

Smart Contract Security Audit V1

Emergency Timelock Controller Smart Contract Audit

Jun 24, 2025



<https://saferico.com/>

business@saferico.com

https://t.me/SFI_ANN

—

Table of Contents

Table of Contents

Background

Project Information

Smart Contract Information

Executive Summary

File and Function Level Report

File in Scope:

Issues Checking Status

SWC Attack Analysis

Severity Definitions

Audit Findings

Automatic testing

Testing proves

Inheritance graph

Call graph

Source lines

Risk level

Source units in scope

Capabilities

Unified Modeling Language (UML)

Functions signature

Automatic general report

Conclusion

Disclaimer

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

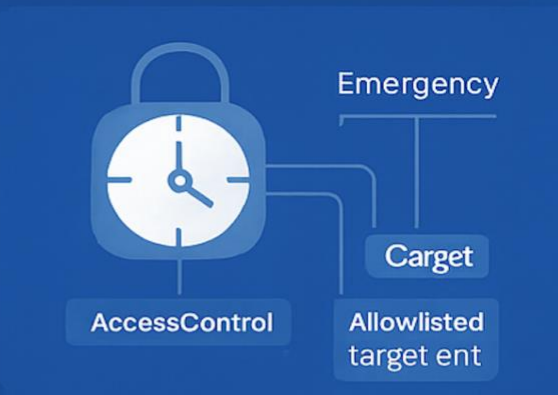
- **Platform:** Binance Smart Chain
- **Name:** EmergencyTimelockController
- **Language :** Solidity
- **Contract Address:** 0x37738e8cfaecc315407b48a4799bb4f8b28e1e26, 0x52468d6a9abbf4e72bbf2603ceed882b5c3b9157
- **Code Source:** <https://github.com/TerraDharitri/drt-pREWA/tree/main/contracts>

What is EmergencyTimelockController?

- Manages privileged emergency actions with a timelock delay for
- Actions executable, cancel actions on allowlisted contracts after a lowlisted period
- Upgradable contract with role-based access control



Ensures critical actions are reviewed before



Ensures critical actions are reviewed before execution

Key Features



Propose Actions

EMERGENCY_ROLE holders propose actions on allowlisted contracts



Timelock Delay

Actions executable only after 1 hour to 7d delay



Execute Actions

Executes approved actions via low-level calls to target contracts



Cancel Actions

Allows cancellation of proposed action before execution

Timelock ensures transparency and security for critical actions

How Emergency Actions Work



ReentrancyGuard

Prevents reentrancy attacks



AccessControl

Restricts actions to EMERGENCY_ROLE holders



Allowlist

Only approved contracts and function selectors can



Timelock

Delays execution for review (1 hour to 7 days)



Upgradeability

Reserved storage (._gap) for future upgrades

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, 0 low, 0 very low-level issues and 2 notes in all solidity files of the contract

The files:

EmergencyTimelockController.sol

Audit Score:

100% secure



File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
EmergencyTimelockController.sol	94735e88ad7f750ead488b7ad618b1486cbdb483	0x37738e8cfaecc315407b48a4799bb4f8b28e1e26,0x52468d6a9abbf4e72bbf2603ceed882b5c3b9157

- Contract: EmergencyTimelockController
- Inherit: Initializable, ReentrancyGuardUpgradeable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
accessControl	✓	Read / public	Passed
allActionIds	✓	Read / public	Passed
allowedFunctionSelectors	✓	Read / public	Passed
allowedTargets	✓	Read / public	Passed
emergencyActions	✓	Read / public	Passed
getActionDetails	✓	Read / public	Passed
getActionStatus	✓	Read / public	Passed
getAllActionIds	✓	Read / public	Passed
isFunctionSelectorAllowed	✓	Read / public	Passed
isTargetAllowed	✓	Read / public	Passed
MAX_TIMELOCK_DURATION	✓	Read / public	Passed
MIN_TIMELOCK_DURATION	✓	Read / public	Passed
timelockDuration	✓	Read / public	Passed

cancelEmergencyAction	✓	Write / public	Passed
executeEmergencyAction	✓	Write / public	Passed
proposeEmergencyAction	✓	Write / public	Passed
setAllowedFunctionSelector	✓	Write / public	Passed
initialize	✓	Write / public	Passed
setAllowedTarget	✓	Write / public	Passed
updateTimelockDuration	✓	Write / public	Passed

Issues Checking Status

SWC Attack Analysis

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) for more info check

<https://swcregistry.io/>

No.	Issue Description	Checking Status
136	Unencrypted Private Data On-Chain	Passed
135	Code With No Effects	Passed
134	Message call with hardcoded gas amount	Passed
133	Hash Collisions With Multiple Variable Length Arguments	Passed
132	Unexpected Ether balance	Passed
131	Presence of unused variables	Passed
130	Right-To-Left-Override control character (U+202E)	Passed
129	Typographical Error	Passed
128	DoS with block gas limit.	Passed
127	Arbitrary Jump with Function Type Variable	Passed
126	Insufficient Gas Griefing	Passed
125	Incorrect Inheritance Order	Passed
124	Write to Arbitrary Storage Location	Passed
123	Requirement Violation	Passed
122	Lack of Proper Signature Verification	Passed
121	Missing Protection against Signature Replay Attacks	Passed
120	Weak Sources of Randomness from Chain Attributes	Passed
119	Shadowing State Variables	Passed

118	Incorrect Constructor Name	Passed
117	Signature Malleability	Passed
116	Block values as a proxy for time	Not Passed
115	Authorization through tx.origin	Passed
114	Transaction Order Dependence	Passed
113	DoS with Failed Call	Passed
112	Delegatecall to Untrusted Callee	Passed
111	Use of Deprecated Solidity Functions	Passed
110	Assert Violation	Passed
109	Uninitialized Storage Pointer	Passed
108	State Variable Default Visibility	Passed
107	Reentrancy	Passed
106	Unprotected SELFDESTRUCT Instruction	Passed
105	Unprotected Ether Withdrawal	Passed
104	Unchecked Call Return Value	Passed
103	Floating Pragma	Not Passed
102	Outdated Compiler Version	Passed
101	Integer Overflow and Underflow	Passed
100	Function Default Visibility	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 Low severity vulnerabilities were found.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.28 instead of ^0.8.28). 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. And avoid Solidity compiler Bugs check here

<https://sepolia.etherscan.io/solcbuginfo>

Remediation

Remove the ^ sign to lock the pragma version.

Use of block.timestamp for comparisons

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

```
if (!allowedFunctionSelectors[functionSelector]) revert  
NotAuthorized(bytes32(0));
```

```
        actionId = keccak256(abi.encodePacked(block.timestamp,  
emergencyLevel, msg.sender, target, data));
```

```
EmergencyAction storage existingAction =  
emergencyActions[actionId];  
    if (existingAction.proposalTime != 0) revert  
InvalidAmount();
```

Recommendation

Avoid use of `block.timestamp`.

Automatic Testing

1- 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

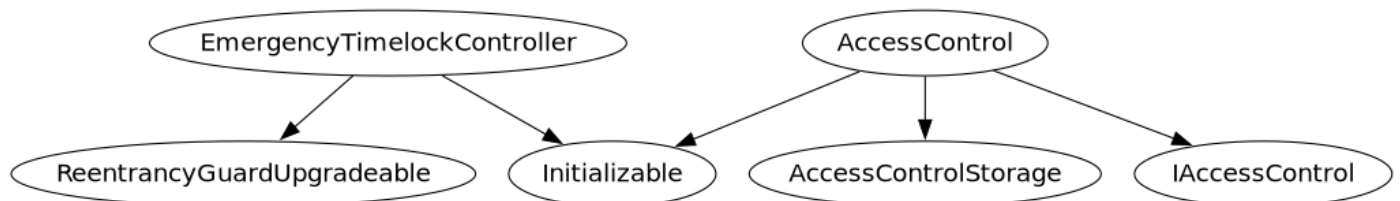
- ☒ **ERC20:**
'decimals' should be 'uint8'

Miscellaneous

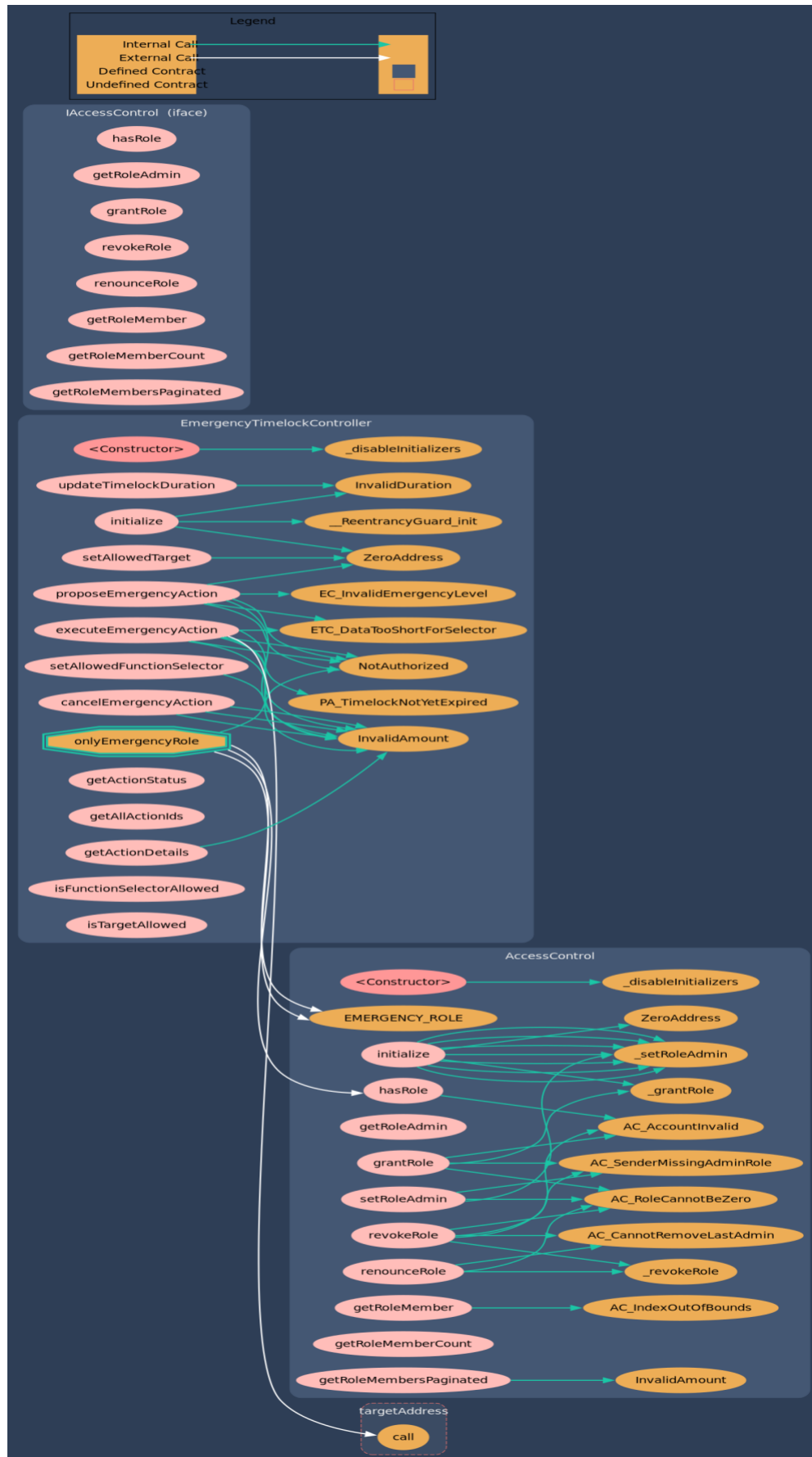
☒ Select Miscellaneous

- ☒ **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

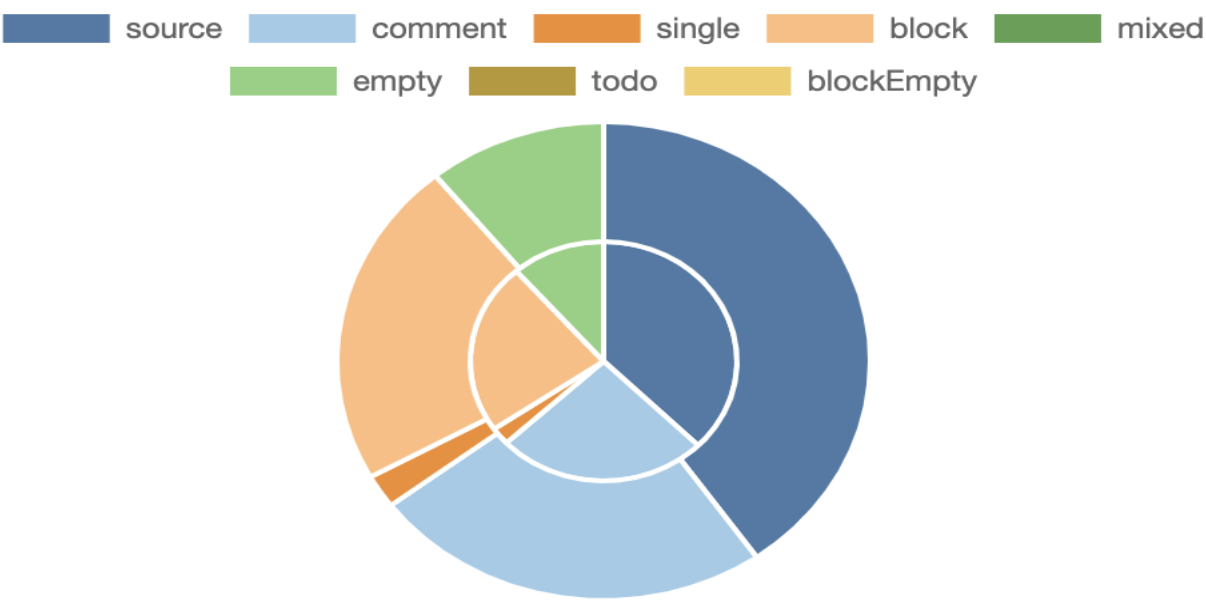
2- Inheritance graph



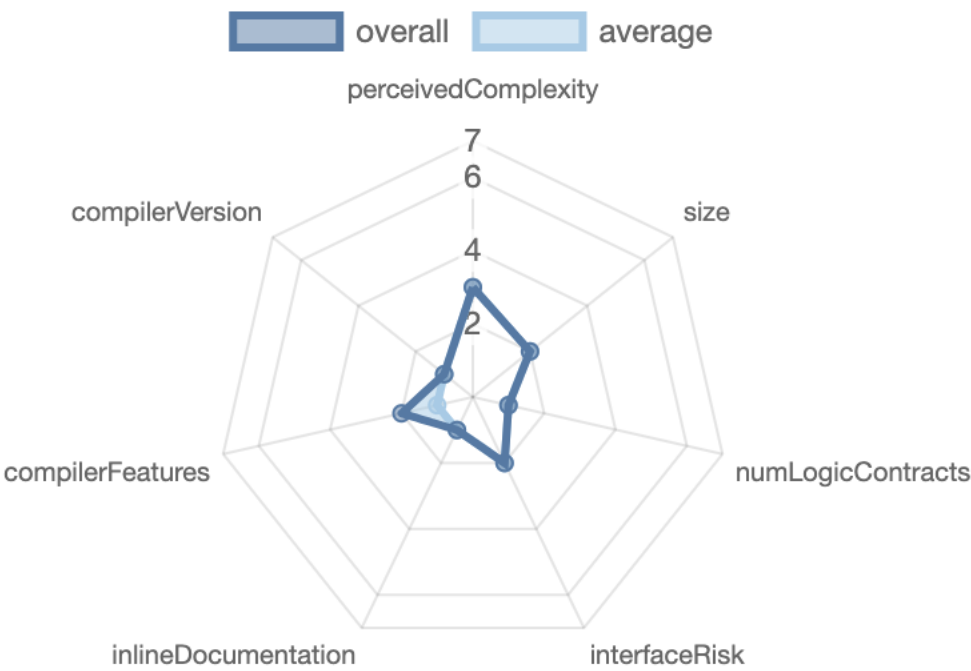
3- Call graph



Source lines



Risk level



Source units in scope

Source Units in Scope

Source Units Analyzed: 1
Source Units in Scope: 1 (100%)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/controllers/EmergencyTimelockController.sol	1	————	501	471	237	162	155	
	Totals	1	————	501	471	237	162	155	

Legend: [—]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Capabilities

Components

 Contracts	 Libraries	 Interfaces	 Abstract
1	0	0	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable
12	0

External	Internal	Private	Pure	View
12	8	0	0	5

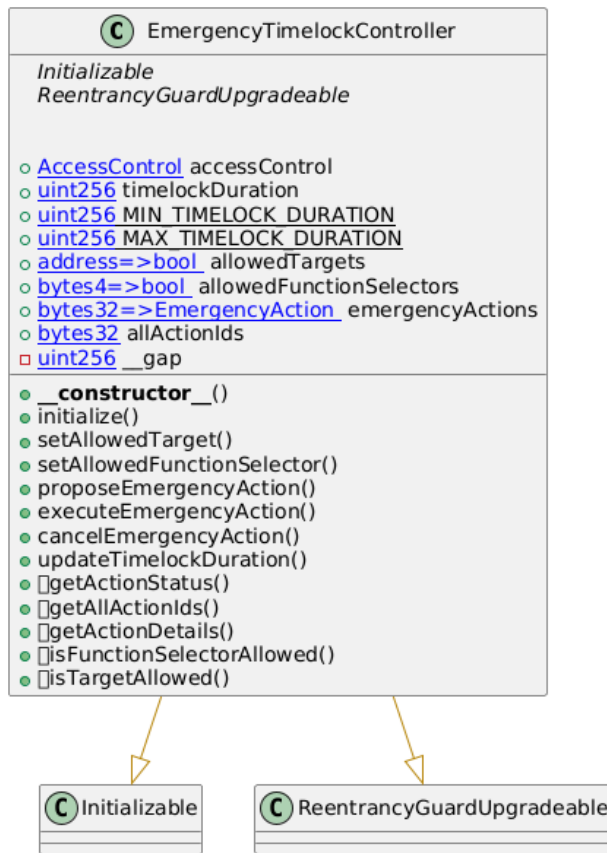
StateVariables

Total	 Public
9	8

Capabilities

Solidity Versions observed	✔ Experimental Features	💰 Can Receive Funds	💻 Uses Assembly	🌐 Has Destroyable Contracts	
0.8.28			yes (3 asm blocks)		
📡 Transfers ETH	⚡ Low-Level Calls	👤 DelegateCall	📦 Uses Hash Functions	📄 ECREcover	🌀 New/Create/Create2
			yes		

Unified Modeling Language (UML)



Functions signature

Function Name	Sighash	Function Signature
initialize	cd6dc687	initialize(address,uint256)
setAllowedTarget	caldd22e	setAllowedTarget(address,bool)
setAllowedFunctionSelector	30a18268	setAllowedFunctionSelector(bytes4,bool)
proposeEmergencyAction	89ff0ff2	proposeEmergencyAction(uint8,address,bytes)
executeEmergencyAction	74cd33e6	executeEmergencyAction(bytes32)
cancelEmergencyAction	dd5b7623	cancelEmergencyAction(bytes32)
updateTimelockDuration	184466c9	updateTimelockDuration(uint256)
getActionStatus	bea0897c	getActionStatus(bytes32)
getAllActionIds	7266948f	getAllActionIds()
getActionDetails	cd00c3e1	getActionDetails(bytes32)
isFunctionSelectorAllowed	c745082e	isFunctionSelectorAllowed(bytes4)
isTargetAllowed	5309fa47	isTargetAllowed(address)

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/drt-pREWA/contracts/controllers/EmergencyTimelockController.sol	94735e88ad7f750ead488b7ad618b1486cbdb483
/Users/macbook/Desktop/drt-pREWA/contracts/access/AccessControl.sol	40163fa249f10365c03cf1fa9ddd26e2f6eb1005
/Users/macbook/Desktop/drt-pREWA/contracts/access/storage/AccessControlStorage.sol	2a1f4c3d6956a89011b090a62b15254b1d720d65
/Users/macbook/Desktop/drt-pREWA/contracts/access/interfaces/IAccessControl.sol	540ec54eaadelc05a650af086ebf959654ec6322
/Users/macbook/Desktop/drt-pREWA/contracts/libraries/Errors.sol	0974f6f49e3b655fa93a2792154f1b506ec03c74

Contracts Description Table

Contract	Type	Bases
L	**Function Name**	**Visibility**
Modifiers		**Mutability**
EmergencyTimelockController	Implementation	Initializable, ReentrancyGuardUpgradeable
L <Constructor>	Public !	! NO !
L initialize	External !	! initializer
L setAllowedTarget	External !	! onlyEmergencyRole
L setAllowedFunctionSelector	External !	! onlyEmergencyRole
L proposeEmergencyAction	External !	! onlyEmergencyRole nonReentrant
L executeEmergencyAction	External !	! onlyEmergencyRole nonReentrant
L cancelEmergencyAction	External !	! onlyEmergencyRole nonReentrant
L updateTimelockDuration	External !	! onlyEmergencyRole
L getActionStatus	External !	! NO !
L getAllActionIds	External !	! NO !
L getActionDetails	External !	! NO !
L isFunctionSelectorAllowed	External !	! NO !

L	isTargetAllowed	External	!		NO!	
AccessControl Implementation Initializable,						
AccessControlStorage, IAccessControl						
L	<Constructor>	Public	!	⬢	NO!	
L	initialize	External	!	⬢		initializer
L	hasRole	External	!		NO!	
L	getRoleAdmin	External	!		NO!	
L	grantRole	External	!	⬢	NO!	
L	revokeRole	External	!	⬢	NO!	
L	renounceRole	External	!	⬢	NO!	
L	setRoleAdmin	External	!	⬢	NO!	
L	getRoleMember	External	!		NO!	
L	getRoleMemberCount	External	!		NO!	
L	getRoleMembersPaginated	External	!		NO!	
L	_grantRole	Internal	🔒	⬢		
L	_revokeRole	Internal	🔒	⬢		
L	_setRoleAdmin	Internal	🔒	⬢		
AccessControlStorage Implementation						
IAccessControl Interface						
L	hasRole	External	!		NO!	
L	getRoleAdmin	External	!		NO!	
L	grantRole	External	!	⬢	NO!	
L	revokeRole	External	!	⬢	NO!	
L	renounceRole	External	!	⬢	NO!	
L	getRoleMember	External	!		NO!	
L	getRoleMemberCount	External	!		NO!	
L	getRoleMembersPaginated	External	!		NO!	

Legend

Symbol	Meaning
:-----:	-----
⬢	Function can modify state
🔒	Function is payable

Conclusion

The contracts are written systematically. Team found no critical 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.