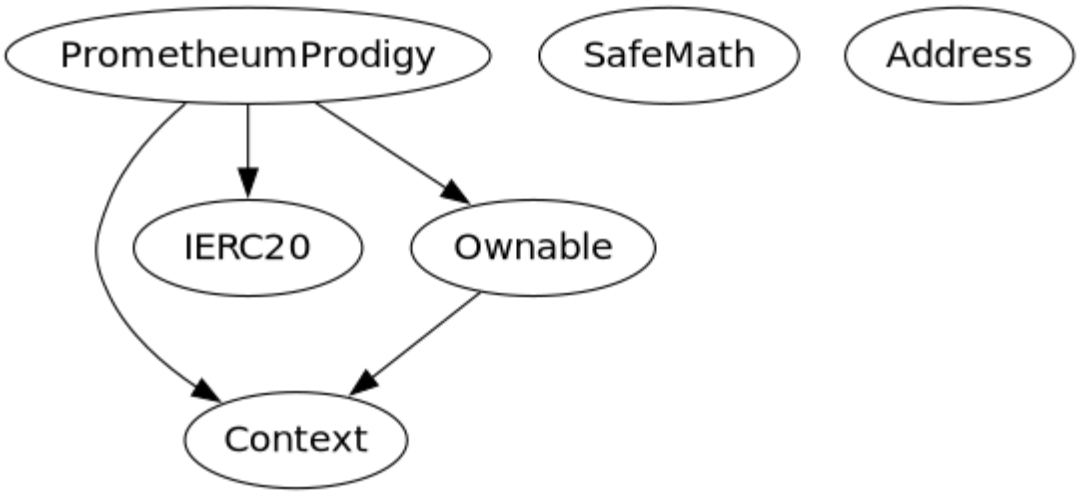
No Notes were found.

Automatic Testing

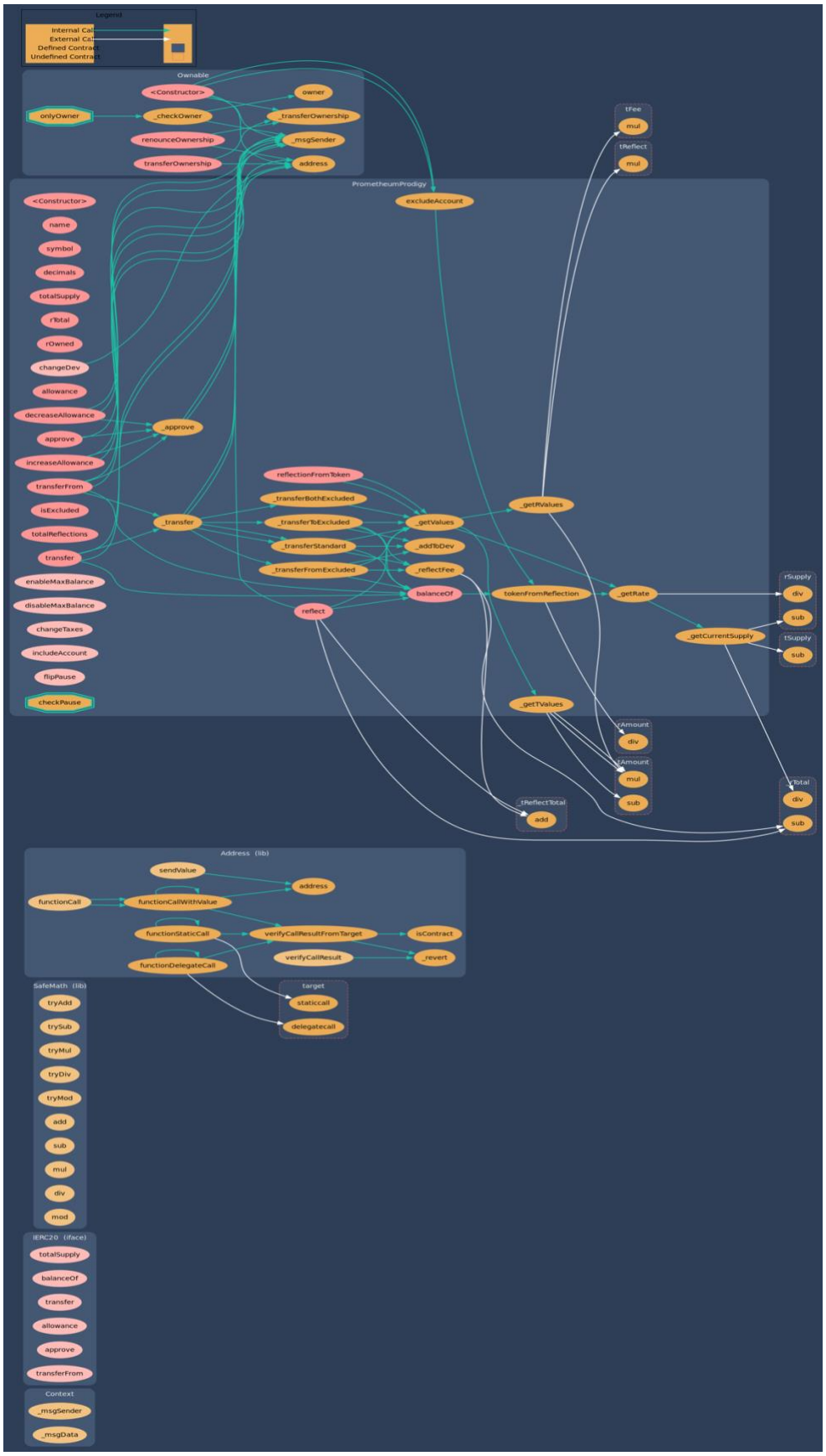
1- SOLIDITY STATIC ANALYSIS



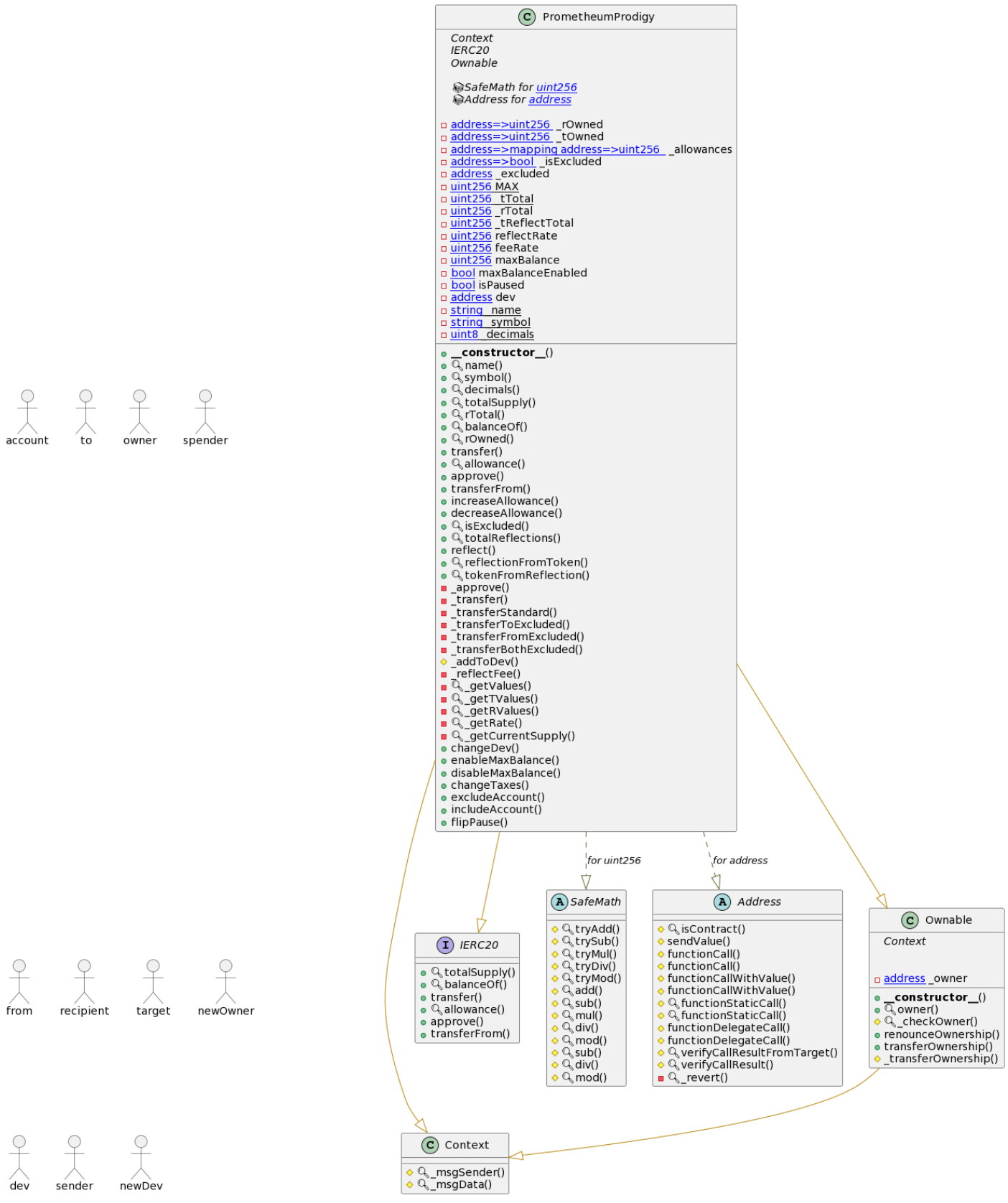
2- Inheritance graph



3- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
<Constructor>
owner
onlyOwner
_checkOwner
_transferOwnership
renounceOwnership
_msgSender
transferOwnership
address
PrometheumProdigy
excludeAccount
<Constructor>
name
symbol
decimals
totalSupply
rTotal
rOwned
changeDev
allowance
decreaseAllowance
approve
increaseAllowance
transferFrom
isExcluded
totalReflections
transfer
enableMaxBalance
disableMaxBalance
changeTaxes
includeAccount
flipPause
checkPause
_approve
_transfer
reflectionFromToken
_transferBothExcluded
_transferToExcluded
_transferStandard
_transferFromExcluded
_getValues
_addTbDev
_reflectFee
balanceOf
reflect
_getRValues
tokenFromReflection
_getRate
_getCurrentSupply
_getTValues
tFee
mul
tReflect
mul
rSupply
div
sub
tSupply
sub
rAmount
div
tAmount
mul
sub
rTotal
div
sub
tReflectTotal
add
Address (lib)
sendValue
address
functionCall
functionCallWithValue
functionStaticCall
verifyCallResultFromTarget
isContract
functionDelegateCall
verifyCallResult
_revert
target
staticcall
delegatecall
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
11902160  =>  _getTValues(uint256)
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
884557bf  =>  tryAdd(uint256,uint256)
a29962b1  =>  trySub(uint256,uint256)
6281efa4  =>  tryMul(uint256,uint256)
736ecb18  =>  tryDiv(uint256,uint256)
38dc0867  =>  tryMod(uint256,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
f43f523a  =>  mod(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
71af23e8  =>  mod(uint256,uint256,string)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
c21d36f3  =>  functionStaticCall(address,bytes)
dbc40fb9  =>  functionStaticCall(address,bytes,string)
ee33b7e2  =>  functionDelegateCall(address,bytes)
57387df0  =>  functionDelegateCall(address,bytes,string)
1daa78c1  =>  verifyCallResultFromTarget(address,bool,bytes,string)
946b5793  =>  verifyCallResult(bool,bytes,string)
6cadf5e1  =>  _revert(bytes,string)
8da5cb5b  =>  owner()
53a72975  =>  _checkOwner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
```

```
313ce567  =>  decimals()
622a69c6  =>  rTotal()
88a06f8b  =>  rOwned(address)
a457c2d7  =>  decreaseAllowance(address,uint256)
cba0e996  =>  isExcluded(address)
018763ed  =>  totalReflections()
053ab182  =>  reflect(uint256)
4549b039  =>  reflectionFromToken(uint256,bool)
2d838119  =>  tokenFromReflection(uint256)
104e81ff  =>  _approve(address,address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
2852df65  =>  _transferStandard(address,address,uint256)
16f1cc83  =>  _transferToExcluded(address,address,uint256)
c7d9be66  =>  _transferFromExcluded(address,address,uint256)
6ff6cdf4  =>  _transferBothExcluded(address,address,uint256)
c762d702  =>  _addToDev(uint256,uint256)
184d894e  =>  _reflectFee(uint256,uint256)
d4780e36  =>  _getValues(uint256)
1d5671e4  =>  _getRValues(uint256,uint256,uint256,uint256)
94e10784  =>  _getRate()
97a9d560  =>  _getCurrentSupply()
88a8c95c  =>  changeDev(address)
ebea7cda  =>  enableMaxBalance(uint256)
15232b6d  =>  disableMaxBalance()
fe2314d6  =>  changeTaxes(uint256,uint256)
f2cc0c18  =>  excludeAccount(address)
f84354f1  =>  includeAccount(address)
385df649  =>  flipPause()
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/PrometheumProdigy.sol	8e63dcf5ae9024b5e7615064ea2a7b54cf0f1b98

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	

└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	verifyCallResultFromTarget	Internal 🔒		
└	verifyCallResult	Internal 🔒		
└	_revert	Private 🔐		

Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	

PrometheumProdigy	Implementation	Context, IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	Ownable
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	rTotal	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	rOwned	Public ❗		NO❗
└	transfer	Public ❗	🛑	checkPause
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	checkPause
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	isExcluded	Public ❗		NO❗
└	totalReflections	Public ❗		NO❗
└	reflect	Public ❗	🛑	checkPause
└	reflectionFromToken	Public ❗		NO❗
└	tokenFromReflection	Public ❗		NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_transferStandard	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	
└	_transferBothExcluded	Private 🔐	🛑	
└	_addToDev	Internal 🔒	🛑	
└	_reflectFee	Private 🔐	🛑	
└	_getValues	Private 🔐		

└	_getTValues	Private 🔓		
└	_getRValues	Private 🔓		
└	_getRate	Private 🔓		
└	_getCurrentSupply	Private 🔓		
└	changeDev	External ❗	🛑	onlyOwner
└	enableMaxBalance	External ❗	🛑	onlyOwner
└	disableMaxBalance	External ❗	🛑	onlyOwner
└	changeTaxes	External ❗	🛑	onlyOwner
└	excludeAccount	Public ❗	🛑	onlyOwner
└	includeAccount	External ❗	🛑	onlyOwner
└	flipPause	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts are written systematically. Team found no high issues. So, it is good to go for production.

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 “Well Secured”.

✓ No volatile code.
✓ No 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.