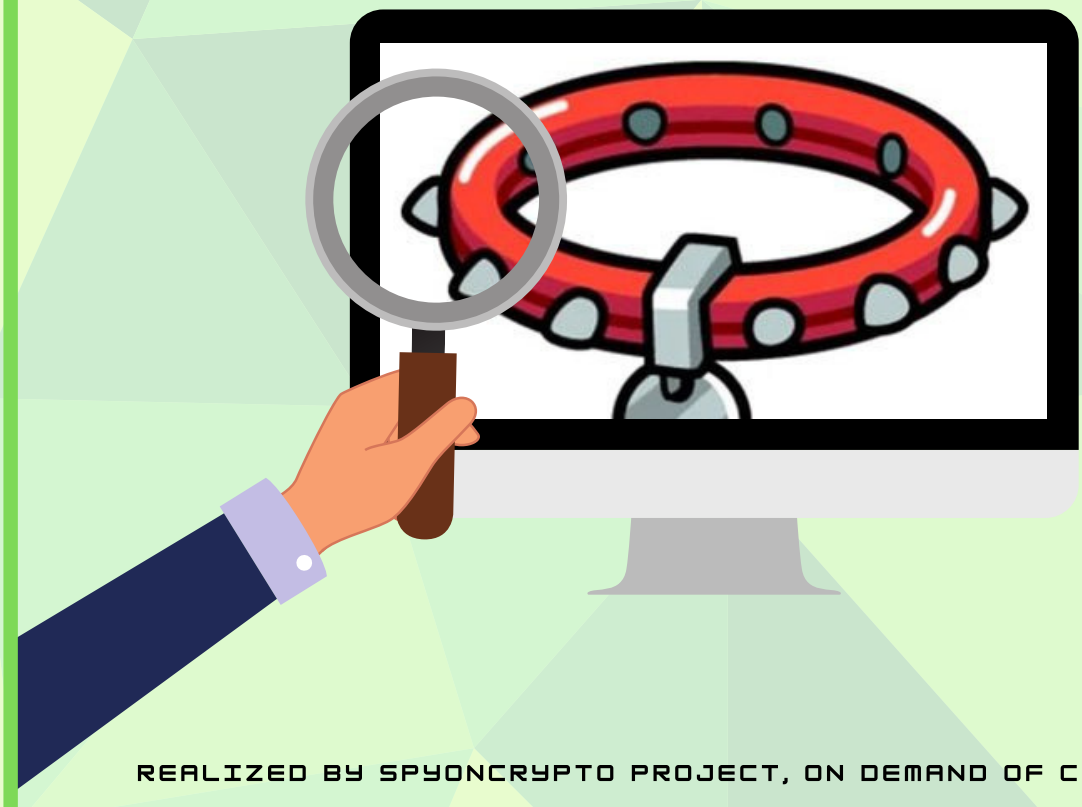
SPY ON CRYPTO

AUDIT COLLAR



REALIZED BY SPYONCRYPTO PROJECT, ON DEMAND OF COLLAR TEAM

DISCLAIMER

This file is an audit carried out at the request of the interested party.

This report is based on a multitude of analyses and research carried out by our team according to a predefined scheme.

The various steps set out in this file will make it possible to display any vulnerabilities relating to the cybersecurity of the project studied.

These searches are based on the information available to us through the smart contract, but also through information provided by the project developers.

In order to have a better overview of the possible vulnerabilities of this project, the complete reading of this file is recommended.

However, even if this report is available to you, it is only an additional element that can help you in your investigations.

Although a great deal of background work has been done in our investigations, we may have missed some elements, so further research on your part is necessary and advisable.

The conditions mentioned above in the disclaimer are not optional, so if you are not satisfied with them, we strongly urge you to stop reading and analyzing this file and to destroy any copies you have downloaded and/or printed.

These analyses and conclusions are not intended as investment advice. SpyonCrypto is not responsible for any loss of capital, which you are the only owner of.

This report is provided to you as, and without any conditions guaranteed.

SpyonCrypto disclaims any and all liability to the law for any claim or demand by you or any other person for damages.

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.

SUMMARY

PRESENTATION

No ready white paper or actual presentation of the project, or roadmap or website to present the project so far.

We have gathered various information shared by the COLLAR team on social media but no official document.

We have gathered various information shared by the COLLAR team.

COLLAR wants to deploy NFTs,

donate 10% of the fees to charity, the remaining 90% of fees will be redistributed to the CLR holder through contests and social interactions. They inform that the DEV portfolio is 10%, 15% of the supply is revoked and 70% of the LPs are blocked.

Disclaimer: Its information is shared on social networks but no official document has been published.

CONTRACT DETAILS

CONTRACT NAME
COLLAR

SUBMITTED FOR VERIFICATION AT BSCSCAN
2021-05-29

CONTRACT ADDRESS
0XDB26E2BFE43A95A32A3153FF7D15F5F022425A3F

TOTAL SUPPLY
2E+27

TOKEN TICKER
CLR

DECIMALS
18

TOKEN HOLDERS
119

TRANSACTIONS COUNT
703

TOP 100 HOLDERS DOMINANCE
99,96%

CONTRACT DEPLOYER ADDRESS
0X83B0A8BF538FEA477482D1FC32C3358E73F29E94

CONTRACT'S CURRENT OXNER ADDRESS
0X0000000000000000000000000000000000000000

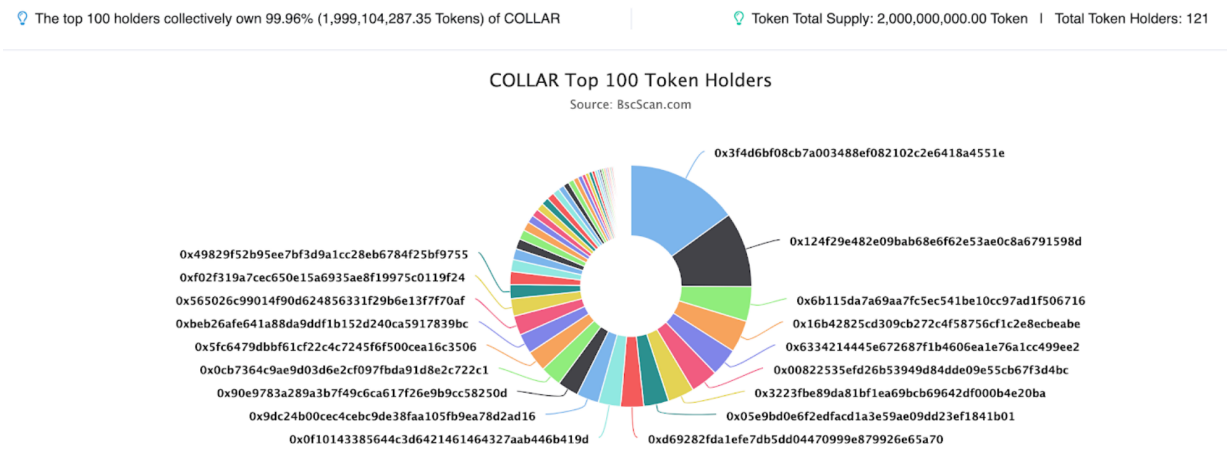
DEPLOYED AT TRANSACTION
0X8181519955AA9E8B86B29516056F917E69F4421D85FF3FCBB1F6207405BE08D1

CREATED
2021-04-27 04:28:50 UTC IN BLOCK 6920341

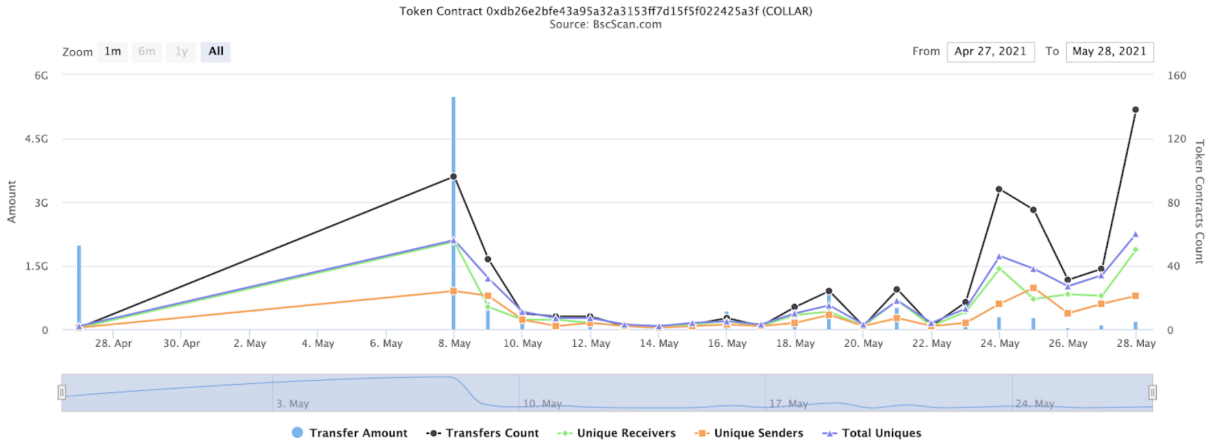
CREATOR
0XDF6FEE057222D2F7933C215C11E5150BD2EFC53E

GRAPHIC ANALYSIS

"COLLAR" Token distribution



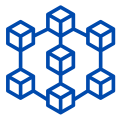
"COLLAR" contract interaction details



DETECTED VULNERABILITIES

2

0



4

SECURITY ISSUES

HIGH

1 - ***The arithmetic operator can overflow.***

It is possible to cause an integer overflow or underflow in the arithmetic operation.

```
239         name = _name;
240         symbol = _symbol;
241         decimals = _decimals;
242         totalSupply = _supply * 10**_decimals;
243         balances[tokenOwner] = totalSupply;
```

2 - ***The arithmetic operator can overflow.***

It is possible to cause an integer overflow or underflow in the arithmetic operation.

```
239         name = _name;
240         symbol = _symbol;
241         decimals = _decimals;
242         totalSupply = _supply * 10**_decimals;
243         balances[tokenOwner] = totalSupply;
244         owner = tokenOwner;
```

LOW

1 - A floating pragma is set.

The current pragma Solidity directive is ""^0.4.24"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

```
3  */
4
5  pragma solidity ^0.4.24;
6
7  library SafeMath {
```

2 - ***State variable visibility is not set.***

It is best practice to set the visibility of state variables explicitly. The default visibility for "tokenBlacklist" is internal. Other possible visibility settings are public and private.

```
120
121    mapping (address => mapping (address => uint256)) internal allowed;
122      mapping(address => bool) tokenBlacklist;
123      event Blacklist(address indexed blackListed, bool value);
124
```

3 - ***State variable visibility is not set.***

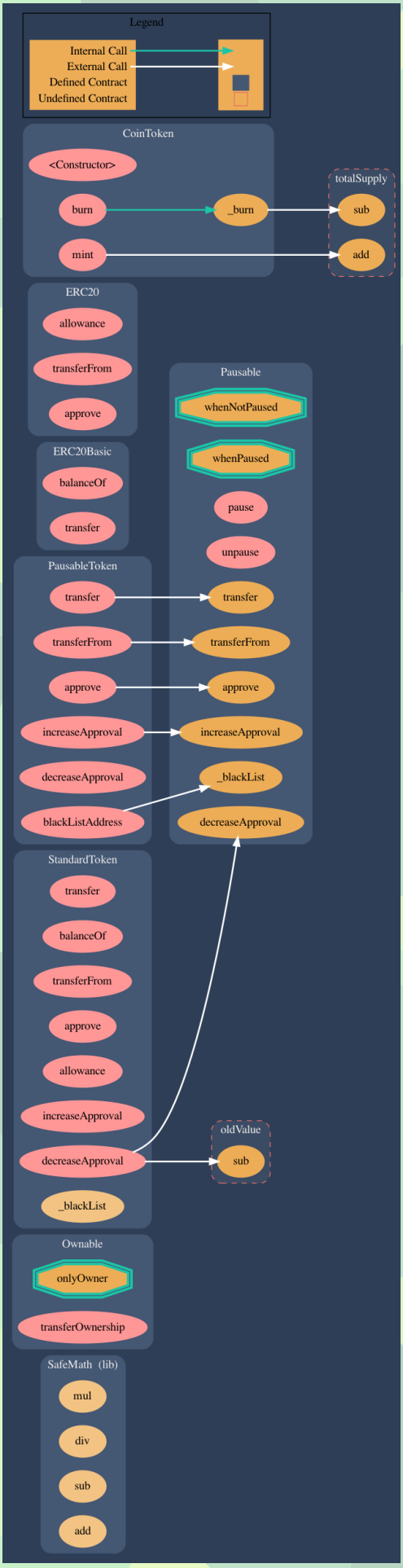
It is best practice to set the visibility of state variables explicitly. The default visibility for "balances" is internal. Other possible visibility settings are public and private.

```
124
125
126    mapping(address => uint256) balances;
127
128
```

4 - ***An assertion violation was triggered.***

It is possible to cause an assertion violation. Note that Solidity assert() statements should only be used to check invariants. Review the transaction trace generated for this issue and either make sure your program logic is correct, or use require() instead of assert() if your goal is to constrain user inputs or enforce preconditions. Remember to validate inputs from both callers (for instance, via passed arguments) and callees (for instance, via return values).

```
28
29   function add(uint256 a, uint256 b) internal pure returns (uint256) {
30     uint256 c = a + b;
31     assert(c >= a);
32     return c;
```


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
CoinToken
<Constructor>
burn
_burn
mint
totalSupply
sub
add
ERC20
allowance
transferFrom
approve
ERC20Basic
balanceOf
transfer
Pausable
whenNotPaused
whenPaused
pause
unpause
PausableToken
transfer
transferFrom
approve
increaseApproval
decreaseApproval
blackListAddress
_blackList
StandardToken
transfer
balanceOf
transferFrom
approve
allowance
increaseApproval
decreaseApproval
_blackList
oldValue
sub
Ownable
onlyOwner
transferOwnership
SafeMath (lib)
mul
div
sub
add

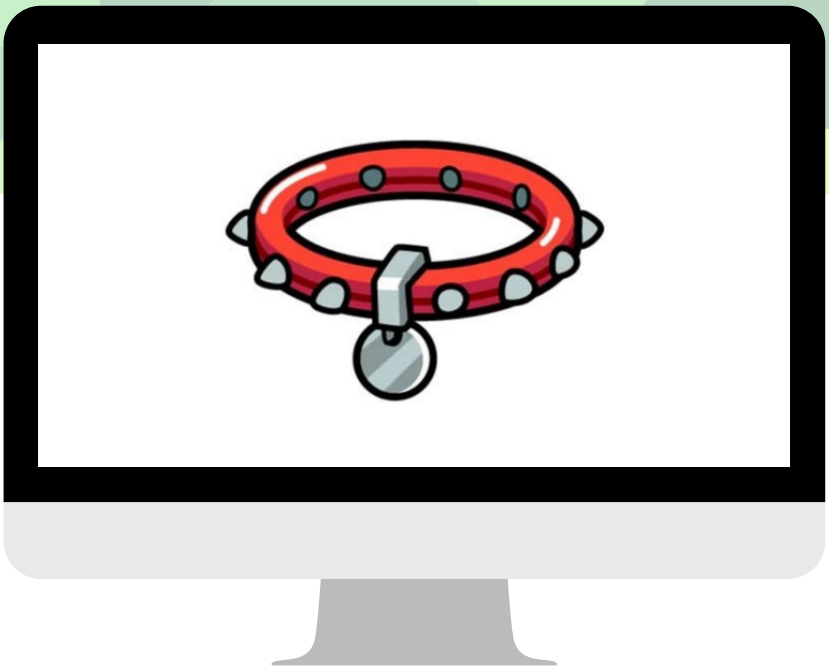
Parent	Function Name	Visibility	Mutability	Modifiers
SafeMath	////	////	////	////
	Mul	Internal		
	Div	Internal		
	Sub	Internal		
	Add	Internal		
Ownable	////	////	////	////
	TransferOwnership	Public	🛑	OnlyOwner
Pausable	////	////	////	////
	Pause	Public	🛑	OnlyOwner WhenNotPaused
	Unpause	Public	🛑	OnlyOwner WhenPaused
ERC20Basic	////	////	////	////
	BalanceOf	Public	🛑	NO❗
	Transfer	Public	🛑	NO❗
ERC20	////	////	////	////
	Allowance	Public		NO❗
	TransferFrom	Public	🛑	NO❗
	Approve	Public	🛑	NO❗
StandardToken	////	////	////	////
	Transfer	Public	🛑	NO❗
	BalanceOf	Public		NO❗
	TransferFrom	Public	🛑	NO❗
	Approve	Public	🛑	NO❗
	Allowance	Public		NO❗
	IncreaseApproval	Public	🛑	NO❗
	DecreaseApproval	Public	🛑	NO❗
	_Blacklist	Internal	🛑	
PausableToken	////	////	////	////
	Transfer	Public	🛑	WhenNotPaused
	TransferFrom	Public	🛑	WhenNotPaused
	Approve	Public	🛑	WhenNotPaused
	IncreaseApproval	Public	🛑	WhenNotPaused
	DecreaseApproval	Public	🛑	WhenNotPaused
	BlacklistAddress	Public	🛑	WhenNotPaused OnlyOwner
CoinToken	////	////	////	////
	<Constructor>	Public	🛑	NO❗
	Burn	Public	🛑	NO❗
	_Burn	Internal	🛑	
	Mint	Public	🛑	OnlyOwner

🔸 = Function can modify state

LOCATION TEAM

TEAM USA (TEXAS)

SOCIAL MEDIA



https://twitter.com/collartoken

https://t.me/collartoken

https://www.instagram.com/deficollar/

https://www.tiktok.com/@collartoken

https://www.snapchat.com/add/collartoken

NOTE AND CONCLUSION

In conclusion, the SPYON crypto's team found some functions that may be study again.
Here are all those that we find vulnerable and that should be improved :

First of all the mint function :
The possibility to increase the supply by the owner which is for us something that can conduct to a dump risk.

In a second time the pause function :
possibility to add pause in contract transfer as the transfers by the owner. For the team it's a function with a high risk.

In a third part the Blacklist : possible blacklist address. Medium risk.

Then the transfer ownership : you have the possibility to change the owner contract. This conduct to a medium risk according to the team

To finish the underflow : using safemath, you can, by an other person, exploit the contract looking to the graph functions.

In conclusion we can say that $CLR isn't safe for mains reasons :

- the contract wasn't renounced
- the MINT function
- the PAUSE function
- the BLACKLIST function
- the TRANSFERT OWNERSHIP function
- OVERFLOW or UNDERFLOW risks

TG : SPYONCRYPTO
TG CHAT: SPYONCRYPTO CHAT
TWITTER: @spyoncrypto
MAIL: SOCIAL@SPYONCRYPTO.COM
SITE WEB: WWW.SPYONCRYPTO.COM