SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

Origin.cash

21 Sept 2024

This security assessment report was prepared by SolidityScan.com, a cloud-based Smart Contract Scanner.

Table of Contents

MISSING PAYABLE IN CALL FUNCTION
MISSING UNDERSCORE IN NAMING VARIABLES
USE CALL INSTEAD OF TRANSFER OR SEND
IN-LINE ASSEMBLY DETECTED
VARIABLES SHOULD BE IMMUTABLE
ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT
AVOID RE-STORING VALUES
CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE
CHEAPER INEQUALITIES IN IF()
CHEAPER INEQUALITIES IN REQUIRE()
DEFINE CONSTRUCTOR AS PAYABLE
DUPLICATED REQUIRE/REVERT SHOULD BE A MODIFIER
FUNCTIONS CAN BE IN-LINED
NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT
STORAGE VARIABLE CACHING IN MEMORY
UNNECESSARY CHECKED ARITHMETIC IN LOOP
USE BYTES.CONCAT() INSTEAD OF ABI.ENCODEPACKED
USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

05 Scan History

06 Disclaimer

01. Vulnerability Classification and Severity

Description

To enhance navigability, the document is organized in descending order of severity for easy reference. Issues are categorized as *Fixed*, *Pending Fix*, or *Won't Fix*, indicating their current status. *Won't Fix* denotes that the team is aware of the issue but has chosen not to resolve it. Issues labeled as *Pending Fix* state that the bug is yet to be resolved. Additionally, each issue's severity is assessed based on the risk of exploitation or the potential for other unexpected or unsafe behavior.

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

High-severity vulnerabilities pose a significant risk to both the Smart Contract and the organization. They can lead to user fund losses, may have conditional requirements, and are challenging to exploit.

Medium

The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.

Low

The issue has minimal impact on the contract's ability to operate.

Informational

The issue does not affect the contract's operational capability but is considered good practice to address.

Gas

This category deals with optimizing code and refactoring to conserve gas.

02. Executive Summary

Origin.cash

Uploaded Solidity File(s)

Language	Audit Methodology	Website
Solidity	**Static Scanning**	-
Publishers/Owner Name	**Organization**	**Contact Email**
-	-	-

Security Score is AVERAGE

The SolidityScan score is calculated based on lines of code and weights assigned to each issue depending on the severity and confidence. To improve your score, view the detailed result and leverage the remediation solutions provided.

This report has been prepared for Origin.cash using SolidityScan to scan and discover vulnerabilities and safe coding practices in their smart contract including the libraries used by the contract that are not officially recognized. The SolidityScan tool runs a comprehensive static analysis on the Solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over (100+) modules. The scanning and auditing process covers the following areas:

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date.

The SolidityScan Team recommends running regular audit scans to identify any vulnerabilities that are introduced after Origin.cash introduces new features or refactors the code.

03. Findings Summary

Origin.cash
File Scan

Security Score	Scan duration	Lines of code
73.08/100	3 secs	561

5	0	3	28	25	36
Crit	High	Med	Low	Info	Gas

This audit report has not been verified by the SolidityScan team. To learn more about our published reports. click here

ACTION TAKEN

0	0	0	97
Fixed	False Positive	Won't Fix	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
C001	Critical	INCORRECT ACCESS CONTROL	2	Automated	Pending Fix
C002	Critical	LOCKED ETHER	2	Automated	Pending Fix
C003	Critical	PHANTOM PERMIT	1	Automated	Pending Fix
M001	Medium	HASH COLLISIONS WITH BYTES ARGUMENT ON ABI.ENCODEPACKED	1	Automated	Pending Fix
M002	Medium	USING EXTCODESIZE TO CHECK FOR EXTERNALLY OWNED ACCOUNTS	2	Automated	Pending Fix
L001	Low	USE OF FLOATING PRAGMA	8	Automated	Pending Fix
L002	Low	MISSING EVENTS	10	Automated	Pending Fix
L003	Low	OUTDATED COMPILER VERSION	8	Automated	Pending Fix
L004	Low	USE OWNABLE2STEP	2	Automated	Pending Fix
I001	Informational	ABI.ENCODEPACKED MAY CAUSE COLLISION	2	Automated	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I002	Informational	ADDING A RETURN STATEMENT WHEN THE FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT	1	Automated	*Pending Fix*
I003	Informational	BLOCK VALUES AS A PROXY FOR TIME	3	Automated	*Pending Fix*
I004	Informational	IF-STATEMENT REFACTORING	1	Automated	*Pending Fix*
I005	Informational	MISSING PAYABLE IN CALL FUNCTION	1	Automated	*Pending Fix*
I006	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	12	Automated	*Pending Fix*
I007	Informational	USE CALL INSTEAD OF TRANSFER OR SEND	2	Automated	*Pending Fix*
I008	Informational	IN-LINE ASSEMBLY DETECTED	2	Automated	*Pending Fix*
I009	Informational	VARIABLES SHOULD BE IMMUTABLE	1	Automated	*Pending Fix*
G001	Gas	ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT	2	Automated	*Pending Fix*
G002	Gas	AVOID RE-STORING VALUES	2	Automated	*Pending Fix*
G003	Gas	CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE	2	Automated	*Pending Fix*
G004	Gas	CHEAPER INEQUALITIES IN IF()	1	Automated	*Pending Fix*
G005	Gas	CHEAPER INEQUALITIES IN REQUIRE()	2	Automated	*Pending Fix*
G006	Gas	DEFINE CONSTRUCTOR AS PAYABLE	4	Automated	*Pending Fix*
G007	Gas	DUPLICATED REQUIRE/REVERT SHOULD BE A MODIFIER	7	Automated	*Pending Fix*
G008	Gas	FUNCTIONS CAN BE IN-LINED	2	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
G009	Gas	NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT	3	Automated	*Pending Fix*
G010	Gas	STORAGE VARIABLE CACHING IN MEMORY	3	Automated	*Pending Fix*
G011	Gas	UNNECESSARY CHECKED ARITHMETIC IN LOOP	4	Automated	*Pending Fix*
G012	Gas	USE BYTES.CONCAT() INSTEAD OF ABI.ENCODEPACKED	2	Automated	*Pending Fix*
G013	Gas	USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE	2	Automated	*Pending Fix*

04. Vulnerability Details

Issue Type

INCORRECT ACCESS CONTROL

S. No.	Severity	Detection Method	Instances
C001	Critical	Automated	2

Upgrade your Plan to view the full report

2 Critical Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

HASH COLLISIONS WITH BYTES ARGUMENT ON ABI.ENCODEPACKED

S. No.	Severity	Detection Method	Instances
M001	Medium	Automated	1

Bug ID	File Location	Line No.	Action Taken

Upgrade your Plan to view the full report

1 Medium Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

USE OF FLOATING PRAGMA

S. No.	Severity	Detection Method	Instances
L001	Low	Automated	8

Bug ID	File Location	Line No.	Action Taken

Upgrade your Plan to view the full report

8 Low Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

ABI.ENCODEPACKED MAY CAUSE COLLISION

S. No.	Severity	Detection Method	Instances
I001	Informational	Automated	2

Bug ID	File Location	Line No.	Action Taken

Upgrade your Plan to view the full report

2 Informational Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT

S. No.	Severity	Detection Method	Instances
G001	Gas	Automated	2

Description

The contract is found to contain a struct with multiple variables defined in it. When a struct is assigned in a single op eration, Solidity may perform costly storage operations, which can be inefficient. This often results in increased gas costs due to multiple SLOAD and SSTORE operations happening at once

Bug ID	File Location	Line No.	Action Taken
SSP_24004_49	/OriginCash/OriginCash/OriginCash.sol	L137 - L137	*Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L137 - L137

```
136           if (depositsLength < 10) {
137               deposits.push(Deposit(msg.value, block.timestamp));
138           } else {
139               // When the array reaches 10 elements, overwrite the oldest element (circular
      buffer)
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_50	/OriginCash/OriginCash/OriginCash.sol	L140 - L140	⚠ *Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L140 - L140

```
139             // When the array reaches 10 elements, overwrite the oldest element (circular
    buffer)
140             deposits[depositIndex] = Deposit(msg.value, block.timestamp);
141         }
142         // Update the index to overwrite the next element (circular buffer)
```

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G002	Gas	Automated	2

Description

The function is found to be allowing re-storing the value in the contract's state variable even when the old value is eq ual to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operation (2900 ga s), which could be avoided. If the old value and the new value are the same, not updating the storage would avoid thi s cost and could instead incur a `Gcoldsload` (2100 gas) or a `Gwarmaccess` (100 gas), potentially saving gas.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_53	/OriginCash/OriginCash/Ownable.sol	L24 - L27	*Pending Fix*

/OriginCash/OriginCash/Ownable.sol L24 - L27

```
23
24          function changeOwner(address newOwner) public onlyOwner {
25              require(newOwner != address(0), "New owner address cannot be zero");
26              owner = newOwner;
27          }
28
29          function getOwner() public view returns (address) {
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_54	/OriginCash/OriginCash/OriginCash.sol	L68 - L77	⚠ *Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L68 - L77

```
67
68      function changeFeeReceiptAddress(address _newFeeReceiptAddress)
69          public
70          onlyOwner
71      {
72          require(
73              _newFeeReceiptAddress != address(0),
74              "FeeReceiptAddress: Zero address"
75          );
76          feeReceiptAddress = _newFeeReceiptAddress;
77      }
78
79      // Handles user deposits
```

Issue Type

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

S. No.	Severity	Detection Method	Instances
G003	Gas	Automated	2

Description

The repeated usage of `address(this)` within the contract could result in increased gas costs due to multiple exec utions of the same computation, potentially impacting efficiency and overall transaction expenses.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_38	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L42 - L42	*Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L42 - L42

```
41          require(
42              address(this).balance >= _amountAfterFee + _fee,
43              "Insufficient funds"
44          );
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_39	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L75 - L75	⚠ *Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L75 - L75

```
74          require(_recipient != address(0), "Invalid recipient address");
75          uint256 balance = address(this).balance;
76          (bool success, ) = _recipient.call{value: balance}("");
77          require(success, "Transfer failed");
```

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G004	Gas	Automated	1

Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Bug ID	File Location	Line No.	Action Taken
SSP_24004_68	/OriginCash/OriginCash/OriginCash.sol	L136 - L136	*Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L136 - L136

```
135         uint256 depositsLength = deposits.length;
136         if (depositsLength < 10) {
137             deposits.push(Deposit(msg.value, block.timestamp));
138         } else {
```

Issue Type

CHEAPER INEQUALITIES IN REQUIRE()

S. No.	Severity	Detection Method	Instances
G005	Gas	Automated	2

Description

The contract was found to be performing comparisons using inequalities inside the `require` statement. When inside the `require` statements, non-strict inequalities `(>=, <=)` are usually costlier than strict equalities `(>, <)`.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_92	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L42 - L42	*Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L42 - L42

```
41          require(
42              address(this).balance >= _amountAfterFee + _fee,
43              "Insufficient funds"
44          );
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_93	/OriginCash/OriginCash/OriginCash.sol	L162 - L162	⚠ *Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L162 - L162

```
161         Account storage account = accounts[_privateKeyHash];
162         require(account.balance >= _amount, "Insufficient balance");
163
164         // Ensure the proof is not reused
```

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G006	Gas	Automated	4

Description

Developers can save around 10 opcodes and some gas if the constructors are defined as payable.
However, it should be noted that it comes with risks because payable constructors can accept ETH during deployment.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_6	/OriginCash/OriginCash/Ownable.sol	L9 - L12	*Pending Fix*

/OriginCash/OriginCash/Ownable.sol L9 - L12

```
8
9         constructor(address _originCash) {
10            owner = msg.sender;
11            originCash = _originCash;
12        }
13
14        modifier onlyOwner() {
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_7	/OriginCash/OriginCash/ReentrancyGuard.sol	L14 - L16	*Pending Fix*

/OriginCash/OriginCash/ReentrancyGuard.sol L14 - L16

```
13
14        constructor() {
15            _status = _NOT_ENTERED;
16        }
17
18        /**
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_8	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L10 - L10	*Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L10 - L10

```
9
10        constructor(address _originCash) Ownable(_originCash) {}
11
12        receive() external payable {
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_9	/OriginCash/OriginCash/OriginCash.sol	L56 - L58	⚠ *Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L56 - L58

```
55        // Passes the address of this contract (OriginCash) to the Ownable constructor.
56        constructor() Ownable(address(this)) {
57            feeReceiptAddress = msg.sender;
58        }
59
60        // Stores only the last 10 transactions
```

Issue Type

DUPLICATED REQUIRE/REVERT SHOULD BE A MODIFIER

S. No.	Severity	Detection Method	Instances
G007	Gas	Automated	7

Description

The contract is found to have redundant `require()` or `if` and `revert()` statements verifying the same conditions multiple times. This not only increases the complexity but also results in higher gas costs during deployment and execution. Using Solidity's modifiers or internal functions, we can refactor these redundant checks to enhance code reusability and save deployment gas costs.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_28	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L13 - L13	*Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L13 - L13

```
12        receive() external payable {
13            require(msg.sender != address(0), "Invalid sender");
14            require(msg.value != 0, "No asset sent");
15        }
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_29	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L49 - L49	*Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L49 - L49

```
48            }
49            require(size == 0, "Address is a contract");
50
51            assembly {
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_30	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L14 - L14	*Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L14 - L14

```
13            require(msg.sender != address(0), "Invalid sender");
14            require(msg.value != 0, "No asset sent");
15        }
16
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_31	/OriginCash/OriginCash/OriginCash.sol	L124 - L124	⚠ *Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L124 - L124

```
123
124             if (!transferSuccess) {
125                 // If the transfer fails, refund the user
126                 anonymousWallet.returnFunds(payable(msg.sender));
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_32	/OriginCash/OriginCash/OriginCash.sol	L90 - L90	⚠ *Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L90 - L90

```
89              // Ensure the proof is not reused
90              require(
91                  !isProofUsed(_messageHash, _commitment),
92                  "Proof has already been used"
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_33	/OriginCash/OriginCash/OriginCash.sol	L103 - L103	*Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L103 - L103

```
102         // Verify the hashes match, data integrity
103         require(messageHash == _messageHash, "Invalid hash transaction");
104
105         // Initialize account if not already done
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_34	/OriginCash/OriginCash/OriginCash.sol	L86 - L86	*Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L86 - L86

```
85          // Check Private format
86          require(validatePrivateKey(_privateKeyHash), "Invalid private key");
87
88          require(msg.value != 0, "Zero amount");
```

Issue Type

FUNCTIONS CAN BE IN-LINED

S. No.	Severity	Detection Method	Instances
G008	Gas	Automated	2

Description

The internal function was called only once throughout the contract. Internal functions cost more gas due to additiona l `JUMP` instructions and stack operations.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_47	/OriginCash/OriginCash/Validator.sol	L30 - L36	*Pending Fix*

/OriginCash/OriginCash/Validator.sol L30 - L36

```
29        // Validate commitment (only valid if not equal to bytes32(0))
30        function validateCommitment(bytes32 _commitment)
31            internal
32            pure
33            returns (bool)
34        {
35            return _commitment != bytes32(0);
36        }
37
38        // Validate private key by calling checkPrivateKey function
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_48	/OriginCash/OriginCash/FeeCalculator.sol	L14 - L22	*Pending Fix*

/OriginCash/OriginCash/FeeCalculator.sol L14 - L22

```
13          */
14         function calculateAmountAfterFeeAndFee(uint256 _amount)
15             internal
16             view
17             returns (uint256 amountAfterFee, uint256 fee)
18         {
19             fee = (_amount * feePercentage) / 10000;
20             amountAfterFee = _amount - fee;
21             return (amountAfterFee, fee);
22         }
23
24         function getFeePercentage() external view returns (uint256) {
```

Issue Type

NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT

S. No.	Severity	Detection Method	Instances
G009	Gas	Automated	3

Description

The function having a return type is found to be declaring a local variable for returning, which causes extra gas cons umption. This inefficiency arises because creating and manipulating local variables requires additional computational steps and memory allocation.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_41	/OriginCash/OriginCash/Validator.sol	L22 - L27	*Pending Fix*

/OriginCash/OriginCash/Validator.sol L22 - L27

```
21        //Get commitment first
22        function getCommitment() public view returns (bytes32) {
23            bytes32 commitment = keccak256(
24                abi.encodePacked(block.timestamp, msg.sender, block.prevrandao)
25            );
26            return commitment;
27        }
28
29        // Validate commitment (only valid if not equal to bytes32(0))
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_43	/OriginCash/OriginCash/OriginCash.sol	L211 - L223	*Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L211 - L223

```
210       // Get the latest transactions
211       function getLatestDepositTransaction(uint256 _limit)
212           public
213           view
214           returns (Deposit[] memory)
215       {
216           uint256 depoLength = deposits.length;
217           uint256 count = depoLength > _limit ? _limit : depoLength;
218           Deposit[] memory latestDeposits = new Deposit[](count);
219           for (uint256 i = 0; i < count; ++i) {
220               latestDeposits[i] = deposits[depoLength - count + i];
221           }
222           return latestDeposits;
223       }
224
225       // Get balance by private key
```

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G010	Gas	Automated	3

Description

The contract is using the state variable multiple times in the function.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Bug ID	File Location	Line No.	Action Taken
SSP_24004_11	/OriginCash/OriginCash/ProofMapping.sol	L27 - L39	*Pending Fix*

/OriginCash/OriginCash/ProofMapping.sol L27 - L39

```
26        // Function to mark _messageHash + commitment pair as used
27        function markProofAsUsed(bytes32 _messageHash, bytes32 _commitment)
28            internal
29        {
30            // Load the value from state once into a local variable
31            bool proofAlreadyUsed = usedProofs[_messageHash][_commitment];
32
33            require(!proofAlreadyUsed, "Proof already used");
34
35            // Update the state only once after verification
36            usedProofs[_messageHash][_commitment] = true;
37
38            emit ProofUsed(_messageHash, _commitment);
39        }
40    }
41
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_12	/OriginCash/OriginCash/OriginCash.sol	L80 - L148	⚠ *Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L80 - L148

```
79        // Handles user deposits
80        function deposit(
81            string calldata _privateKeyHash,
82            bytes32 _messageHash,
83            bytes32 _commitment
84        ) public payable nonReentrant {
85            // Check Private format
86            require(validatePrivateKey(_privateKeyHash), "Invalid private key");
87
88            require(msg.value != 0, "Zero amount");
89            // Ensure the proof is not reused
90            require(
91                !isProofUsed(_messageHash, _commitment),
92                "Proof has already been used"
93            );
94
95            // Retrieve the messageHash and commitment
96            bytes32 messageHash = getMessageHash(
97                msg.value,
98                msg.sender,
99                _commitment
100           );
101
```

/OriginCash/OriginCash/OriginCash.sol

L80 - L148

```
99                _commitment
100           );
101
102           // Verify the hashes match, data integrity
103           require(messageHash == _messageHash, "Invalid hash transaction");
104
105           // Initialize account if not already done
106           Account storage account = accounts[_privateKeyHash];
107           if (bytes(account.privateKeyHash).length == 0) {
108               account.privateKeyHash = _privateKeyHash;
109           }
110
111           // The anonymous wallet contract for you to interact with
112           DynamicShieldWallet anonymousWallet = new DynamicShieldWallet(
113               originCash // Pass the originCash object
114           );
115
116           // Transfer assets to anonymous wallet
117           payable(anonymousWallet).transfer(msg.value);
118
119           // Transfer assets to OriginCash
120           bool transferSuccess = anonymousWallet.transferWhenUserDeposit(
121               msg.value
122           );
123
124           if (!transferSuccess) {
125               // If the transfer fails, refund the user
126               anonymousWallet.returnFunds(payable(msg.sender));
127               revert Warning("Deposit failed");
128           }
129
130           account.balance += msg.value; // Update account balance
131           setTotalTransactions();
132
133           // Stores only the last 10 transactions
134           uint256 depositIndex;
135           uint256 depositsLength = deposits.length;
136           if (depositsLength < 10) {
137               deposits.push(Deposit(msg.value, block.timestamp));
138           } else {
139               // When the array reaches 10 elements, overwrite the oldest element (circular buffer)
140               deposits[depositIndex] = Deposit(msg.value, block.timestamp);
141           }
142           // Update the index to overwrite the next element (circular buffer)
```

/OriginCash/OriginCash/OriginCash.sol L80 - L148

```
141             }
142             // Update the index to overwrite the next element (circular buffer)
143             depositIndex = (depositIndex + 1) % 10;
144
145             // Mark the proof as used
146             markProofAsUsed(_messageHash, _commitment);
147             emit Message(address(anonymousWallet), account.balance);
148         }
149
150         // Handles user withdrawals
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_13	/OriginCash/OriginCash/OriginCash.sol	L211 - L223	⚠ *Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L211 - L223

```
210         // Get the latest transactions
211         function getLatestDepositTransaction(uint256 _limit)
212             public
213             view
214             returns (Deposit[] memory)
215         {
216             uint256 depoLength = deposits.length;
217             uint256 count = depoLength > _limit ? _limit : depoLength;
218             Deposit[] memory latestDeposits = new Deposit[](count);
219             for (uint256 i = 0; i < count; ++i) {
220                 latestDeposits[i] = deposits[depoLength - count + i];
221             }
222             return latestDeposits;
223         }
224
225         // Get balance by private key
```

Issue Type

UNNECESSARY CHECKED ARITHMETIC IN LOOP

S. No.	Severity	Detection Method	Instances
G011	Gas	Automated	4

Description

Increments inside a loop could never overflow due to the fact that the transaction will run out of gas before the varia ble reaches its limits. Therefore, it makes no sense to have checked arithmetic in such a place.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_2	/OriginCash/OriginCash/Validator.sol	L55 - L55	*Pending Fix*

/OriginCash/OriginCash/Validator.sol L55 - L55

```
54            // Iterate to find the last dash ('-') in the input string
55            for (uint256 i = 0; i < strLength; ++i) {
56                if (strBytes[i] == "-") {
57                    lastDash = int256(i);
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_3	/OriginCash/OriginCash/Validator.sol	L72 - L72	*Pending Fix*

/OriginCash/OriginCash/Validator.sol L72 - L72

```
71            // Loop through the characters and store valid ones
72            for (uint256 i = start; i < strLength && j < 128; ++i) {
73                bytes1 tempChar = strBytes[i];
74                // Allow only alphanumeric characters (0-9, A-Z, a-z)
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_4	/OriginCash/OriginCash/Validator.sol	L81 - L81	*Pending Fix*

/OriginCash/OriginCash/Validator.sol L81 - L81

```
80                    tempResult[j] = tempChar;
81                    ++j;
82                }
83            }
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_5	/OriginCash/OriginCash/OriginCash.sol	L219 - L219	*Pending Fix*

/OriginCash/OriginCash/OriginCash.sol L219 - L219

```
218         Deposit[] memory latestDeposits = new Deposit[](count);
219         for (uint256 i = 0; i < count; ++i) {
220             latestDeposits[i] = deposits[depoLength - count + i];
221         }
```

Issue Type

USE BYTES.CONCAT() INSTEAD OF ABI.ENCODEPACKED

S. No.	Severity	Detection Method	Instances
G012	Gas	Automated	2

Description

The contract is found to use `abi.encodePacked` for concatenating byte variables, which is less gas-efficient compared to using `bytes.concat`. When concatenation isn't used for hashing operations, preferring `bytes.concat` can result in more optimized and cost-effective gas consumption.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_36	/OriginCash/OriginCash/Validator.sol	L17 - L17	*Pending Fix*

/OriginCash/OriginCash/Validator.sol L17 - L17

```
16                 keccak256(
17                     abi.encodePacked(_amount, _wallet, _commitment)
18                 );
19         }
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_37	/OriginCash/OriginCash/Validator.sol	L24 - L24	*Pending Fix*

/OriginCash/OriginCash/Validator.sol L24 - L24

```
23            bytes32 commitment = keccak256(
24                abi.encodePacked(block.timestamp, msg.sender, block.prevrandao)
25            );
26            return commitment;
```

Issue Type

USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

S. No.	Severity	Detection Method	Instances
G013	Gas	Automated	2

Description

In Solidity, efficient use of gas is paramount to ensure cost-effective execution on the Ethereum blockchain. Gas can be optimized when obtaining contract balance by using `selfbalance()` rather than `address(this).balance` because it bypasses gas costs and refunds, which are not required for obtaining the contract's balance.

Bug ID	File Location	Line No.	Action Taken
SSP_24004_44	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L42 - L42	*Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L42 - L42

```
41          require(
42              address(this).balance >= _amountAfterFee + _fee,
43              "Insufficient funds"
44          );
```

Bug ID	File Location	Line No.	Action Taken
SSP_24004_45	/OriginCash/Ori...inCash/DynamicShieldWallet.sol	L75 - L75	⚠ *Pending Fix*

/OriginCash/OriginCash/DynamicShieldWallet.sol L75 - L75

```
74          require(_recipient != address(0), "Invalid recipient address");
75          uint256 balance = address(this).balance;
76          (bool success, ) = _recipient.call{value: balance}("");
77          require(success, "Transfer failed");
```

05. Scan History

Critical High Medium Low Informational Gas

No	Date	Security Score	Scan Overview
1.	2024-09-21	73.08	5 0 3 28 25 36

06. Disclaimer

The Reports neither endorse nor condemn any specific project or team, nor do they guarantee the security of any specific project. The contents of this report do not, and should not be interpreted as having any bearing on, the economics of tokens, token sales, or any other goods, services, or assets.

The security audit is not meant to replace functional testing done before a software release.

There is no warranty that all possible security issues of a particular smart contract(s) will be found by the tool, i.e., It is not guaranteed that there will not be any further findings based solely on the results of this evaluation.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. There is no warranty or representation made by this report to any Third Party in regards to the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business.

In no way should a third party use these reports to make any decisions about buying or selling a token, product, service, or any other asset. It should be noted that this report is not investment advice, is not intended to be relied on as investment advice, and has no endorsement of this project or team. It does not serve as a guarantee as to the project's absolute security.

The assessment provided by SolidityScan is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. SolidityScan owes no duty to any third party by virtue of publishing these Reports.

As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits including manual audit and a public bug bounty program to ensure the security of the smart contracts.