CERTIK

Portkey - zklogin audit

Security Assessment

CertiK Assessed on Dec 7th, 2024

CertiK Assessed on Dec 7th, 2024

Portkey - zklogin audit

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

Executive Summary

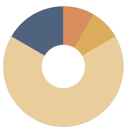
TYPES	ECOSYSTEM	METHODS
Wallet	Aelf	Manual Review, Static Analysis

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

CODEBASE
https://github.com/Portkey-Wallet/portkey-contracts/
View All in Codebase Page

COMMITS
975a55861226fcc2b84b4d9fb5ee7ffb445ca885
c92913d350423dce1b28eb50fc64c1bb4658de21
View All in Codebase Page

Vulnerability Summary



12	10	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.
🟧	1	Major	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.
🟨	1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	8	Minor	7 Resolved, 1 Acknowledged	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.
🟦	2	Informational	2 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 - ZKLOGIN AUDIT

CODEBASE | PORTKEY - ZKLOGIN AUDIT

Repository

https://github.com/Portkey-Wallet/portkey-contracts/

Commit

975a55861226fcc2b84b4d9fb5ee7ffb445ca885

c92913d350423dce1b28eb50fc64c1bb4658de21

AUDIT SCOPE | PORTKEY - ZKLOGIN AUDIT

10 files audited ● 1 file with Acknowledged findings ● 1 file with Mitigated findings ● 3 files with Resolved findings ● 5 files without findings

ID	Repo	File	SHA256 Checksum
● CAA	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/CAContract_Actions.cs	8ef0c0cb2e5158474856b985d9a43678c6978bd7e393f425c748a3e8400387bf
● CAM	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/CAContract_Managers.cs	ab68b75a8ffa6b974c83b1cb7c252941efd439820be47b2bb630f45a8f30f60d
● BEP	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/BytesExtension.cs	87985805d4b7e60b1e1b016a1fecd832d388209125b37aa88f60c16a245bfcd0
● CAG	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/CAContract_Guardians.cs	32d816a8d11751edbc10c0175494011d6006df98f4c03a1299624b4851f0a144
● CAZ	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/CAContract_ZkLogin.cs	b16049b930b694a0bdbce5a579ff8274f8fa20e765350aae3befbdb7d5287592
● CAC	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/CAContractState.cs	4ca1b879309aafa0f0e52e0daf8c7089878a361d69bb97844217b986d9bdaeb5
● CAR	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/CAContractReferenceState.cs	d305fba0b3b3c9833ecbb5d9a8dcb3db26dd82227b60c11786dc8ae545c57edc
● CAL	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/CAContract_LoginGuardianAccount.cs	b06d5673a2154c19a95b301c47051daf93412871a7ca4e3548ad0e98a0efeede
● HHP	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/HexHelper.cs	a7364edffa617b201b76f91210ddd3a1da005631807ecf22b5dad64cb1da5436

ID	Repo	File	SHA256 Checksum
● CAS	Portkey-Wallet/portkey-contracts	Portkey.Contracts.CA/CAContract_ACS2_StatePathsProvider.cs	f7c98ee6ce000eaa89f2cfd93cdec815b9ef8b766720a1855d3c9441d2a7c56a

APPROACH & METHODS | PORTKEY - ZKLOGIN AUDIT

This report has been prepared for Portkey to discover issues and vulnerabilities in the source code of the Portkey - zklogin 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.

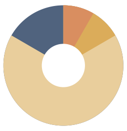
The auditing process pays special attention to the following considerations:

- 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 - ZKLOGIN AUDIT



12	0	1	1	8	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Portkey - zklogin audit. Through this audit, we have uncovered 12 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
PCP-02	**Centralization Related Risks**	**Centralization**	**Major**	● **Mitigated**
CAM-01	Users Can Bypass The Daily Transfer Amount Check Through The `ManagerTransferFrom()` Function	Volatile Code	Medium	● Resolved
BEP-01	Potential Out-Of-Bounds Error	Volatile Code	Minor	● Resolved
CAA-01	Missing Check For `Kid` And `N` In The Response	Volatile Code	Minor	● Resolved
CAA-03	Third-Party Dependencies	Design Issue	Minor	● Acknowledged
CAG-01	Optimization Of The `AppendGuardianPoseidonHash` Function	Volatile Code	Minor	● Resolved
CAG-02	The `AddGuardian()` Function May Potentially Revert Due To The Missing Input Parameter `operateDetails` In The `CheckVerifierSignatureAndData()` Function	Volatile Code	Minor	● Resolved
CAG-03	`ManuallySupportForZk` Is Not Set To `true`	Logical Issue	Minor	● Resolved
PCP-01	The Design Of ZKLogin	Design Issue	Minor	● Resolved

ID	Title	Category	Severity	Status
PCP-03	Inconsistent Generation Rule Of The Verifier ID When The Guardian Supports Zero-Knowledge (ZK)	Design Issue	Minor	● Resolved
CAA-02	Verification Information Does Not Need To Be Checked When The `CanZkLoginExecute()` Function Returns `true`	Volatile Code	Informational	● Resolved
CAA-04	Incorrect Error Message	Coding Style	Informational	● Resolved

PCP-02 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	● **Major**	**Portkey.Contracts.CA/CAContract_Actions.cs: 443, 477, 515, 533, 542; Portkey.Contracts.CA/CAContract_Managers.cs: 310, 322**	● **Mitigated**

Description

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

- AddManagerApproveSpenderWhitelist()
- RemoveManagerApproveSpenderWhitelist()
- AppendGuardianPoseidonHash()
- AppendGoogleGuardianPoseidon()
- AppendAppleGuardianPoseidon()
- AddOrUpdateJwtIssuer()
- AddKidPublicKey()
- AddOrUpdateVerifyingKey()
- SetOracleAddress()
- StartOracleDataFeedsTask()

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

- add the spender to the manager approval whitelist
- remove the spender from the manager approval whitelist
- append guardian PoseidonHash
- append guardian PoseidonHash of `Google` type
- append guardian PoseidonHash of `Apple` type
- add or update Json-Web-Token(JWT) issuer
- add apple public key
- add or update verifying key
- set oracle contract address
- start oracle data feeds task

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, 11/29/2024]:

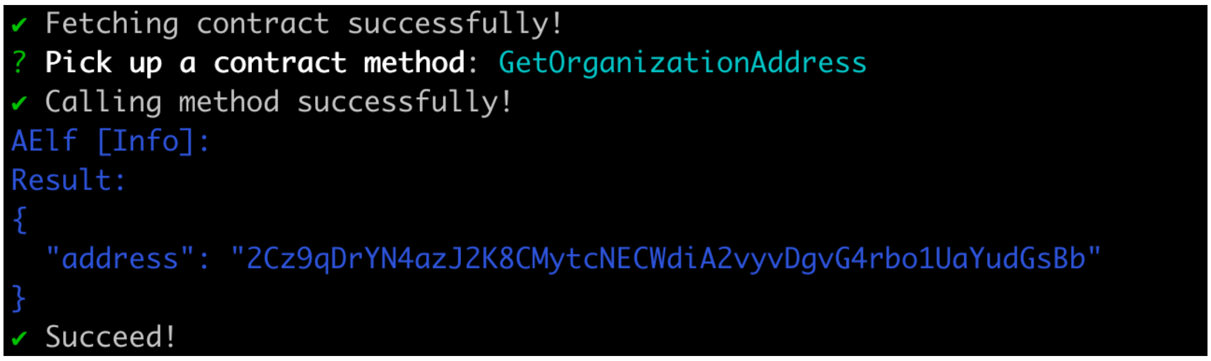
1. We have removed AppendGuardianPoseidonHash,AppendGoogleGuardianPoseidon and AppendAppleGuardianPoseidon contract methods.

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/c92913d350423dce1b28eb50fc64c1bb4658de21



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

[CertiK, 11/29/2024]: 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.

CAM-01 | USERS CAN BYPASS THE DAILY TRANSFER AMOUNT CHECK THROUGH THE `ManagerTransferFrom()` FUNCTION

Category	Severity	Location	Status
Volatile Code	● Medium	Portkey.Contracts.CA/CAContract_Managers.cs: 255	● Resolved

Description

In the `ManagerTransfer()` and `ManagerForwardCall()` functions, the `UpdateDailyTransferredAmount()` function is invoked to verify if the transfer amount exceeds the single transfer limit and the daily transfer limit, as well as to ensure adequate transfer security level. However, the `ManagerTransferFrom()` function does not call this verification function. Consequently, users can bypass the transfer amount check, and their accounts may fall into a low-security level for transfers.

Recommendation

We recommend invoking the `UpdateDailyTransferredAmount()` function in the `ManagerTransferFrom()` function to ensure that users' accounts adhere to a high-security level for transfers.

Alleviation

The client revised the code and resolved this issue in commit : 3ae69a02a46f4e97bb92df68e668a8a99d8c61c8

BEP-01 | POTENTIAL OUT-OF-BOUNDS ERROR

Category	Severity	Location	Status
Volatile Code	● Minor	Portkey.Contracts.CA/BytesExtension.cs: 7, 37, 53	● Resolved

Description

The `ShiftArrayRight` method shifts the bits of the `array` to the right by a specified number of bits (`shiftBits`). If shiftBits is larger than the total number of bits in the array (i.e., `shiftBits > array.Length * 8`), the method does not handle this scenario explicitly. This could lead to unintended behavior such as filling the entire array with zeros or potentially accessing out-of-bounds indices in other contexts.

This `ShiftBytesRight` method shifts the bytes of the `array` to the right by a specified number of bytes (`shiftBytes`). If shiftBytes is greater than or equal to the length of the `array` , all elements in the array will be shifted out, effectively leaving the array filled with zeros. While this may be the intended behavior, it is not explicitly handled, and in other contexts, this could lead to unexpected results.

In the `Mask` method, `maskBits` is calculated bitwise, but the case where `maskBits` exceeds the actual bit length of the `array` is not addressed. If `maskBits` surpasses the total length of the array, unnecessary operations might occur.

Another potential issue is that the logic in the `Mask` method may not correctly handle cases where `maskBits` is less than 8.

Recommendation

We recommend implementing boundary checks or adding explicit conditions to handle these scenarios.

Alleviation

The client revised the code and resolved this issue in commit : 799d946db2f1570a3cfebdbb515d1852b36633a2

CAA-01 | MISSING CHECK FOR `Kid` AND `N` IN THE RESPONSE

Category	Severity	Location	Status
Volatile Code	● Minor	Portkey.Contracts.CA/CAContract_Actions.cs: 601, 609	● Resolved

Description

The `HandleOracleFulfillment()` function ensures that the response is not null, but it does not verify the `Kid` and `N` parameters before invoking the `SetPublicKeyAndChunks()` function.

Recommendation

We recommend adding a check to ensure `Kid` and `N` are not null.

Alleviation

The client revised the code and resolved this issue in commit : 5e344fb54f67ea8ff67c538ba61325ed495eff57

CAA-03 | THIRD-PARTY DEPENDENCIES

Category	Severity	Location	Status
Design Issue	● Minor	Portkey.Contracts.CA/CAContract_Actions.cs: 554, 559	● Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party AetherLink contracts, including Consumer contract, Coordinate contract and Oracle Contract. The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets.

Recommendation

We recommend that the project team constantly monitor the functionality of third party dependencies to mitigate any side effects that may occur when unexpected changes are introduced.

Alleviation

[Portkey Team, 09/03/2024]: The jwksEndpoint is passed by CA contract, and the AetherLink contracts is also aelf's project team, we will constantly monitor the functionality, and AetherLink team will also do the same.

CAG-01 | OPTIMIZATION OF THE `AppendGuardianPoseidonHash` FUNCTION

Category	Severity	Location	Status
Volatile Code	● Minor	Portkey.Contracts.CA/CAContract_Guardians.cs: 279, 324	● Resolved

Description

The controller can append a guardian Poseidon Hash through the `AppendGuardianPoseidonHash()` function. The `IsZkLoginSupported()` function indicates that only `Google` and `Apple` type are supported. Consequently, if the guardian type is neither `Google` nor `Apple`, the loop skips this guardian. The `SetPoseidonHash()` function removes all underscores from the `PoseidonIdentifierHash`, whereas the loop does not perform this action.

Furthermore, the `AppendGoogleGuardianPoseidon()` function handles empty strings (""), but neither `AppendAppleGuardianPoseidon()` nor `AppendGuardianPoseidonHash()` include this handling.

```
private static bool IsZkLoginSupported(GuardianType type)
    {
        return GuardianType.OfGoogle.Equals(type) ||
GuardianType.OfApple.Equals(type);
    }
```

```
input.PoseidonIdentifierHash.Contains("");
```

Recommendation

We recommend adding some checks in the `AppendGuardianPoseidonHash` and `AppendAppleGuardianPoseidon` functions to maintain the consistency.

Alleviation

The client revised the code and resolved this issue in commit : 1b0422dfcc0b4ec53737db4e2936038da92bb920

CAG-02 THE `AddGuardian()` FUNCTION MAY POTENTIALLY REVERT DUE TO THE MISSING INPUT PARAMETER `operateDetails` IN THE `CheckVerifierSignatureAndData()` FUNCTION

Category	Severity	Location	Status
Volatile Code	● Minor	Portkey.Contracts.CA/CAContract_Guardians.cs: 37	● Resolved

Description

In the `AddGuardian()` function, if the guardian supports ZK, the `CheckZkLoginVerifierAndData()` function is invoked to ensure the added guardian is valid. Otherwise, the `CheckVerifierSignatureAndData()` function is invoked. Within the `CheckVerifierSignatureAndData()` function, there is a `CheckVerifierSignatureOperationDetail()` function that verifies the Hash value of operation details. The function will return false if the operation details are null or whitespace when the `CheckOperationDetailsInSignatureEnabled` flag is `true` or if the length of `verifierDoc` is greater than 7 . On line 37 of the `AddGuardian()` function, there is no operation details parameter provided, causing the function to revert.

```
private bool CheckVerifierSignatureOperationDetail(string operationTypeName,
string[] verifierDoc,
        string operationDetails)
    {
        if (State.CheckOperationDetailsInSignatureEnabled.Value
            || verifierDoc.Length >= 8)
        {
            if (verifierDoc.Length < 8 || string.IsNullOrWhiteSpace(verifierDoc[7])
||
                string.IsNullOrWhiteSpace(operationDetails))
            {
                return false;
            }

            return verifierDoc[7] ==
HashHelper.ComputeFrom(operationDetails).ToHex();
        }

        return true;
    }
```

Recommendation

We recommend passing the `operateDetails` parameter to prevent the `AddGuardian()` function from reverting.

Alleviation

The client revised the code and resolved this issue in commit : 2f08d978463bac4e13a82893403d21efe4d82535

CAG-03 | `ManuallySupportForZk` IS NOT SET TO `true`

Category	Severity	Location	Status
Logical Issue	● Minor	Portkey.Contracts.CA/CAContract_Guardians.cs: 56	● Resolved

Description

In the `CreateCAHolder()` function, when a new user registers, the `ManuallySupportForZk` variable is set to `true`. However, the `AddGuardian()` function does not set the `ManuallySupportForZk` to `true` even if the guardian supports Zero-Knowledge (ZK).

Recommendation

We recommend setting `ManuallySupportForZk` to `true` when the guardian supports Zero-Knowledge (ZK) in the `AddGuardian()` function.

Alleviation

The client revised the code and resolved this issue in commit : f91fe5d0c699d04708547e2a3a1622b5be521039

PCP-01 | THE DESIGN OF ZKLOGIN

Category	Severity	Location	Status
Design Issue	● Minor	Portkey.Contracts.CA/CAContract_Actions.cs: 248; Portkey.Contracts.CA/CAContract_ZkLogin.cs: 58~59	● Resolved

Description

Based on the comments and the implementation of the actual code, we have identified the following issue:

1. For new users of the latest version, the portkey contract should generate a random verifier for the guardian supporting ZK. However, in the `CreateCaHolderInfoWithCaHashAndCreateChainIdForZkLogin()` function, the `GetOneVerifierFromServers()` function always returns the first verifier instead of a random one when the `VerificationInfo` does not exist.

```
private Hash GetOneVerifierFromServers()
    {
        var verifiers = State.VerifiersServerList.Value.VerifierServers;
        return verifiers[0].Id;
    }
```

2. The `DoCheckZkLoginBasicParams()` function ensures that the nonce cannot be used more than once. The existing nonce list is retrieved from the return value of the `InitZkNonceInfos()` function. However, this function clears nonces that are overdue (i.e., older than one day). This means the nonce can still be used multiple times if it is older than one day, which could lead to a replay attack.

```
 private List<string> InitZkNonceInfos(Hash caHash, Timestamp currentTime)
 {
     State.ZkNonceInfosByCaHash[caHash] ??= new ZkNonceList();
     //clear overdue nonces, prevent the nonce data growing all the time
     foreach (var zkNonceInfo in
State.ZkNonceInfosByCaHash[caHash].ZkNonceInfos)
     {
         var nonceDatetime =
DateTime.SpecifyKind(Convert.ToDateTime(zkNonceInfo.Datetime),
DateTimeKind.Utc);
         if (nonceDatetime.ToTimestamp().AddDays(1) <= currentTime)
         {

State.ZkNonceInfosByCaHash[caHash].ZkNonceInfos.Remove(zkNonceInfo);
         }
     }

     return State.ZkNonceInfosByCaHash[caHash].ZkNonceInfos.Select(nonce =>
nonce.Nonce).ToList();
 }
```

3. In the `UpdateGuardian()` function, there is a `UpdateSupportZk` variable for `GuardianToUpdatePre` and `GuardianToUpdateNew`. How can the input values be controlled to ensure they are as expected? There is no validation for these variables, and if they are input incorrectly, the verifier cannot be validated correctly.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

Issue 1: The client revised the code and resolved this issue in commit : 55db76a0fa8799c6ac154ee9ff007b6111da9603

Issue 2: The client revised the code and resolved this issue in commit : 55db76a0fa8799c6ac154ee9ff007b6111da9603

Issue 3: The client revised the code and resolved this issue in commit : c2ac5c87ed81953c7efa41b67f8975b44da9ec28

PCP-03 INCONSISTENT GENERATION RULE OF THE VERIFIER ID WHEN THE GUARDIAN SUPPORTS ZERO-KNOWLEDGE (ZK)

Category	Severity	Location	Status
Design Issue	● Minor	Portkey.Contracts.CA/CAContract_Actions.cs: 246~248; Portkey.Contracts.CA/CAContract_Guardians.cs: 61	● Resolved

Description

In the `CreateCaHolderInfoWithCaHashAndCreateChainIdForZkLogin()` function, if the `VerificationInfo` is null or if `guardianApproved.VerificationInfo.Id` is an empty hash, `VerifierId` returns the first verifier from the server list. Otherwise, it returns `guardianApproved.VerificationInfo.Id`.

```
246
247   VerifierId = guardianApproved.VerificationInfo == null
248                          || Hash.Empty.Equals(guardianApproved.VerificationInfo.Id)
249                    ? GetOneVerifierFromServers() : guardianApproved.VerificationInfo.Id,
250
```

However, in the `AddGuardian()` function, `VerifierId` is directly assigned the value of `input.GuardianToAdd.VerificationInfo.Id` without validating `VerificationInfo`.

```
61
62  VerifierId = input.GuardianToAdd.VerificationInfo.Id,
63
```

Recommendation

We recommend maintaining consistent logic for the `VerifierId`.

Alleviation

The client revised the code and resolved this issue in commit : c728f466bcab1e0047ab53f4e74c91912217667f

CAA-02 | VERIFICATION INFORMATION DOES NOT NEED TO BE CHECKED WHEN THE `CanZkLoginExecute()` FUNCTION RETURNS `true`

Category	Severity	Location	Status
Volatile Code	● Informational	Portkey.Contracts.CA/CAContract_Actions.cs: 47, 169~172	● Resolved

Description

If the guardian supports Zero-Knowledge (ZK), this contract verifies using ZK. Otherwise, it uses the original verifier. As a result, if the guardian supports ZK, the verification information does not need to be checked. Therefore, the aforementioned lines should be checked only if the `CanZkLoginExecute()` function returns `false` .

Recommendation

We recommend adding a limit before checking the verification information to prevent unrelated checks from causing a revert.

Alleviation

[Portkey Team, 09/03/2024]: In order to be compatible with older versions of users, even if a guardian supports zk, it is necessary to pass the verifier Id of the verifier.

CAA-04 | INCORRECT ERROR MESSAGE

Category	Severity	Location	Status
Coding Style	● Informational	Portkey.Contracts.CA/CAContract_Actions.cs: 480, 649	● Resolved

Description

The error message in the following lines do not accurately describe the error condition:

```
480  Assert(input != null, "Invalid input when AddJwtIssuer.");
```

```
649   Assert(Context.Sender == State.Admin.Value, "No SetCircomBigInt permission.");
```

Recommendation

We recommend changing the error message to align with the conditional check.

Alleviation

The client revised the code and resolved this issue in commit : 4d3ca730d91dfc6f5d005c25eebd9f55ee88a9ae

APPENDIX | PORTKEY - ZKLOGIN AUDIT

Finding Categories

Categories	Description
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.
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.