PUBLIC ACCESS

CYBERSECURITY AUDIT REPORT

Version v1.2

This document details the process and results of the smart contract audit performed independently by Electropact from 01/08/2025 to 05/08/2025.

Audited for

SM9 (Samsy Capital)

Audited by

Team Electropact

https://electropact.live
info@electropact.live

Contents

Disclaimer

Smart Contract Audit only provides findings and recommendations for an exact commitment of a smart contract codebase. The results, hence, is not guaranteed to be accurate outside of the commitment, or after any changes or modifications made to the codebase. The evaluation result does not guarantee the nonexistence of any further findings of security issues.

Time-limited engagements do not allow for a comprehensive evaluation of all security controls, so this audit does not give any warranties on finding all possible security issues of the given smart contract(s). Electro-Pact prioritized the assessment to identify the weakest security controls an attacker would exploit.We recommend other token conducting similar assessments on an annual basis by internal, third-party assessors, or a public bug bounty program to ensure the security of smart contract(s).

This security audit should never be used as an investment advice.

Version History

Version	Date	Release notes
1.0	01/08/2025	The first report was sent to the client. All findings were in the open status.
1.1	05/08/2025	All findings are accepted and resolved in the new GitHub commit.
1.2	06/08/2025	<SM9> 0x7307dA39F48029eAC35d6a74a249Ff2c0DfF7F72 allowed Electro-Pact to publish the auditreport publicly.

Auditors

Fullname	Role	Email address
Nikolai	Head of Security	info@electropact.live nikolai@electropact.live

Introduction

From 01/08/2025 to 05/08/2025, Electro-Pact to evaluate the security postureof contract system. Our findings and recommendations are detailed here in this initial report.

The report will be continually updated to correctly reflect the mitigation and remediation state of each finding.

1.1 Audit Details

Audit Target

The SM9 connectedwith blockchain technology and based on BEP20 Token. The total supply of 99,000,000,000tokens Each Token is based on Tron based Reputation System, and having following information for public.

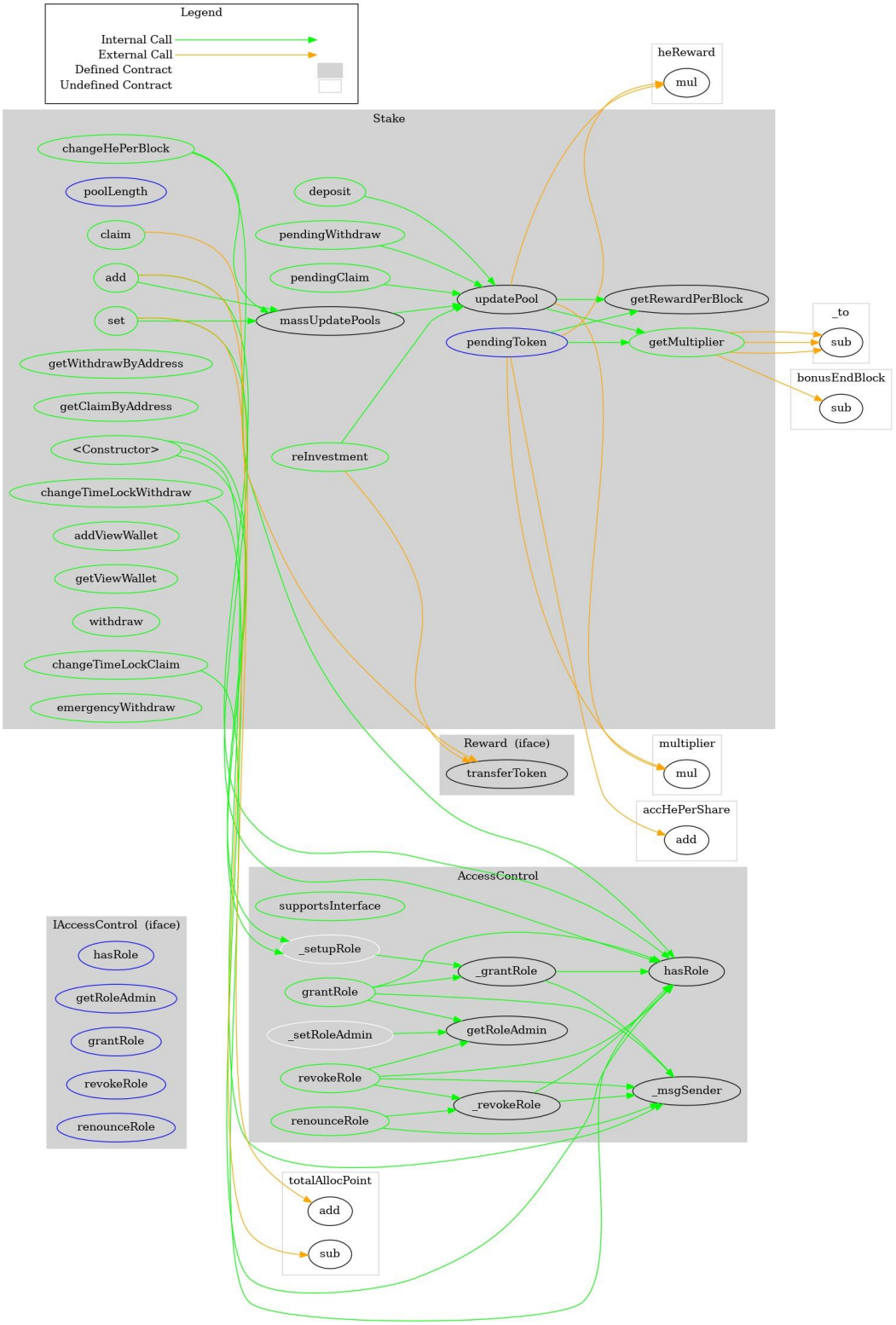
According to the roadmap of SM9 , in August 2025, staking functions for token will be released. To ensure the safety of every customers' assets, SM9 have requested asecurity assessment on the related file SamsyToken.sol.

The basic information of this file is as follows:

Item	Description
Project Name	SamsyToken (SM9)
Issuer	https://samsycapital.com - 0x7307dA39F48029eAC35d6a74a249Ff2c0DfF7F72
Website	https://samsycapital.com / https://samsycapital.tech
Platform	Smart Contract
Language	Solidity
Commit	0x7307dA39F48029eAC35d6a74a249Ff2c0DfF7F72
Audit method	Whitebox

With the contract SamsyToken.sol, users can receive rewards after certain amount of time, corresponding to the HE staking policy, by supplying into the pools a proper quantity of HE tokens. Users can claim all the rewards, withdraw their supply from the staking pools and make reinvestments to these pools anytime. The pools can only be initialized by users with the administration role (DEFAULT_ADMIN_ROLE).

The architecture of SamsyToken.sol is illustrated in the following graph:



Audit Service Provider

Electro-Pact is a leading security company in USA with the goal of building the next generation of cybersecurity solutions to protect businesses against threats from the Internet.

1.2 Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient and working according to its specifications. The audit activities can be grouped in the following three categories:

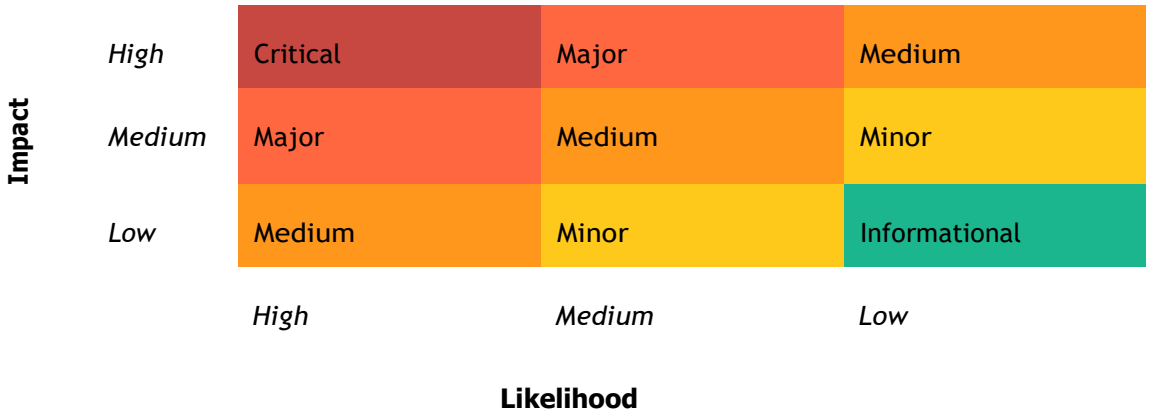
1. **Security:** Identifying security related issues within each contract and within the system of contracts.
2. **Sound Architecture:** Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. **Code Correctness and Quality:** A full review of the contract source code. The primary areas of focus include:
    - Correctness
    - Readability
    - Sections of code with high complexity
    - Improving scalability
    - Quantity and quality of test coverage

1.3 Audit Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology:

- **Likelihood** represents how likely a particular vulnerability is to be uncovered and exploited in the wild;
- **Impact** measures the technical loss and business damage of a successful attack;
- **Severity** demonstrates the overall criticality of the risk.

Likelihood and impact are categorized into three ratings: High, Medium and Low, i.e., H, M and L respectively. Severity is determined by likelihood and impact and can be classified into four categories accordingly, i.e., Critical, Major, Medium, Minor and Informational (Info) as the table below:



Electro-Pact firstly analyses the smart contract with open-source and also our own security assessment tools to identify basic bugs related to general smart contracts. These tools include Slither, securify,Mythril, Sūrya, Solgraph, Truffle, Geth, Ganache, Mist, Metamask, solhint, mythx, etc. Then, our security specialists will verify the tool results manually, make a description and decide the severity for each of them.

After that, we go through a checklist of possible issues that could not be detected with automatic tools, conduct test cases for each and indicate the severity level for the results. If no issues are found after manual analysis, the contract can be considered safe within the test case. Else, if any issues are found, we might further deploy contracts on our private testnet and run tests to confirm the findings. We would additionally build a PoC to demonstrate the possibility of exploitation, if required or necessary.

The standard checklist, which applies for every SCA, strictly follows the Smart Contract Weakness Classification Registry (SWC Registry). SWC Registry is an implementation of the weakness classification scheme proposed in The Ethereum Improvement Proposal project under the code EIP-1470. The checklist of testing according to SWC Registry is shown in Appendix A.

In general, the auditing process focuses on detecting and verifying the existence of the following issues:

- **Coding Specification Issues:** Focusing on identifying coding bugs related to general smart contract coding conventions and practices.
- **Design Defect Issues:** Reviewing the architecture design of the smart contract(s) and working on test cases, such as self-DoS attacks, incorrect inheritance implementations, etc.
- **Coding Security Issues:** Finding common security issues of the smart contract(s), for example integer overflows, insufficient verification of authenticity, improper use of cryptographic signature, etc.
- **Coding Design Issues:** Testing the code logic and error handlings in the smart contract code base, such as initializing contract variables, controlling the balance and flows of token transfers, verifying strong randomness, etc.
- **Coding Hidden Dangers:** Working on special issues, such as data privacy, data reliability, gas consumption optimization, special cases of authentication and owner permission, fallback functions, etc.

For better understanding of found issues' details and severity, each SWC ID is mapped to the most closely related Common Weakness Enumeration (CWE) ID. CWE is a category system for software weaknesses and vulnerabilities to help identify weaknesses surrounding software jargon. The list in Appendix B provides an overview on specific similar software bugs that occur in Smart Contract coding.

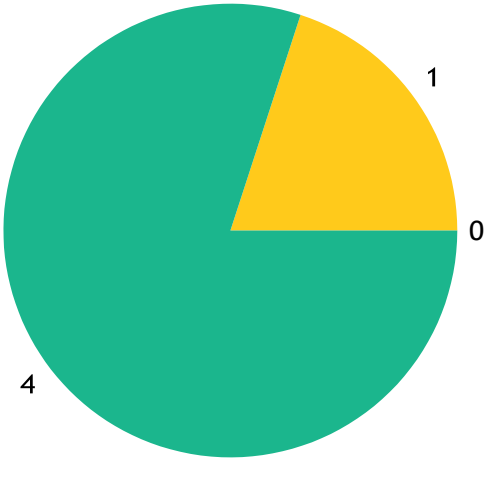
The final report will be sent to the smart contract issuer with an executive summary for overview and detailed results for acts of remediation.

1.4 Audit Scope

Assessment	Target	Type
White-box testing	SamsyToken.sol	Solidity code file

Executive Summary

Security issues by severity



Legend

- Critical
- Major
- Medium
- Minor
- Info

Security issues by SWC

Function Default Visibility (SWC-100)	1	
Floating Pragma (SWC-103)	1	
Use of Deprecated Solidity Functions (SWC-111)	1	
Requirement Violation (SWC-123)	1	
Code With No Effects (SWC-135)	1	

Security issues by CWE

Use of Obsolete Function (CWE-477)	1
Improper Following of Specification by Caller (CWE-573)	1
Improper Control of a Resource Through its Lifetime (CWE-664)	1
Improper Adherence to Coding Standards (CWE-710)	1
Irrelevant Code (CWE-1164)	1

Table of security issues

ID	Status	Vulnerability	Severity
#hne-001	Resolved	Floating pragma	INFO
#hne-002	Resolved	Code with no effects *owner*	INFO
#hne-003	Resolved	Inefficient function declarations	INFO
#hne-004	Resolved	Ignored constructor visibility	INFO
#hne-005	Resolved	Requirements on always-true conditions	MINOR

Recommendations

Based on the results of this smart contract audit, Electro-Pact has the following high-level key recommendations:

Key recommendations	
Issues	Electro-Pact conducted a security assessment of smart contracts for SM9. No issues with severity higher than low had been found. Theseissues do not represent actual bugs or security problems. After SM9 committed the new codebase for staking functions on GitHub, Electro-Pact produced the re-test and confirmed that all issues were resolved.
Recommendations	Electro-Pact recommends SM9 to evaluate the audit results withseveral different security audit third-parties for the most accurate conclusion.

Detailed Results

1. Floating pragma

Issue ID	#hne-001
Category	SWC-103 - Floating Pragma
Description	Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.
Severity	INFO
Location(s)	SamsyToken.sol
Status	Resolved
Reference	CWE-664 - Improper Control of a Resource Through its Lifetime
Remediation	Lock the pragma version and also consider known bugs (BEP-20 Token | Address: 0x7307da39...c0dff7f72 | BscScan) for the compiler version that ischosen.

Description

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Pausable.sol";
import "@openzeppelin/contracts/utils/ReentrancyGuard.sol";

/**
 * @title SamsyCapitalToken
 * @dev Enhanced ERC20 token with security features and proper access control
 */
contract SamsyCapitalToken is ERC20, Ownable(msg.sender), Pausable, ReentrancyGuard {

    // Constants
    uint256 public constant TOTAL_SUPPLY = 99_000_000_000 * 10**18; // 99 billion tokens
total (99 crore)

    // State variables
```

```
mapping(address => bool) public isWhitelisted;
bool public transferLocked = true;

// Events
event WhitelistUpdated(address indexed account, bool status);
event TransfersUnlocked();
event TokensBurned(address indexed account, uint256 amount);

// Errors
error TransfersLocked();
error ZeroAddress();
error ZeroAmount();
error NotWhitelisted();

modifier onlyWhenTransfersAllowed(address from, address to) {
    if (transferLocked && !isWhitelisted[from] && !isWhitelisted[to]) {
        revert TransfersLocked();
    }
    _;
}

modifier validAddress(address addr) {
    if (addr == address(0)) {
        revert ZeroAddress();
    }
    _;
}

modifier validAmount(uint256 amount) {
    if (amount == 0) {
        revert ZeroAmount();
    }
    _;
}

constructor() ERC20("Samsy Capital", "SM9") {
    _mint(msg.sender, TOTAL_SUPPLY);

    // Whitelist the owner initially
    isWhitelisted[msg.sender] = true;
    emit WhitelistUpdated(msg.sender, true);
}

/**
 * @dev Override transfer to include transfer lock mechanism
 */
function transfer(address to, uint256 amount)
    public
    override
    whenNotPaused
```

```
    nonReentrant
    onlyWhenTransfersAllowed(msg.sender, to)
    validAddress(to)
    validAmount(amount)
    returns (bool)
{
    return super.transfer(to, amount);
}

/**
 * @dev Override transferFrom to include transfer lock mechanism
 */
function transferFrom(address from, address to, uint256 amount)
    public
    override
    whenNotPaused
    nonReentrant
    onlyWhenTransfersAllowed(from, to)
    validAddress(to)
    validAmount(amount)
    returns (bool)
{
    return super.transferFrom(from, to, amount);
}

/**
 * @dev Unlock transfers permanently - can only be called once
 */
function unlockTransfers() external onlyOwner {
    require(transferLocked, "Transfers already unlocked");
    transferLocked = false;
    emit TransfersUnlocked();
}

/**
 * @dev Update whitelist status for an address
 */
function whitelistAddress(address addr, bool status)
    external
    onlyOwner
    validAddress(addr)
{
    isWhitelisted[addr] = status;
    emit WhitelistUpdated(addr, status);
}

/**
 * @dev Batch whitelist multiple addresses
 */
function batchWhitelist(address[] calldata addresses, bool status)
```

```
    external
    onlyOwner
{
    for (uint256 i = 0; i < addresses.length; i++) {
        if (addresses[i] != address(0)) {
            isWhitelisted[addresses[i]] = status;
            emit WhitelistUpdated(addresses[i], status);
        }
    }
}

/**
 * @dev Burn tokens from caller's balance
 */
function burn(uint256 amount)
    external
    validAmount(amount)
{
    _burn(msg.sender, amount);
    emit TokensBurned(msg.sender, amount);
}

/**
 * @dev Burn tokens from specified address (requires allowance)
 */
function burnFrom(address account, uint256 amount)
    external
    validAddress(account)
    validAmount(amount)
{
    _spendAllowance(account, msg.sender, amount);
    _burn(account, amount);
    emit TokensBurned(account, amount);
}

/**
 * @dev Emergency pause function
 */
function pause() external onlyOwner {
    _pause();
}

/**
 * @dev Unpause function
 */
function unpause() external onlyOwner {
    _unpause();
}

/**
```

```
     * @dev Check if an address can transfer tokens
     */
    function canTransfer(address from, address to) external view returns (bool) {
        return !transferLocked || isWhitelisted[from] || isWhitelisted[to];
    }

    /**
     * @dev Get contract information
     */
    function getContractInfo() external view returns (
        uint256 totalTokenSupply,
        uint256 currentSupply,
        bool transfersLocked,
        bool contractPaused
    ) {
        return (
            TOTAL_SUPPLY,
            totalSupply(),
            transferLocked,
            paused()
        );
    }

}
```

Contract ABI

[{"inputs":[],"stateMutability":"nonpayable","type":"constructor"},{"inputs":[{"internalType":"address","name":"spender","type":"address"},{"internalType":"uint256","name":"allowance","type":"uint256"},{"internalType":"uint256","name":"needed","type":"uint256"}],"name":"ERC20InsufficientAllowance","type":"error"},{"inputs":[{"internalType":"address","name":"sender","type":"address"},{"internalType":"uint256","name":"balance","type":"uint256"},{"internalType":"uint256","name":"needed","type":"uint256"}],"name":"ERC20InsufficientBalance","type":"error"},{"inputs":[{"internalType":"address","name":"approver","type":"address"}],"name":"ERC20InvalidApprover","type":"error"},{"inputs":[{"internalType":"address","name":"receiver","type":"address"}],"name":"ERC20InvalidReceiver","type":"error"},{"inputs":[{"internalType":"address","name":"sender","type":"address"}],"name":"ERC20InvalidSender","type":"error"},{"inputs":[{"internalType":"address","name":"spender","type":"address"}],"name":"ERC20InvalidSpender","type":"error"},{"inputs":[],"name":"EnforcedPause","type":"error"},{"inputs":[],"name":"ExpectedPause","type":"error"},{"inputs":[],"name":"NotWhitelisted","type":"error"},{"inputs":[{"internalType":"address","name":"owner","type":"address"}],"name":"OwnableInvalidOwner","type":"error"},{"inputs":[{"internalType":"address","name":"account","type":"address"}],"name":"OwnableUnauthorizedAccount","type":"error"},{"inputs":[],"name":"ReentrancyGuardReentrantCall","type":"error"},{"inputs":[],"name":"TransfersLocked","type":"error"},{"inputs":[],"name":"ZeroAddress","type":"error"},{"inputs":[],"name":"ZeroAmount","type":"error"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"owner","type":"address"},{"indexed":true,"internalType":"address","name":"spender","type":"address"},{"indexed":false,"internalType":"uint256","name":"value","type":"uint256"}],"name":"Approval","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"previousOwner","type":"address"},{"indexed":true,"internalType":"address","name":"newOwner","type":"address"}],"name":"OwnershipTransferred","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"internalType":"address","name":"account","type":"address"}],"name":"Paused","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"account","type":"address"},{"indexed":false,"internalType":"uint256","name":"amount","type":"uint256"}],"name":"TokensBurned","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"from","type":"address"},{"indexed":true,"internalType":"address","name":"to","type":"address"},{"indexed":false,"internalType":"uint256","name":"value","type":"uint256"}],"name":"Transfer","type":"event"},{"anonymous":false,"inputs":[],"name":"TransfersUnlocked","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"internalType":"address","name":"account","type":"address"}],"name":"Unpaused","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"account","type":"address"},{"indexed":false,"internalType":"bool","name":"status","type":"bool"}],"name":"WhitelistUpdated","type":"event"},{"inputs":[],"name":"TOTAL_SUPPLY","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"owner","type":"address"},{"internalType":"address","name":"spender","type":"address"}],"name":"allowance","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"spender","type":"address"},{"internalType":"uint256","name":"value","type":"uint256"}],"name":"approve","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"account","type":"address"}],"name":"balanceOf","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address[]","name":"addresses","type":"address[]"},{"internalType":"bool","name":"status","type":"bool"}],"name":"batchWhitelist","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"burn","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"account","type":"address"},{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"burnFrom","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"from","type":"address"},{"internalType":"address","name":"to","type":"address"}],"name":"canTransfer","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"decimals","outputs":[{"internalType":"uint8","name":"","type":"uint8"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"getContractInfo","outputs":[{"internalType":"uint256","name":"totalTokenSupply","type":"uint256"},{"internalType":"uint256","name":"currentSupply","type":"uint256"},{"internalType":"bool","name":"transfersLocked","type":"bool"},{"internalType":"bool","name":"contractPaused","type":"bool"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"","type":"address"}],"name":"isWhitelisted","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"na

me","outputs":[{"internalType":"string","name":"","type":"string"}],"stateMutability":"view","type":"functio
n"},{"inputs":[],"name":"owner","outputs":[{"internalType":"address","name":"","type":"address"}],"stateM
utability":"view","type":"function"},{"inputs":[],"name":"pause","outputs":[],"stateMutability":"nonpayable
","type":"function"},{"inputs":[],"name":"paused","outputs":[{"internalType":"bool","name":"","type":"bool"
}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"renounceOwnership","outputs":[],"state
Mutability":"nonpayable","type":"function"},{"inputs":[],"name":"symbol","outputs":[{"internalType":"strin
g","name":"","type":"string"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"totalSupply",
"outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function
"},{"inputs":[{"internalType":"address","name":"to","type":"address"},{"internalType":"uint256","name":"a
mount","type":"uint256"}],"name":"transfer","outputs":[{"internalType":"bool","name":"","type":"bool"}],"s
tateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"from","type"
:"address"},{"internalType":"address","name":"to","type":"address"},{"internalType":"uint256","name":"am
ount","type":"uint256"}],"name":"transferFrom","outputs":[{"internalType":"bool","name":"","type":"bool"}]
,"stateMutability":"nonpayable","type":"function"},{"inputs":[],"name":"transferLocked","outputs":[{"inter
nalType":"bool","name":"","type":"bool"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalT
ype":"address","name":"newOwner","type":"address"}],"name":"transferOwnership","outputs":[],"stateMut
ability":"nonpayable","type":"function"},{"inputs":[],"name":"unlockTransfers","outputs":[],"stateMutabilit
y":"nonpayable","type":"function"},{"inputs":[],"name":"unpause","outputs":[],"stateMutability":"nonpayab
le","type":"function"},{"inputs":[{"internalType":"address","name":"addr","type":"address"},{"internalType"
:"bool","name":"status","type":"bool"}],"name":"whitelistAddress","outputs":[],"stateMutability":"nonpaya
ble","type":"function"}]

Byte Code

60806040526008805460ff191660011790553480156200001e57600080fd5b503360405180604001604052806
00d81526020016c14d85b5cde4810d85c1a5d185b609a1b8152506040518060400160405280603815260200
162534d3960e81b8152508160039081620000739190620003 9e565b506004620000828262000039e565b505
0506001600160a01b038116620000b557604051631e4fbdf760e01b8152600060048201526024015b60405180
910390fd5b620000c08162000136565b506001600655620000df336c013fe2e171cda1978db80000006200018
8565b336000818152600760209081526040918290208054 60ff191660019081179091559151918252 7ff93f9a7
6c1bf3444d22400a00cb9fe990e6abe9dbb333fda48859cfee864543d91016040518091039a262000492565b
60058054600160016 0a01b038381166001600160a01b0319831681179093556040519116919082907f8be0079
c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e090600090a35050565b6001600160a01b0
38216620001b45760405163ec442f0560e01b8152600060048201526024016200 0ac565b620001c260008383
620001c6565b5050565b6001600160a01b038316620001f557806002600082825462000 1e991906200046a565
b9091555062000269905056 5b6001600160a01b0383166000908152602081905260409020548181101562000
24a576040516339143 4e360e21b8152600160016 0a01b0385166004820152602481018290526044810183905
2606401620000ac565b6001600160a01b0384166000908152602081905260409020908290039055 5b600160016
0a01b0382166200028757600280548290039055620002a6565b6001600160a01b0382166000908152602081 9
052604090208054820190555b816001600160a01b031683600160016 0a01b03167fddf252ad1be2c89b69c2b0
68fc378daa952ba7f163c4a11628f55a4df523b3ef83604051620002ec9181526020019056 5b60405180910390
a3505050565b634e487b7160e01b6000526041600452602460 00fd5b600181811c908216806200032457607f8
21691505b602082108103620003455 7634e487b7160e01b600052602260045260246000fd5b50919050565b60
1f821115620003995760008181526020812060 1f850160051c810160208610156200037457508 05b601f850160
051c820191505b818110156200039557828155600101620003805 65b5050505b505050565b81516001600160
401b038111156200003ba57620003ba620002f9565b620003d281620003cb8454620003 0f565b846200034b565
b602080601f831160018114620004 0a576000841562000 3f15750858301515b600019600386901b1c19166001
85901b1785556200039556 5b600085815260208120601f198616915b828110156200043b57888601518255948
401946001909101908401620004 1a565b5085821015620004 5a57875015160001960038 8901b60f8161c1916
81555b5050505050600190811b01905550565b808201808211156200048c57634e487b7160e01b6000526011
60045260246000fd5b92915050565b61123880620004a26000396000f3fe6080604052348015610010576000 8
0fd5b506004361061017357600035 60e01c806370a08231116100de578063902d55a511610097578063b5adce
3f11610071578063b5adce3f1461032d578063b9a45aac1461034057 8063dd62ed3e1461035357 8063f2fde38
b1461038c57600080fd5b8063902d55a5146102fe57806395d89b411461031257 8063a9059cbb1461031a5760
0080fd5b806370a08231146102675780637 15018a61461029057806379cc67901461029780637cc1f867146
102ab5780638456cb59146102db5780638da5cb5b146102e357600080fd5b8063313ce567116101305780633 1
3ce567146101f55780633af32abf146102045780633f4ba83a1461022757806342966c681461022f5780635c97

5abb146102425780636c85cf671461025457600080fd5b806306fdde03146101785780630 95ea7b3146101965
7806312686aae146101b957806318160ddd146101c657806321842be3146101d857806323b872dd146101e25
75b600080fd5b61018061039f565b60405161018d9190610f7b565b6040518091039 0f35b6101a96101a43660
04610fe5565b610431565b604051901515815260200161018d565b6008546101a99060ff1681565b6002545b6
040519081526020016101 8d565b6101e061044b565b005b6101a96101f036600461100f565b6104df565b6040
516012815260200161018d565b6101a96102123660046110 4b565b600760205260009081526040902054 60ff1
681565b6101e06105cf565b6101e061023d366004611066565b6105e1565b600554600160a01b900460ff1661
01a9565b6101a961026236600461107f565b610646565b6101ca6102753660046110 4b565b6001600160a01b0
3166000908152602081905260409020549056 5b6101e061069a565b6101e06102a6366004610fe5565b6106ac
565b6102b3610754565b604080519485526020850193909352901515918301919091521515606082015260800
161018d565b6101e0610799565b600554604051600160016 0a01b039091168152602001618d565b6101ca6
c013fe2e171cda1978db800000081565b6101806107a9565b6101a961032836600461 0fe5565b6107b8565b61
01e061033b3660046110c2565b6108a0565b6101e061034e366004611146565b6109c2565b6101ca61036136
600461107f565b6001600160a01b039182166000908152600160209081526040808320939094168252919091 5
2205490565b6101e061039a36600461104b565b610a52565b6060600380546103ae90611170565b80601f0160
208091040260200160405190810160405280929190818152602001828054 6103da90611170565b80156104275
780601f106103fc576101008083540402835291602001919610427565b820191906000526020600020905b8154
8152906001019060200180831161040a57829003601f168201915b505050505090509056 5b60003361043f818
585610a90565b60019150505b92915050565b610453610aa2565b60085460ff166104aa5760405162461bcd60
e51b8152602060048201526 01a6024820152 7f5472616e736665727320616c726561647920756e6c6f636b656
400000000000060448201526064015b6040518091039 0fd5b6008805460ff191690556040517f1d8b2f61c84f3
31c359476b447a0ddc4fd75f10d265a30e609526e440cdc3a4790600090a1565b60006104e9610acf565b6104f
1610afa565b600854849084906 0ff16801561052057506001600160a01b03821660009081526007602052604
090205460ff16155b801561054557506001600160a01b038116600090815260076020526040902054 60ff16155b
156105635760405163336e278fd60e21b8152600401604051809 10390fd5b846001600160a01b03811661058b5
760405163d92e233d60e01b8152600401604051809 10390fd5b84806000036105ad57604051631f2a200560e0
1b8152600401604051809 10390fd5b6105b8888888610b24565b945050505050 6105c86001600655565b93925
05050565b6105d7610aa2565b6105df610b48565b565b808060000361060357604051631f2a200560e01b8152
6004016040518091039 0fd5b61060d3383610b9d565b6040518281523390 7ffd38818f5291bf0bb3a2a48aadc0
6ba8757865d1dabd804585338aab3009dcb690602001604051809 10390a25050565b60085460009060ff16158
061067357506001600160a01b038316600090815260076020526040902054 60ff165b806105c8575050600160
0160a01b031660009081526007602052604090205460ff16919050565b6106a2610aa2565b6105df6000610bd
7565b816001600160a01b0381166106d45760405163d92e233d60e01b8152600401604051809 10390fd5b818
06000036106f657604051631f2a200560e01b8152600401604051809 10390fd5b610701843385610c29565b61
070b8484610b9d565b836001600160a01b03167ffd38818f5291bf0bb3a2a48aadc06ba8757865d1dabd80458
5338aab3009dcb6846040516107469181526020019056 5b6040518091039 0a250505050565b6000806000806
c013fe2e171cda1978db800000061077160025490565b60085460ff1661078b60055460ff600160a01b9091041
690565b935093509350935090919293565b6107a1610aa2565b6105df610ca2565b6060600480546103ae9061
1170565b60006107c2610acf565b6107ca610afa565b600854339084906 0ff1680156107f957506001600160a0
1b038216600090815260076020526040902054 60ff16155b801561081e57506001600160a01b0381166000908
152600760205260409020 5460ff16155b1561083c576040516336e278fd60e21b8152600401604051809 10390f
d5b846001600160a01b0381166108645760405163d92e233d60e01b8152600401604051809 10390fd5b84806
000036108657604051631f2a200560e01b8152600401604051809 10390fd5b6108908787610ce5565b945050
505050610445600160065 5565b6108a8610aa2565b60005b828110156109bc576000848438181106108c7576
108c76111aa565b905060200201602081019061091691906104b565b6001600160a01b0316146109aa57816
0076000868685818110610901576109016111aa565b905060200201602081019061091691906104b565b600
1600160a01b0316815260208101919091526040016000208054 60ff191691151591909117905583838281811 0
6109057610950611 1aa565b905060200201602081019061096591906104b565b6001600160a01b03167ff93
f9a76c1bf3444d22400a00cb9fe990e6abe9dbb333fda48859cfee864543d836040516109a1911515815260200
190565b6040518091039 0a25b806109b4816111d6565b9150506108ab565b50505050565b6109ca610aa2565
b816001600160a01b0381166109f25760405163d92e233d60e01b8152600401604051809 10390fd5b60016001
60a01b0383166000818152600760209081526040918290208054 60ff191686151590811790915591519182527f
f93f9a76c1bf3444d22400a00cb9fe990e6abe9dbb333fda48859cfee864543d9101604051809 10390a2505050
565b610a5a610aa2565b6001600160a01b038116610a8457604051631e4fbdf760e01b8152600060048201526
024016104a1565b610a8d81610bd7565b50565b610a9d838836001610cf3565b505050565b6005546001600
160a01b031633146105df5760405163118cdaa760e01b8152336004820152602401610 4a1565b600554600160
a01b900460ff16156105df5760405163d93c066560e01b8152600401604051809 10390fd5b600260065403610
b1d57604051633ee5aeb560e01b8152600401604051809 10390fd5b6002600655565b600033610b328582856

10c29565b610b3d858585610dc8565b5060019493505050565b610b50610e27565b6005805460ff60a01b19
1690557f5db9ee0a495bf2e6ff9c91a7834c1ba4fdd244a5e8aa4e537bd38aeae4b073aa335b6040516001600
160a01b039091168152602001604051809103 90a1565b6001600160a01b038216610bc757604051634b637e8f
60e11b8152600060048201526024016104a1565b610bd382600083610e51565b5050565b6005805460016001
60a01b038381166001600160a01b0319831681179093556040519116919082907f8be0079c531659141344cd1
fd0a4f28419497f9722a3daafe3b4186f6b6457e090600090a35050565b6001600160a01b03838116600090815
260016020908152604080832093861683529290522054600019811015610 9bc5781811015610c935760405163
7dc7a0d960e11b81526001600160a01b038416600482015260248101829052604481018390526064016104a1
565b6109bc84848484036000610cf3565b610caa610acf565b6005805460ff60a01b1916600160a01b1790557f
62e78cea01bee320cd4e420270b5ea74000d11b0c9f74754ebdbfc544b05a258610b803390565b60003361043
f818585610dc8565b6001600160a01b038416610d1d5760405163e602df0560e01b8152600060048201526024
016104a1565b6001600160a01b038316610d4757604051634a1406b160e11b8152600060048201526024016 1
04a1565b6001600160a01b03808516600090815260016020908152604080832093871683529290522082905 58
0156109bc57826001600160a01b031684600160016 0a01b03167f8c5be1e5ebec7d5bd14f71427d1e84f3dd03
14c0f7b2291e5b200ac8c7c3b92584604051610dba91815260200190565b60405180910390a350505050565b6
001600160a01b038316610df257604051634b637e8f60e11b8152600060048201526024016104a1565b600160
0160a01b038216610e1c5760405163ec442f0560e01b8152600060048201526024016104a1565b610a9d83838
3610e51565b600554600160a01b900460ff166105df57604051638dfc202b60e01b8152600401604051809103
90fd5b6001600160a01b038316610e7c57806002600082825461 0e7191906111ef565b90915550610eee90505
65b6001600160a01b038316600090815260208190526040902054818110 15610ecf5760405163391434e360e2
1b81526001600160a01b03851660048201526024810182905260448101839052606 4016104a1565b60016001
60a01b038416600090815260208190526040902090208290039055 5b6001600160a01b038216610f0a576002805
48290039055610f29565b6001600160a01b038216600090815260208190526040902080548201905 55b816001
600160a01b031683600160016 0a01b03167fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a
4df523b3ef83604051610f6e91815260200190565b60405180910390a3505050565b6000602080835283528 51808
285015260005b81811015610fa857858101830151858201604001528201610f8c565b5060006040828601015 2
6040601f19601f830116850101925050509291505056 5b8035600160016 0a01b0381168114610fe057600080fd
5b919050565b6000806040838503121561 0ff857600080fd5b61100183610fc9565b9460209390930135935050
50565b6000806000606084860312156110245760008 0fd5b61102d84610fc9565b925061103b60208501610fc
9565b9150604084013590509250925092565b6000602082840312156110 5d57600080fd5b6105c882610fc956
5b6000602082840312156110785760008 0fd5b5035919050565b6000806040838503121561109257600080fd5
b61109b83610fc9565b91506110a960208401610fc9565b90509250929050565b8035801515811 4610fe05760
0080fd5b6000806000604084860312156110d757600080fd5b833567ffffffffffffffff8082111561 10ef57600080
fd5b818601915086601f830112611110357600080fd5b8135818111156111125760008 0fd5b8760208260051b8
501011115611127576000 80fd5b602092830195509350611 13d91860190506110b2565b90509250925092565b
6000806040838503121561115957600080fd5b61116283610fc9565b91506110a9602084016110b2565b60018
1811c908216806111845760 7f821691505b602082108103611 1a457634e487b7160e01b600052602260045260
246000fd5b50919050565b634e487b7160e01b600052603260045260246000fd5b634e487b7160e01b6000526
0116004526024600 0fd5b6000600182016111e8576111e86111c0565b5060010190565b808201808211156104
45576104456111c056fea2646970667358221220 2959e6a24dc053671f43d9e34f8d6ef33381b9b2f45e3937d6
9d2446d1d532c664736f6c63430008140033

Conclusion

Electro-Pact had conducted a security audit for SAMSYTOKEN (SM9) staking functions. Total 0 issues werefound, but none of these issues represented actual bugs or security problems. These issues thenwere accepted by the SAMSYTOKEN (SM9) team.

To improve the quality for this report, and for Electro-Pact's Smart Contract Audit report in general, wegreatly appreciate any constructive feedback or suggestions, on our methodology, audit findings, orpotential gaps in scope/coverage.

Appendices

Appendix A – Security Issue Status Definitions

Status	Definition
Open	The issue has been reported and currently being review by the smart contract developers/issuer.
Unresolved	The issue is acknowledged and planned to be addressed in future. At the time of the corresponding report version, the issue has not been fixed.
Resolved	The issue is acknowledged and has been fully fixed by the smart contract developers/issuer.
Rejected	The issue is considered to have no security implications or to make only little security impacts, so it is not planned to be addressed and won't be fixed.

Appendix B – Severity Explanation

Severity	Definition
CRITICAL	Issues, considered as critical, are straightforwardly exploitable bugs and security vulnerabilities. It is advised to immediately resolve these issues in order to prevent major problems or a full failure during contract system operation.
MAJOR	Major issues are bugs and vulnerabilities, which cannot be exploited directly without certain conditions. It is advised to patch the codebase of the smart contract as soon as possible, since these issues, with a high degree of probability, can cause certain problems for operation of the smart contract or severe security impacts on the system in some way.
MEDIUM	In terms of medium issues, bugs and vulnerabilities exist but cannot be exploited without extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
MINOR	Minor issues are generally objective in nature but do not represent actual bugs or security problems. It is advised to address these issues, unless there is a clear reason not to.
INFO	Issues, regarded as informational (info), possibly relate to “guides for the best practices” or “readability”. Generally, these issues are not actual bugs or vulnerabilities. It is recommended to address these issues, if it make effective and secure improvements to the smart contract codebase.

Appendix C – Smart Contract Weakness Classification Registry (SWC Registry)

ID	Name	Description
	Coding Specification Issues	
SWC-100	Function Default Visibility	It is recommended to make a conscious decision on which visibility type (*external*, *public*, *internal* or *private*) is appropriate for a function. By default, functions without concrete specifiers are *public*.
SWC-102	Outdated Compiler Version	It is recommended to use a recent version of the Solidity compiler to avoid publicly disclosed bugs and issues in outdated versions.
SWC-103	Floating Pragma	It is recommended to lock the pragma to ensure that contracts do not accidentally get deployed using.
SWC-108	State Variable Default Visibility	Variables can be specified as being *public*, *internal* or *private*. Explicitly define visibility for all state variables.
SWC-111	Use of Deprecated Solidity Functions	Solidity provides alternatives to the deprecated constructions, the use of which might reduce code quality. Most of them are aliases, thus replacing old constructions will not break current behavior.
SWC-118	Incorrect Constructor Name	It is therefore recommended to upgrade the contract to a recent version of the Solidity compiler and change to the new constructor declaration (the keyword *constructor*).
	Design Defect Issues	
SWC-113	DoS with Failed Call	External calls can fail accidentally or deliberately, which can cause a DoS condition in the contract. It is better to isolate each external call into its own transaction and implement the contract logic to handle failed calls.

SWC-119	Shadowing State Variables	Review storage variable layouts for your contract systems carefully and remove any ambiguities. Always check for compiler warnings as they can flag the issue within a single contract.
SWC-125	Incorrect Inheritance Order	When inheriting multiple contracts, especially ifthey have identical functions, a developer should carefully specify inheritance in the correct order (from more /general/ to more /specific/).
SWC-128	DoS With Block Gas Limit	Modifying an array of unknown size, that increases in size over time, can lead to such a Denial of Service condition. Actions that require looping across the entire data structure should be avoided.
	Coding Security Issues	
SWC-101	Integer Overflow and Underflow	It is recommended to use safe math libraries for arithmetic operations throughout the smart contract system to avoid integer overflows and underflows.
SWC-107	Reentrancy	Make sure all internal state changes are performed before the call is executed or use a reentrancy lock.
SWC-112	Delegatecall to Untrusted Callee	Use *delegatecall* with caution and make sure to never call into untrusted contracts. If the target address is derived from user input ensure to check it against a whitelist of trusted contracts.
SWC-117	Signature Malleability	A signature should never be included into a signed message hash to check if previously messages have been processed by the contract.
SWC-121	Missing Protection against Signature Replay Attacks	In order to protect against signature replay attacks, store every message hash that has been processed by the smart contract, include the address of the contract that processes the message and never generate the message hash including the signature.
SWC-122	Lack of Proper Signature Verification	It is not recommended to use alternate verification schemes that do not require proper signature verification through *ecrecover()*.

SWC-130	Right-To-Left-Override control character (U+202E)	The character *U+202E* should not appear in the source code of a smart contract.
	Coding Design Issues	
SWC-104	Unchecked Call Return Value	If you choose to use low-level call methods (e.g. *call()*), make sure to handle the possibility that the call fails by checking the return value.
SWC-105	Unprotected Ether Withdrawal	Implement controls so withdrawals can only be triggered by authorized parties or according to the specs of the smart contract system.
SWC-106	Unprotected SELFDESTRUCT Instruction	Consider removing the self-destruct functionality. If absolutely required, it is recommended to implement a multisig scheme so that multiple parties must approve the self-destruct action.
SWC-110	Assert Violation	Consider whether the condition checked in the *assert()* is actually an invariant. If not, replace the *assert()* statement with a *require()* statement.
SWC-116	Block values as a proxy for time	Developers should write smart contracts with the notion that block values are not precise, and the use of them can lead to unexpected effects. Alternatively, they may make use oracles.
SWC-120	Weak Sources of Randomness from Chain Attributes	To avoid weak sources of randomness, use commitment scheme, e.g. RANDAO, external sources of randomness via oracles, e.g. Oraclize, or Bitcoin block hashes.
SWC-123	Requirement Violation	If the required logical condition is too strong, it should be weakened to allow all valid external inputs. Otherwise, make sure no invalid inputs are provided.
SWC-124	Write to Arbitrary Storage Location	As a general advice, given that all data structures share the same storage (address) space, one should make sure that writes to one data structure cannot inadvertently overwrite entries of another data structure.

SWC-132	Unexpected Ether balance	Avoid strict equality checks for the Ether balance in a contract.
SWC-133	Hash Collisions With Multiple Variable Length Arguments	When using *abi.encodePacked()*, it's crucial to ensure that a matching signature cannot beachieved using different parameters. Alternatively, you can simply use *abi.encode()* instead. It is also recommended to use replay protection.
	Coding Hidden Dangers	
SWC-109	Uninitialized Storage Pointer	Uninitialized local storage variables can point to unexpected storage locations in the contract. If a local variable is sufficient, mark it with *memory*, else *storage* upon declaration. As of compiler version 0.5.0 and higher this issue has been systematically resolved.
SWC-114	Transaction Order Dependence	A possible way to remedy for race conditions in submission of information in exchange for a reward is called a commit reveal hash scheme. The best fix for the ERC20 race condition is to add a field to the inputs of approve which is the expected current value and to have approve revert or add a safe approve function.
SWC-115	Authorization through tx.origin	*tx.origin* should not be used for authorization. Use *msg.sender* instead.
SWC-126	Insufficient Gas Griefing	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract. To avoid them, only allow trusted users to relay transactions and require that the forwarder provides enough gas.
SWC-127	Arbitrary Jump with Function Type Variable	The use of assembly should be minimal. A developer should not allow a user to assign arbitrary values to function type variables.

SWC-129	Typographical Error	The weakness can be avoided by performing pre-condition checks on any math operation or using a vetted library for arithmetic calculations such as SafeMath developed by OpenZeppelin.
SWC-131	Presence of unused variables	Remove all unused variables from the code base.
SWC-134	Message call with hardcoded gas amount	Avoid the use of *transfer()* and *send()* and do not otherwise specify a fixed amount of gas when performing calls. Use *.call.value(...)(“”)* instead.
SWC-135	Code With No Effects	It’s important to carefully ensure that your contract works as intended. Write unit tests to verify correct behaviour of the code.
SWC-136	Unencrypted Private Data On-Chain	Any private data should either be stored off-chain, or carefully encrypted.

Appendix D – Related Common Weakness Enumeration (CWE)

The SWC Registry loosely aligned to the terminologies and structure used in the CWE while overlaying a wide range of weakness variants that are specific to smart contracts.

CWE IDs *, to which SWC Registry is related, are listed in the following table:

CWE ID	Name	Related SWC IDs
CWE-284	**Improper Access Control**	SWC-105, SWC-106
CWE-294	Authentication Bypass by Capture-replay	SWC-133
CWE-664	**Improper Control of a Resource Through its Lifetime**	SWC-103
CWE-123	Write-what-where Condition	SWC-124
CWE-400	Uncontrolled Resource Consumption	SWC-128
CWE-451	User Interface (UI) Misrepresentation of Critical Information	SWC-130
CWE-665	Improper Initialization	SWC-118, SWC-134
CWE-767	Access to Critical Private Variable via Public Method	SWC-136
CWE-824	Access of Uninitialized Pointer	SWC-109
CWE-829	Inclusion of Functionality from Untrusted Control Sphere	SWC-112, SWC-116
CWE-682	**Incorrect Calculation**	SWC-101
CWE-691	**Insufficient Control Flow Management**	SWC-126
CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization (“Race Condition”)	SWC-114
CWE-480	Use of Incorrect Operator	SWC-129
CWE-667	Improper Locking	SWC-132
CWE-670	Always-Incorrect Control Flow Implementation	SWC-110
CWE-696	Incorrect Behavior Order	SWC-125
CWE-841	Improper Enforcement of Behavioral Workflow	SWC-107
CWE-693	**Protection Mechanism Failure**	

CWE-330	Use of Insufficiently Random Values	SWC-120
CWE-345	Insufficient Verification of Data Authenticity	SWC-122
CWE-347	Improper Verification of Cryptographic Signature	SWC-117, SWC-121
CWE-703	**Improper Check or Handling of Exceptional Conditions**	SWC-113
CWE-252	Unchecked Return Value	SWC-104
CWE-710	**Improper Adherence to Coding Standards**	SWC-100, SWC-108, SWC-119
CWE-477	Use of Obsolete Function	SWC-111, SWC-115
CWE-573	Improper Following of Specification by Caller	SWC-123
CWE-695	Use of Low-Level Functionality	SWC-127
CWE-1164	Irrelevant Code	SWC-131, SWC-135
CWE-937	**Using Components with Known Vulnerabilities**	SWC-102

** CWE IDs, which are presented in bold, are the greatest parent nodes of those nodes following it.*

All IDs in the CWE list above are relevant to the view "Research Concepts" (CWE-1000), except for CWE-937, which is relevant to the "Weaknesses in OWASP Top Ten (2013)" (CWE-928).