Portkey Finance - audit

Security Assessment

CertiK Assessed on Dec 7th, 2024

CertiK Assessed on Dec 7th, 2024

Portkey Finance - audit

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Wallet	Aelf	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
C#	Delivered on 12/07/2024	N/A

CODEBASE

https://github.com/Portkey-Wallet/portkey-contracts/tree/053b4d9f69ff84ccab6b925765c7b058c9f6ccf0/contract/Portkey.Contracts.CA\ https://github.com/Portkey-Wallet/portkey-

View All in Codebase Page

Vulnerability Summary

26	24	1	0	1	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
2	Major	1 Resolved, 1 Mitigated	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
2	Medium	1 Resolved, 1 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
5	Minor	5 Resolved	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
17	Informational	17 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | PORTKEY FINANCE - AUDIT

Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

Findings

CCC-06 : `SyncHolderInfo()` function Vulnerable to Replay Attacks Due to Missing Execution Checks

GLOBAL-01 : Centralization Related Risks

CAT-01 : Unable to transfer funds when the balance exceeds the specified threshold

PCP-02 : Third-Party Dependency

CAT-03 : Flawed Logic in Default Transfer Limit Computation due to Uninitialized Variable

CCC-04 : Missing check for `IndentifierHash`: all logins have been unset when unbound

CCC-09 : Flawed Guardian Authentication Logic in `ValidateLoginGuardian()` function

CCG-01 : Flawed Guardian Removal Mechanism

CCG-03 : Lack of Verifier ID Validation in Guardian Addition Processes

CAA-01 : Missing Check for `Address` and `ExtraData` of `ManagerInfo`

CAD-01 : Unrestricted Access to Register Project Delegatee

CAH-01 : Inconsistency in the return values between the `CheckVerifierSignatureAndDataWithCreateChainId()` and the `CheckOnCreateChain()` functions

CAM-01 : Manager information can be overridden if the address is the same, and the `ExtraData` is not the same

CAM-02 : Inadequate Case Sensitivity

CAP-01 : Add `CAServerUpdated` event to distinguish update operation

CAT-02 : The `IsTransferSecurity()` function lacks the `Symbol` parameter

CAV-01 : Incapability to Update Image URL for Verifier Servers in `AddVerifierServerEndPoints()` function

CCC-01 : The `UpgradeSecondaryDelegatee()` function should be called before removing `ManagerInfos`

CCC-02 : Flawed `NotLoginGuardians` Verification Mechanism

CCC-05 : The different between the `SyncHolderInfo()` and `RemoveGuardian()` functions when removing guardians

CCC-08 : Unrestricted Access to Holder Information Synchronization Functions

CCP-01 : discrepancy in Guardian Validation Procedure for Add/Update Operations

CCS-01 : Unnecessary Code Duplication Found in `SyncHolderInfo` Function

PCP-03 : Incomplete `Symbol` Validation

PCP-04 : Presence of Unutilized Functions in Contract Code

PCP-06 : Missing Emit Events

Optimizations

CAH-02 : Inefficiency Due to Redundant Key Calculation

CAV-02 : Take no action if the count of the verifier server list is 0

PCP-01 : Redundancy in Address Conversion Leading to Inefficiencies

Appendix

Disclaimer

CODEBASE | PORTKEY FINANCE - AUDIT

Repository

https://github.com/Portkey-Wallet/portkey-contracts/tree/053b4d9f69ff84ccab6b925765c7b058c9f6ccf0/contract/Portkey.Contracts.CA\ https://github.com/Portkey-Wallet/portkey-contracts/tree/edc220af5d468817976e6ebbbd5ade1f6fb2ed9d/contract/Portkey.Contracts.CA\ https://github.com/Portkey-Wallet/portkey-contracts/commit/557023bfb087f2379d013473c3e5279181e259f7

AUDIT SCOPE | PORTKEY FINANCE - AUDIT

17 files audited ● 2 files with Acknowledged findings ● 7 files with Resolved findings ● 8 files without findings

ID	Repo	File	SHA256 Checksum
● CAP	Portkey-Wallet/portkey-contracts	CAContract_CAServers.cs	2fbf3d9e267a8265bf0d5780eda39a7132cbd799f1226e714a79a35810de9740
● CAV	Portkey-Wallet/portkey-contracts	CAContract_Verifiers.cs	27160b4b744496478416283a644d8cd8e22c1d552fbc27a2d70c98fff5c57616
● CAA	Portkey-Wallet/portkey-contracts	CAContract_Actions.cs	35983aeaef69fa8ca3239e321c4393b2c7e7b4bc6fbcaabe7b5ffee96ca98792
● CCG	Portkey-Wallet/portkey-contracts	CAContract_Guardians.cs	b28354358bef12118e86bbbf8039fabee8db448e360734ff955df764278943df
● CAH	Portkey-Wallet/portkey-contracts	CAContract_Helpers.cs	2288e296124e3b13570b4c0a0f1ff98ed37ea866d5f176021684b90fb5ab7371
● CAL	Portkey-Wallet/portkey-contracts	CAContract_LoginGuardianAccount.cs	e6b31301f27184223bbc59435b9af5a992e8d73e27c0324e1503595a5e896af5
● CAM	Portkey-Wallet/portkey-contracts	CAContract_Managers.cs	eb2d39ec05c463bc969756ae2b314804f8b7350c353182185e3ef557868c50f4
● CCC	Portkey-Wallet/portkey-contracts	CAContract_SyncCAHolder.cs	f4832d6d7349a905f264325a921464f05742179d7ac1b00bf4b7a4b9aa28754c
● CAT	Portkey-Wallet/portkey-contracts	CAContract_TransferLimit.cs	20638bcbfdda439c3d0ece0ab7de271c0ec620423730f71f75e9acb9baf4e901

ID	Repo	File	SHA256 Checksum
CAC	Portkey-Wallet/portkey-contracts	CAContractConstants.cs	263a1bd7697bc20e8697a348e5bb4575aaf6226da93201545d794eb68cc31de4
CAR	Portkey-Wallet/portkey-contracts	CAContractReferenceState.cs	c8e1360297e1e1339f713ca2df0475c3ff00c532f2e6e672b4f68f5fe3f50352
CAS	Portkey-Wallet/portkey-contracts	CAContractState.cs	05777ef156df25b6ffbabcb7d8466314d7aca7d96b637eec39c1560f2dcbcce1
CAW	Portkey-Wallet/portkey-contracts	CAContract_Controller.cs	199fd920aec2e9feafb6b210e49825a03f129f4818ca9a19001a63e49c6c10b4
CAG	Portkey-Wallet/portkey-contracts	CAContract_GetHolderInfo.cs	282c23e069cf1cd698b3ef86e6e01cfdbabf5f94a7da566024ae8439781a9cfb
CAB	Portkey-Wallet/portkey-contracts	CAContract_StrategyBase.cs	165d961986876aebce31f97357f63049d56a5abba392ea4992d8ae1015263817
CCS	Portkey-Wallet/portkey-contracts	CAContract_StrategyHelper.cs	e710391a2ced6a5d3748c448c5af189d98563448f7d922743482cc0031ccc2bf
CAI	Portkey-Wallet/portkey-contracts	CAContract_StrategyImpl.cs	0daf67db56150e67ca4be80b4b3a0894bba5e8e833c69136346581e38b6fb5a3

APPROACH & METHODS | PORTKEY FINANCE - AUDIT

This report has been prepared for Portkey to discover issues and vulnerabilities in the source code of the Portkey Finance - audit project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | PORTKEY FINANCE - AUDIT

26	0	2	2	5	17
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Portkey Finance - audit. Through this audit, we have uncovered 26 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
CCC-06	`SyncHolderInfo()` Function Vulnerable To Replay Attacks Due To Missing Execution Checks	Logical Issue	Major	Resolved
GLOBAL-01	**Centralization Related Risks**	**Centralization**	**Major**	**Mitigated**
CAT-01	Unable To Transfer Funds When The Balance Exceeds The Specified Threshold	Logical Issue	Medium	Resolved
PCP-02	Third-Party Dependency	Design Issue	Medium	Acknowledged
CAT-03	Flawed Logic In Default Transfer Limit Computation Due To Uninitialized Variable	Logical Issue	Minor	Resolved
CCC-04	Missing Check For `IndentifierHash` : All Logins Have Been Unset When Unbound	Volatile Code	Minor	Resolved
CCC-09	Flawed Guardian Authentication Logic In `ValidateLoginGuardian()` Function	Volatile Code	Minor	Resolved
CCG-01	Flawed Guardian Removal Mechanism	Logical Issue	Minor	Resolved
CCG-03	Lack Of Verifier ID Validation In Guardian Addition Processes	Volatile Code	Minor	Resolved
CAA-01	Missing Check For `Address` And `ExtraData` Of `ManagerInfo`	Volatile Code	Informational	Resolved

ID	Title	Category	Severity	Status
CAD-01	Unrestricted Access To Register Project Delegatee	Access Control	Informational	Resolved
CAH-01	Inconsistency In The Return Values Between The `CheckVerifierSignatureAndDataWithCreateChainId()` And The `CheckOnCreateChain()` Functions	Logical Issue	Informational	Resolved
CAM-01	Manager Information Can Be Overridden If The Address Is The Same, And The `ExtraData` Is Not The Same	Logical Issue	Informational	Resolved
CAM-02	Inadequate Case Sensitivity	Volatile Code	Informational	Resolved
CAP-01	Add `CAServerUpdated` Event To Distinguish Update Operation	Coding Style	Informational	Resolved
CAT-02	The `IsTransferSecurity()` Function Lacks The `Symbol` Parameter	Logical Issue	Informational	Resolved
CAV-01	Incapability To Update Image URL For Verifier Servers In `AddVerifierServerEndPoints()` Function	Logical Issue	Informational	Resolved
CCC-01	The `UpgradeSecondaryDelegatee()` Function Should Be Called Before Removing `ManagerInfos`	Logical Issue	Informational	Resolved
CCC-02	Flawed `NotLoginGuardians` Verification Mechanism	Volatile Code	Informational	Resolved
CCC-05	The Different Between The `SyncHolderInfo()` And `RemoveGuardian()` Functions When Removing Guardians	Logical Issue	Informational	Resolved
CCC-08	Unrestricted Access To Holder Information Synchronization Functions	Access Control	Informational	Resolved
CCP-01	Discrepancy In Guardian Validation Procedure For Add/Update Operations	Volatile Code	Informational	Resolved

ID	Title	Category	Severity	Status
CCS-01	Unnecessary Code Duplication Found In `SyncHolderInfo` Function	Coding Style	Informational	Resolved
PCP-03	Incomplete `Symbol` Validation	Volatile Code	Informational	Resolved
PCP-04	Presence Of Unutilized Functions In Contract Code	Coding Style	Informational	Resolved
PCP-06	Missing Emit Events	Coding Style	Informational	Resolved

CCC-06 `SyncHolderInfo()` FUNCTION VULNERABLE TO REPLAY ATTACKS DUE TO MISSING EXECUTION CHECKS

Category	Severity	Location	Status
Logical Issue	Major	CAContract_SyncCAHolder.cs (1.4.9): 104	Resolved

Description

The smart contract's `SyncHolderInfo()` method has been identified as vulnerable to replay attacks. It lacks the necessary validation mechanisms to ascertain that a `verificationTransactionInfo` is not processed more than once. If exploited, this vulnerability permits an attacker to repeatedly execute the same transaction information, leading to the overwriting of significant holder-specific data structures including `ManagerInfos`, `GuardianList`, and `delegators`. This overwriting introduces discrepancies between the contract's state and the actual holder's data, potentially impacting data integrity.

Recommendation

To mitigate this vulnerability, it is imperative to establish a robust tracking and verification protocol for `verificationTransactionInfo` execution. This may include the creation of a mapping structure to store processed transaction identifiers or the implementation of a nonce scheme that conclusively ensures each `verificationTransactionInfo` is singularly processed, preserving the sanctity of the holder's data within the smart contract's ecosystem.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

GLOBAL-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	**Major**		**Mitigated**

Description

In the contract `CAContract` the role `Admin` has authority over the functions

- SetContractDelegationFee()
- SetSecondaryDelegationFee()
- SetCAContractAddresses()
- AddCreatorController()
- RemoveCreatorController()
- AddServerController()
- RemoveServerController()
- ChangeAdmin()
- SetForbiddenForwardCallContractMethod()
- SetDefaultTokenTransferLimit()
- SetCheckChainIdInSignatureEnabled()
- SetTransferSecurityThreshold()
- AddRemovedToCurrentVerifierIdMapper()

Any compromise to the `Admin` account may allow the hacker to take advantage of this authority and

- set contract delegation fee
- set secondary delegation fee
- set CA contract addresses
- add a creator controller to the creator controllers
- remove the creator controller from the creator controllers
- add a server controller to the server controllers
- remove the server controller from the server controllers
- change the admin
- set forbidden forward call contract method
- set default token transfer limit
- set `CheckChainIdInSignatureEnabled`
- set transfer security threshold

- add the removed verifier server to the current verifier ID mapper

In the contract `CAContract` the role `CreatorControllers` has authority over the functions

- CreateCAHolder()

Any compromise to the `CreatorControllers` account may allow the hacker to take advantage of this authority and

- create a CA holder

In the contract `CAContract` the role `ServerControllers` has authority over the functions

- AddCAServer()
- RemoveCAServer()
- AddVerifierServerEndPoints()
- RemoveVerifierServerEndPoints()

Any compromise to the `ServerControllers` account may allow the hacker to take advantage of this authority and

- add a CA server to the server list
- remove the CA server from the server list
- add a verifier server to the verifier server list or update the endpoints of the verifier server
- remove the endpoints of the verifier server

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Portkey Team, 12/07/2024]:

1. Admin Account and ServerControllers have been removed, CreatorControllers are only used in user registration scenarios to prevent malicious attacks such as DDoS attacks.
2. We have switched the administrator account to the organizational address of TMRWDAO(https://tmrwdao.com/network-dao/organization?chainId=AELF). The organization address is 2Cz9qDrYN4azJ2K8CMytcNECWdiA2vyvDgvG4rbo1UaYudGsBb
3. Please refer to this link for detailed parameters of the organization(https://aelfscan.io/AELF/tx/8ee220bbc13ae32bef630dc746e67fd2f04cbd0ec73f7f5bf4f17e381655f2d2).The five members with voting rights are three accounts from Singapore and two accounts from China
   - ELF_2WSGjFpwNLTFT6Snwj9MJZ2shRzWrd9Kf7KxwwmUPHZvkbJRKq_AELF
   - ELF_2XWSnY81Ti5qdQULyEf4v5NHA5caHvsdUWMKXs892BBy7trpdw_AELF
   - ELF_FDzDh61kUTnb7TL9ejdpfTm71rjRv5e5ioXmrknh7K4Qb2RUr_AELF
   - ELF_w1s8hUonC8E1v2SkaNMnjwcRadw7xD8EythpnYUABFf367XJu_AELF
   - ELF_2Jyg891ASxMnQu3GN1Fa6E1pFWpnTXiX4GeCbzzC5ZjCw2fAa1_AELF
4. Changes have been reflected in the commit hash:https://github.com/Portkey-Wallet/portkey-contracts/commit/557023bfb087f2379d013473c3e5279181e259f7

```
✔ Fetching contract successfully!
? Pick up a contract method: GetOrganizationAddress
✔ Calling method successfully!
AElf [Info]:
Result:
{
  "address": "2Cz9qDrYN4azJ2K8CMytcNECWdiA2vyvDgvG4rbo1UaYudGsBb"
}
✔ Succeed!
```

[CertiK, 12/07/2024]: While the use of [multi-sig and/or timelock] strategy has indeed reduced the risk, it's crucial to note that it has not completely eliminated it. CertiK strongly encourages the project team periodically revisit the private key security management of all above-listed addresses.

CAT-01 UNABLE TO TRANSFER FUNDS WHEN THE BALANCE EXCEEDS THE SPECIFIED THRESHOLD

Category	Severity	Location	Status
Logical Issue	Medium	CAContract_TransferLimit.cs (1.4.9): 251	Resolved

Description

The `IsTransferSecurity()` function evaluates transfer security conditions using the `TransferSecurityThresholdList`. If the `TransferSecurityThresholdList` is not set or the count of guardians exceeds the specified threshold, the function returns `true`, signifying that the transfer is secure. Notably, when an account has a token balance that surpasses the defined threshold, the function unexpectedly returns `false`. This questionable behavior creates a scenario where legitimate and potentially high-valued accounts are barred from performing transfers, likely causing disruptions to the users' ability to interact with the contract's features.

```
 private bool IsTransferSecurity(Hash caHash)
    {
        var holderInfo = State.HolderInfoMap[caHash];
        var guardianAmount = holderInfo.GuardianList?.Guardians?.Count;
        if (State.TransferSecurityThresholdList?.Value != null)
        {
            foreach (var securityThreshold in
State.TransferSecurityThresholdList.Value.TransferSecurityThresholds)
            {
                if (guardianAmount > securityThreshold.GuardianThreshold) continue;
                var balance = GetTokenBalance(securityThreshold.Symbol,
                    Context.ConvertVirtualAddressToContractAddress(caHash));
                if (balance.Balance >= securityThreshold.BalanceThreshold) return
false;
            }
        }

        return true;
    }
```

Scenario

When invoking the function `InitTransferLimitTest()`, the holder's balance is set to `1000000000000`, the count of guardians is `1`, the `BalanceThreshold` is `10000`, and the `GuardianThreshold` is `2`. However, the `GetTransferSecurityCheckResult()` function returns `false`.

Proof of Concept

```
[Fact]
    public async Task GetTransferSecurityCheckResultOutputTest()
    {
        await InitTransferLimitTest();

        var result = await CaContractStub.GetTransferSecurityCheckResult.CallAsync(
            new GetTransferSecurityCheckResultInput()
            {
                 CaHash = _transferLimitTestCaHash,
            });

        result.IsSecurity.ShouldBeFalse();
    }
```

Recommendation

The auditor wants to check with the team to determine whether the behavior of the `IsTransferSecurity()` function conforms to the original project specifications. Subsequently, assuming a discrepancy is identified between the intended and actual behavior, a redesign of the logic should be undertaken to modify the condition checks within the function. The goal should be to achieve a proper balance between maintaining security and facilitating user transactions that meet the security standards set forth in the smart contract.

Alleviation

[Portkey Team, 04/11/2024]:

we will make adjustments to the contract to restrict the range in which the admin can set the number of guardians. The minimum value will be set to 1, and the maximum value will be the total number of verifiers. This is done to prevent users from being unable to transfer funds due to not being able to meet the required number of guardians set.

The client revised the code and resolved this issue in commit : 8529efb3f36be56c44853536de1f0483eccfa1c5

PCP-02 | THIRD-PARTY DEPENDENCY

Category	Severity	Location	Status
Design Issue	Medium	CAContract_CAServers.cs (1.4.9): 7; CAContract_Verifiers.cs (1.4.9): 10	Acknowledged

Description

The contract serves as the underlying entity to interact with one or more third-party servers. The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. The responsibility of checking the verifier signature and data lies with the third-party servers. However, in real-world scenarios, third parties can be compromised, potentially resulting in challenges such as the inability to create a CA holder, manage guardians, implement social recovery, remove other manager info, execute manager approvals, and set transfer limits.

Recommendation

The auditors understood that the business logic requires interaction with third parties. It is recommended for the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Portkey Team, 01/11/2024]: It's already supported in a VerifyDao plan. Monitoring systems are in place but have not been publicly disclosed.

CAT-03 FLAWED LOGIC IN DEFAULT TRANSFER LIMIT COMPUTATION DUE TO UNINITIALIZED VARIABLE

Category	Severity	Location	Status
Logical Issue	Minor	CAContract_TransferLimit.cs (1.4.9): 155~156, 160~161	Resolved

Description

The `GetDefaultTransferLimit()` function is designed to compute transfer limits for token transactions. This function looks up the `TokenDefaultTransferLimit` for user-specific limits, and if not found, should revert to `TokenInitialTransferLimit`. However, it is observed that the contract contains no setter method for `TokenInitialTransferLimit`, leaving it with a default uninitialized value of zero. This oversight means that the intended fallback hierarchy (user default → initial limit → constant amount) fails to work correctly, and the function ends up falling back directly to the hardcoded constant `TokenDefaultTransferLimitAmount`. This could lead to unexpected transfer limit configurations, potentially undermining the transfer limitation feature's effectiveness and causing operational discrepancies.

```
private TransferLimit GetDefaultTransferLimit(string symbol)
    {
        return new TransferLimit
        {
            SingleLimit = State.TokenDefaultTransferLimit[symbol] != null
                ? State.TokenDefaultTransferLimit[symbol].SingleLimit
                : State.TokenInitialTransferLimit.Value > 0
                    ? State.TokenInitialTransferLimit.Value
                    : CAContractConstants.TokenDefaultTransferLimitAmount,
            DayLimit = State.TokenDefaultTransferLimit[symbol] != null
                ? State.TokenDefaultTransferLimit[symbol].DayLimit
                : State.TokenInitialTransferLimit.Value > 0
                    ? State.TokenInitialTransferLimit.Value
                    : CAContractConstants.TokenDefaultTransferLimitAmount
        };
    }
```

Recommendation

Modify the contract's logic by introducing a setter function to facilitate the initialization of `TokenInitialTransferLimit`. Ensure that all transfer limits adhere the underlying business rules. If `TokenInitialTransferLimit` is unnecessary, refactor the function to streamline its operation and eliminate reliance on an irrelevant variable.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CCC-04 MISSING CHECK FOR `IndentifierHash` : ALL LOGINS HAVE BEEN UNSET WHEN UNBOUND

Category	Severity	Location	Status
Volatile Code	Minor	CAContract_SyncCAHolder.cs (1.4.9): 231	Resolved

Description

`UnsetGuardianForLogin()` correctly unsets guardians, checking their login states before removal. However, this check is missing in `SyncHolderInfo()`, allowing for direct removals which could lead to erroneous state within the `GuardianMap`.

Recommendation

Revise `SyncHolderInfo()` to include the missing validation step, thereby aligning it with the established guardian unset procedure and improving the contract's reliability.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CCC-09 FLAWED GUARDIAN AUTHENTICATION LOGIC IN `ValidateLoginGuardian()` FUNCTION

Category	Severity	Location	Status
Volatile Code	Minor	CAContract_SyncCAHolder.cs (1.4.9): 51	Resolved

Description

The function `ValidateLoginGuardian()` solely validates that the number of Guardians in the login state of the holder matches the number of input Guardians and exists in the holder's list of Guardians.

However, a problem arises when the holder has two login state identifiers, `IdentifierHash1` and `IdentifierHash2`, but the input Guardians are `IdentifierHash1` and `IdentifierHash1`. This scenario, which excludes `IdentifierHash2`, passes validation as usual.

Recommendation

Ensure that the IdentifierHash corresponding to the input Guardian exists for all Guardians in the holder's login state and verify that the count is the same.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CCG-01 | FLAWED GUARDIAN REMOVAL MECHANISM

Category	Severity	Location	Status
Logical Issue	Minor	CAContract_Guardians.cs (1.4.9): 85	Resolved

Description

The `RemoveGuardian()` function is intended to fully unassign a guardian by deleting their data from all relevant mappings. The function correctly deletes the guardian from the `LoginGuardianMap`, but it doesn't remove the guardian from the `GuardianMap`. This partial deletion creates a discrepancy in smart contract data integrity.

Recommendation

Update the smart contract code to properly remove all data related to the guardian in question. Specifically, the `RemoveGuardian()` function should be refined to remove any associations within the `GuardianMap` whenever a guardian is removed from the system.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CCG-03 LACK OF VERIFIER ID VALIDATION IN GUARDIAN ADDITION PROCESSES

Category	Severity	Location	Status
Volatile Code	Minor	CAContract_Guardians.cs (1.4.9): 24	Resolved

Description

The `AddGuardian()` function in the smart contract is designed to handle guardian records. However, they currently lack validation for the verifier ID provided, which poses a risk of accepting the invalid ID and potentially disrupting the guardian approval process. The intended functionality is to uniquely identify guardians and accurately count those approved, but the absence of proper ID validation jeopardizes this operation.

Recommendation

Incorporate a check mechanism for verifier ID in `AddGuardian()` function to ensure input integrity.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CAA-01 | MISSING CHECK FOR `Address` AND `ExtraData` OF `ManagerInfo`

Category	Severity	Location	Status
Volatile Code	Informational	CAContract_Actions.cs (1.4.9): 48	Resolved

Description

The `CreateCAHolder()` does not enforce the same input validation as the `AddManagerInfo()` function. The omission of null or white space checks on the parameters `Address` and `ExtraData` in the `CreateCAHolder()` function could be exploited by a bad actor to introduce invalid data into the contract state, leading to potential vulnerabilities and exploits within the contract's logic.

Recommendation

To prevent such occurrences and to ensure the contract's integrity, it is strongly recommended to implement a validation mechanism in the `CreateCAHolder()` function that mirrors the parameter checks performed in the `AddManagerInfo()` function.

Alleviation

[Portkey Team, 01/11/2024]: Add not-null check for Address. ExtraData is not key data.

CAD-01 UNRESTRICTED ACCESS TO REGISTER PROJECT DELEGATEE

Category	Severity	Location	Status
Access Control	Informational	CAContract_ProjectDelegatee.cs (fix): 27	Resolved

Description

The `RegisterProjectDelegatee()` function is designed to register project delegatees. It was found that it is currently callable by any user without restrictions. The function should be restricted to authorized entities to prevent malicious actors from becoming the project controller. They could potentially add/remove the delegatees, set the controller and the signer, and withdraw delegatee tokens.

Recommendation

To secure the integrity of the smart contract, access to this function should be restricted through the implementation of permission checks.

Alleviation

[Portkey Team, 01/23/2024]: This is related to business logic.

CAH-01 INCONSISTENCY IN THE RETURN VALUES BETWEEN THE `CheckVerifierSignatureAndDataWithCreateChainId()` AND THE `CheckOnCreateChain()` FUNCTIONS

Category	Severity	Location	Status
Logical Issue	Informational	CAContract_Helpers.cs (1.4.9): 187~190	Resolved

Description

The `CheckOnCreateChain()` function verifies that the holder has guardians, and the count of guardians must be greater than 0. Additionally, if the create chain ID of the holder is greater than 0, it must be equal to the current chain ID. If any of these conditions are not met, the function returns `false`. If the return value is `false`, it indicates that the checked data does not meet the required conditions. Despite `CheckOnCreateChain()` being designed to return `false` when the required conditions of having non-zero guardians and matching chain IDs are not met, the incorrect `if` condition counterintuitively returns `true`. This contradictory behavior undermines the guardian validation process and poses a risk of incorrect signature verification.

```
 if (operationTypeName == nameof(OperationType.AddGuardian).ToLower() &&
!CheckOnCreateChain(State.HolderInfoMap[caHash]))
        {
            return true;
        }
```

```
    private bool CheckOnCreateChain(HolderInfo holderInfo)
    {
        if (holderInfo.GuardianList == null || holderInfo.GuardianList.Guardians ==
null ||
            holderInfo.GuardianList.Guardians.Count == 0)
        {
            return false;
        }
        return holderInfo.CreateChainId == Context.ChainId ||
holderInfo.CreateChainId == 0;
    }
```

Recommendation

To mitigate this issue, a code revision is necessary to align the return value with the actual result of the `CheckOnCreateChain()` function. Specifically, in scenarios where `CheckOnCreateChain()` indicates inadequacies in the guardian list or a mismatch in chain IDs, the response should be `false`.

Alleviation

[Portkey Team, 01/11/2024]: This is a verification meant to avoid detection during the acceleration of guardian synchronization. It's related to business logic and designed to be like this, so it will not be handled.

CAM-01 MANAGER INFORMATION CAN BE OVERRIDDEN IF THE ADDRESS IS THE SAME, AND THE `ExtraData` IS NOT THE SAME

Category	Severity	Location	Status
Logical Issue	Informational	CAContract_Managers.cs (1.4.9): 188	Resolved

Description

In the `UpdateManagerInfos()` function, the deduplication process is undermined by the current implementation at line 188 using the `Distinct()` method, which is tasked with removing duplicate `ManagerInfos` from the contract's storage. It requires both `Address` and `ExtraData` to match exactly. This approach allows for scenarios where an `Address` that already exists with one set of `ExtraData` can be appended again with different `ExtraData`, enabling the latter to erroneously overwrite the former.

Recommendation

To mitigate this security risk, an enhanced duplication verification process should be incorporated into the `UpdateManagerInfos()` function. The focus of this process should be to identify and reject new entries that contain an `Address` already present in the `ManagerInfos`, disregarding the `ExtraData`.

Alleviation

[Portkey Team, 01/11/2024]:

This is related to business logic and will not be handled.

ExtraData update is needed in such a scenario.

CAM-02 | INADEQUATE CASE SENSITIVITY

Category	Severity	Location	Status
Volatile Code	Informational	CAContract_Managers.cs (1.4.9): 248	Resolved

Description

The smart contract contains a vulnerability in the feature that compares `input.MethodName` for authorizing token transfers. Due to the lack of strict case sensitivity during comparison with `nameof(State.TokenContract.Transfer)`, it is possible to circumvent the security mechanism meant to validate transfer requests.

Recommendation

To rectify this, both `input.MethodName` and `nameof(State.TokenContract.Transfer)` should be converted to lowercase prior to comparison to avoid discrepancies due to case sensitivity.

Alleviation

[Portkey Team, 01/11/2024]: Because method names won't match, even after circumventing the verification, it is not possible to execute methods within the Token contract.

CAP-01 ADD `CAServerUpdated` EVENT TO DISTINGUISH UPDATE OPERATION

Category	Severity	Location	Status
Coding Style	Informational	CAContract_CAServers.cs (1.4.9): 29~36	Resolved

Description

Upon inspection, it has been identified that the `AddCAServer()` function inadequately emits only a single type of event, the `CAServerAdded` event, which does not account for the distinction between the addition of new servers or the update of existing ones. As the function is responsible for both operations, this lack of event granularity may cause ambiguity in state changes within the contract.

Recommendation

To rectify this, the smart contract should be revised to include a `CAServerUpdated` event that is specifically emitted after the successful update of an existing CA server entry, thereby ensuring clarity and precision in event-driven mechanisms and log analysis.

CAT-02 THE `IsTransferSecurity()` FUNCTION LACKS THE `Symbol` PARAMETER

Category	Severity	Location	Status
Logical Issue	Informational	CAContract_TransferLimit.cs (1.4.9): 240	Resolved

Description

The function `IsTransferSecurity()` is responsible for validating the security of token transfers within a smart contract, referencing the `TransferSecurityThresholdList`. This list outlines specific parameters using `Symbol`, `GuardianThreshold`, and `BalanceThreshold` to formulate security checks. However, the current implementation of `IsTransferSecurity()` lacks the `Symbol` parameter, which is crucial for correctly matching the token type against its security criteria within `TransferSecurityThresholdList`. The misalignment in the validation logic could lead to incorrect execution of transfers, where valid transactions might be rejected or invalid transactions accepted, based on the security thresholds intended for different tokens.

```
 private bool IsTransferSecurity(Hash caHash)
    {
        var holderInfo = State.HolderInfoMap[caHash];
        var guardianAmount = holderInfo.GuardianList?.Guardians?.Count;
        if (State.TransferSecurityThresholdList?.Value != null)
        {
            foreach (var securityThreshold in
State.TransferSecurityThresholdList.Value.TransferSecurityThresholds)
            {
                if (guardianAmount > securityThreshold.GuardianThreshold) continue;
                var balance = GetTokenBalance(securityThreshold.Symbol,
                    Context.ConvertVirtualAddressToContractAddress(caHash));
                if (balance.Balance >= securityThreshold.BalanceThreshold) return
false;
            }
        }

        return true;
    }
```

Scenario

When invoking the function `InitTransferLimitTest()`, the holder can receive `1000000000000` tokens with the `ELF` symbol, and the count of guardians is `1`. The `TransferSecurityThresholdList` specifies the `Symbol` as `CPU`, the `GuardianThreshold` as `2`, and the `BalanceThreshold` as `10000`. However, the function `GetTransferSecurityCheckResult()` returns `true` because the user's balance of `CPU` is 0.

Proof of Concept

```
[Fact]
    public async Task GetTransferSecurityCheckResultOutputTest()
    {
        await InitTransferLimitTest();

        var result = await CaContractStub.GetTransferSecurityCheckResult.CallAsync(
            new GetTransferSecurityCheckResultInput()
            {
                 CaHash = _transferLimitTestCaHash,
            });

        result.IsSecurity.ShouldBeTrue();
    }

    private async Task InitTransferSecurityBalanceThreshold()
    {
        await CaContractStub.SetTransferSecurityThreshold.SendAsync(new
SetTransferSecurityThresholdInput
        {
            TransferSecurityThreshold = new TransferSecurityThreshold()
            {
                Symbol = "CPU",
                GuardianThreshold = 2,
                BalanceThreshold = 10000
            }
        });
    }
```

Recommendation

To mitigate the vulnerability and streamline the security validation process, we advise updating the `IsTransferSecurity()` function to require a `Symbol` parameter. This addition will ensure that the function accurately aligns the transfer request with the appropriate token security settings and prevents the failure of legitimate transfers.

Alleviation

[Portkey Team, 01/11/2024]: Portkey's security detection is based on the account dimension rather than on the symbol dimension. As long as one symbol does not meet the criteria, the account is considered unsafe.

CAV-01 INCAPABILITY TO UPDATE IMAGE URL FOR VERIFIER SERVERS IN `AddVerifierServerEndPoints()` FUNCTION

Category	Severity	Location	Status
Logical Issue	Informational	CAContract_Verifiers.cs (1.4.9): 56~81	Resolved

Description

The contract's `AddVerifierServerEndPoints()` function is used to manage the verifier servers' data, including endpoints, verifier addresses, and associated image URLs. Proper functioning requires the ability to update these details as servers evolve. Unfortunately, due to a coding oversight, once an `ImageUrl` is associated with a verifier server, it cannot be replaced or updated by any subsequent calls to this function. This static linkage could potentially cause misrepresentation of the verifier server if the `ImageUrl` becomes obsolete.

Recommendation

It is advised to refine the `AddVerifierServerEndPoints()` function, introducing additional logic to determine if the provided `ImageUrl` for an existing verifier server is different from the one already stored. If the `ImageUrl` has indeed changed, the function should be capable of updating it to reflect the new value, maintaining the integrity of the verifier server's information.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CCC-01 THE `UpgradeSecondaryDelegatee()` FUNCTION SHOULD BE CALLED BEFORE REMOVING `ManagerInfos`

Category	Severity	Location	Status
Logical Issue	Informational	CAContract_SyncCAHolder.cs (1.4.9): 132	Resolved

Description

The `UpgradeSecondaryDelegatee()` function is utilized for upgrading the secondary delegatee. Originally, the smart contract utilized the `SetContractDelegator()` for setting up delegation relationships. In the newer version, the method has been replaced by `SetSecondaryDelegator()`, which delegates authority via a holder's virtual address to the CA contract. An oversight was observed, as the update did not remove delegation ties of removed managers because the old relationship was not disbanded prior to the manager's removal. This caused a problematic situation where past managers retained their delegatory powers.

Recommendation

Ensure the contract upgrade procedures that mandate the invalidation of all existing delegations associated with the manager's virtual address by implementing a call to `UpgradeSecondaryDelegatee()`. This prior action must be taken before any manager listed in `ManagerInfos` is officially unlisted.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CCC-02 | FLAWED `NotLoginGuardians` VERIFICATION MECHANISM

Category	Severity	Location	Status
Volatile Code	Informational	CAContract_SyncCAHolder.cs (1.4.9): 13	Resolved

Description

In the current implementation of the `ValidateCAHolderInfoWithManagerInfosExists` function, there is an observed lack of validation logic for the `NotLoginGuardians`. The function does not enforce checks on the validity relating to both the guardian count and the login state of guardians. This gap in the logic could be exploited, potentially leading to discrepancies in CA holder and manager information that could compromise the intended features of the contract.

Recommendation

To mitigate the identified risk, we advise the addition of robust validation checks to the `ValidateCAHolderInfoWithManagerInfosExists` function. This should include both enumeration and login verification of `NotLoginGuardians` to protect against inconsistencies.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CCC-05 THE DIFFERENT BETWEEN THE `SyncHolderInfo()` AND `RemoveGuardian()` FUNCTIONS WHEN REMOVING GUARDIANS

Category	Severity	Location	Status
Logical Issue	Informational	CAContract_SyncCAHolder.cs (1.4.9): 162	Resolved

Description

In the current smart contract, the `SyncHolderInfo()` function does not have the same high level of integrity checks for guardian removal as the `RemoveGuardian()` function, specifically failing to validate guardian logins and to update correlating mappings.

Recommendation

Refactor the `SyncHolderInfo()` function to incorporate a validation mechanism and map updating process, mirroring the logic of the `RemoveGuardian()` function for guardian removals, thereby mitigating potential security issues.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CCC-08 UNRESTRICTED ACCESS TO HOLDER INFORMATION SYNCHRONIZATION FUNCTIONS

Category	Severity	Location	Status
Access Control	Informational	CAContract_SyncCAHolder.cs (1.4.9): 89, 98	Resolved

Description

The `SyncHolderInfos()` and `SyncHolderInfo()` are designed to update and maintain holder information. It was found that they are currently callable by any user without restrictions, posing a threat to the integrity of the holder data. These functions should be restricted to authorized entities to prevent malicious attempts at data manipulation.

Recommendation

To secure the integrity of the smart contract, access to these functions should be restricted through the implementation of permission checks.

Alleviation

[Portkey Team, 01/11/2024]:

Opt not to handle this.

Synchronization costs transaction fees and malicious actions can't forge any data, so it's pointless to do so.

Data may be potentially overwritten only when consecutive transactions involve operations such as AddGuardian, RemoveGuardian, SetLoginGuardian, UnsetLoginGuardian, AddManager, and RemoveManager. However, malicious actors are unable to predict user actions.

CCP-01 DISCREPANCY IN GUARDIAN VALIDATION PROCEDURE FOR ADD/UPDATE OPERATIONS

Category	Severity	Location	Status
Volatile Code	Informational	CAContract_Guardians.cs (fix): 22~25, 171~179	Resolved

Description

The smart contract's guardian management exhibits a inconsistent due to differing validation checks in `AddGuardian()` and `UpdateGuardian()`. The `UpdateGuardian()` checks three parameters, which may permit the bypass of checks that are present in the `AddGuardian()`, where `VerifierId` parameter is not verified. This misalignment can be exploited.

Recommendation

Apply consistent validation checks for `Type`, `IdentifierHash`, and `VerifierId` in both `AddGuardian()` and `UpdateGuardian()` to mitigate risks of unauthorized Guardian updates.

Alleviation

[Portkey Team, 01/23/2024]: This is related to business logic.

CCS-01 UNNECESSARY CODE DUPLICATION FOUND IN `SyncHolderInfo` FUNCTION

Category	Severity	Location	Status
Coding Style	Informational	CAContract_SyncCAHolder.cs (fix): 142~149, 173~181	Resolved

Description

An examination of the smart contract's `SyncHolderInfo()` function reveals that there is an identical piece of code that unnecessarily assigns a `CreateChainId` and initializes an empty `GuardianList`. This is repeated twice within the same function, which is an inefficient use of smart contract resources.

```
holderInfo.CreateChainId = transactionInput.CreateChainId;
    if (holderInfo.GuardianList == null)
    {
        holderInfo.GuardianList = new GuardianList
        {
            Guardians = { }
        };
    }
```

Recommendation

To improve the smart contract's efficiency, the duplicate code should be eliminated. This will reduce the gas cost for executing the function and also enhance the overall code quality.

Alleviation

The client revised the code and resolved this issue in commit : a5c0108661aa086cd2eaf20cc98054e9db3ff0cb

PCP-03 | INCOMPLETE `Symbol` VALIDATION

Category	Severity	Location	Status
Volatile Code	Informational	CAContract_Managers.cs (1.4.9): 253, 264, 282, 311; CAContract_TransferLimit.cs (1.4.9): 39, 196	Resolved

Description

The smart contract functions responsible for managing assets and setting transfer policies do not include checks for the existence of the `Symbol`. This oversight occurs in key functions including `ManagerTransfer()`, `ManagerTransferFrom()`, and others. The absence of `Symbol` validation could lead to transactions being processed without the proper asset identifier, creating vulnerabilities in asset security and management integrity.

Recommendation

To address this security concern, a validation requirement for the `Symbol` should be enforced within each affected function. This validation will help prevent errors associated with unverified asset identifiers.

Alleviation

The client revised the code and resolved this issue in commit : b676fdf2ba1b7173c2442e461e520738e1d2ac33

PCP-04 PRESENCE OF UNUTILIZED FUNCTIONS IN CONTRACT CODE

Category	Severity	Location	Status
Coding Style	Informational	CAContract_Helpers.cs (1.4.9): 226; CAContract_SyncCAHolder.cs (1.4.9): 303	Resolved

Description

An examination of the smart contract code identifies several functions that have been declared but are not called anywhere within the contract.

Recommendation

Conduct a thorough review of the contract to identify and eliminate any functions that are not actively serving a purpose. Removing these functions will help simplify the contract structure.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

PCP-06 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	Informational	CAContract_Actions.cs (1.4.9): 227, 251~252, 269; CAContract_TransferLimit.cs (1.4.9): 166~167	Resolved

Description

Functions that update state variables should emit relevant events as notifications.

Recommendation

We recommend adding events for state-changing actions, and emitting them in their relevant functions.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

OPTIMIZATIONS | PORTKEY FINANCE - AUDIT

ID	Title	Category	Severity	Status
CAH-02	Inefficiency Due To Redundant Key Calculation	Coding Style	Optimization	Resolved
CAV-02	Take No Action If The Count Of The Verifier Server List Is 0	Volatile Code	Optimization	Resolved
PCP-01	Redundancy In Address Conversion Leading To Inefficiencies	Gas Optimization	Optimization	Resolved

CAH-02 | INEFFICIENCY DUE TO REDUNDANT KEY CALCULATION

Category	Severity	Location	Status
Coding Style	Optimization	CAContract_Helpers.cs (1.4.9): 104, 178	Resolved

Description

Investigation of the smart contract indicates that there is an inefficiency resulting from a repeated calculation of a `key` variable. This calculation has previously been performed and thus the repeat instance identified in the given code snippet is unnecessary.

```
var key =
HashHelper.ComputeFrom(guardianInfo.VerificationInfo.Signature.ToByteArray());
```

Recommendation

To optimize the smart contract and avoid wasting computational power, the repetitive lines that recalculate the `key` should be removed from the codebase.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

CAV-02 TAKE NO ACTION IF THE COUNT OF THE VERIFIER SERVER LIST IS 0

Category	Severity	Location	Status
Volatile Code	Optimization	CAContract_Verifiers.cs (1.4.9): 106	Resolved

Description

In the smart contract's `RemoveVerifierServerEndPoints()` function, a distinction fails to be made between two scenarios: a `VerifiersServerList` that has not been initialized (`null`) and an initialized list that contains no elements (with a count of 0). The function correctly performs no action when the list is `null` ; however, when encountering an empty list, it should take no action.

Recommendation

Refine the smart contract's `RemoveVerifierServerEndPoints()` to recognize and appropriately differentiate handling for an empty verifier server list to maintain endpoint management integrity.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

PCP-01 REDUNDANCY IN ADDRESS CONVERSION LEADING TO INEFFICIENCIES

Category	Severity	Location	Status
Gas Optimization	Optimization	CAContract_Actions.cs (1.4.9): 107; CAContract_LoginGuardianAccount.cs (1.4.9): 106	Resolved

Description

The smart contract function `ConvertVirtualAddressToContractAddress()` exhibits inefficiency by redundantly converting a virtual address to a contract address more than once. This repetition occurs despite the correct storage of the result in `caAddress`, resulting in unnecessary gas costs.

Recommendation

Optimize the code by removing the second conversion call on line 107 and using the pre-computed `caAddress`.

Alleviation

The client revised the code and resolved this issue in commit : edc220af5d468817976e6ebbbd5ade1f6fb2ed9d

APPENDIX | PORTKEY FINANCE - AUDIT

Finding Categories

Categories	Description
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Access Control	Access Control findings are about security vulnerabilities that make protected assets unsafe.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK 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. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Elevating Your Entire Web3 Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.