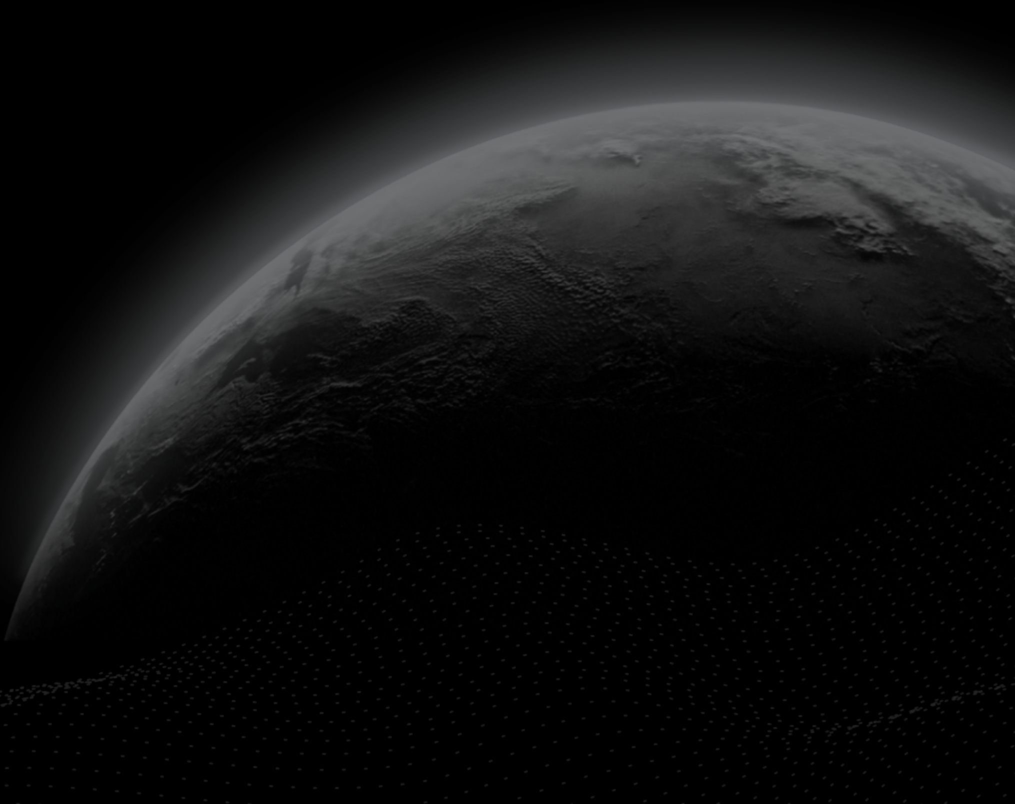
Security Assessment

BackedBy - Audit

CertiK Verified on Jan 4th, 2023



CertiK Verified on Jan 4th, 2023

BackedBy - Audit

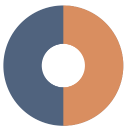
The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 01/04/2023	Subscriptions

CODEBASE	COMMITS
https://github.com/backedby/v1-contracts ...View All	b54513fd86ac4770c8b7f707c5bed7cdf72ce9d1 ...View All

Vulnerability Summary



4	3	0	0	1	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥	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 Acknowledged	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.
🟨	0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	0	Minor		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 | BACKEDBY - AUDIT

CODEBASE | BACKEDBY - AUDIT

Repository

https://github.com/backedby/v1-contracts

Commit

b54513fd86ac4770c8b7f707c5bed7cdf72ce9d1

AUDIT SCOPE | BACKEDBY - AUDIT

13 files audited ● 5 files with Acknowledged findings ● 8 files without findings

ID	File	SHA256 Checksum
● BBP	contracts/BBPosts.sol	8a11671818673616065fedd0586e07ca2a17db668499e0cddd0725a4ecba177a
● BBS	contracts/BBProfiles.sol	4e200f3a581df903982424063dc1f367202cb8b3cbdea8cfde236dd421160a85
● BSP	contracts/BBSubscriptions.sol	b9c7cd1968db80c69a2cfff58bddabe4807a25c0cd2fbc82ab4be531f7407336
● BBF	contracts/BBSubscriptionsFactory.sol	caf504cc855acabcd3859dd26259b21f22a3d4dfa7f825415d2a278649126f5c
● BBT	contracts/BBTiers.sol	d80d51c2bd3b8c7c89c7a7d2ee67c9ba728b56b62e84ef5a01945fd979a2f6fd
● BBE	contracts/BBErrorsV01.sol	316b9a1bc6a0b24ef2b7d6fbe35ad47db4565ea2f20a12025002e99f7c0f03e9
● DTL	contracts/DateTimeLibrary.sol	8a7ee447db7d47541b2da27ab8e8eaf245682950af1583401628eaf8dfdde36f
● IBB	contracts/interfaces/IBBPermissionsV01.sol	ea3cb558b261b3728a95275bffb498512e42e1dfa05e2060385d3fd512469c56
● IBP	contracts/interfaces/IBBPosts.sol	d9cb5cef2d028ad6e3855f6071c16ab4541ee99c2a7fa4731bc783bd065a823c
● IBS	contracts/interfaces/IBBProfiles.sol	23bfebd1d7fd153dfbe1ca8b3c65fa3138b422b477f74c0833b0849a3dc511bd
● IBE	contracts/interfaces/IBBSubscriptions.sol	5e991978c9d26df6244162f7eb8b80334fe972e87941dc1dcb9091e2107ed27e
● IBF	contracts/interfaces/IBBSubscriptionsFactory.sol	6b2fc0f3c4d56476483fe499484b1474165013fa2e0d542f1d5f05c942c821fd
● IBT	contracts/interfaces/IBBTiers.sol	80ad7136152a6abec5ecaa79da6e5f5ccb7401a8ebaf64e61ef4dc0e6517c523

APPROACH & METHODS | BACKEDBY - AUDIT

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

REVIEW NOTES | BACKEDBY - AUDIT

Overview

BackedBy is a decentralized payment platform for content creators. Creators can post encrypted content to smart contracts, and users decrypt and view content by subscribing and making monthly token payments to content creators.

External Dependencies

In BackedBy, the project relies on a few external contracts to fulfill the needs of its business logic.

Contracts

The project uses OpenZeppelin contracts and Chainlink interfaces for contract format and functionality.

The following contracts are referenced in various contracts:

OpenZeppelin

- `IERC20.sol` , `Ownable.sol`

ChainLink

- `KeeperCompatibleInterface.sol`

Oracle

- `IBBGasOracle`

Addresses

The following addresses interact at some point with specified contracts, making them an external dependency. During the review, no hardcoded address values were found in the codebase. All following values are initialized either at deploy time or by specific functions in smart contracts.

BBSubscriptions.sol

- `_currency`

BBSubscriptionsFactory.sol

- `_treasury` , `refundReceiver` , `IBBPermissionsV01(profileOwner)`

It is assumed that these contracts or addresses are valid and non-vulnerable actors and implement proper logic to collaborate with the current project.

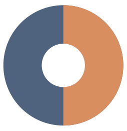
Privileged Roles

To set up the project correctly, improve overall project quality and preserve upgradability, the following roles are adopted in the codebase:

- `_owner` has the authority to set important contract parameters and contract addresses.
- `profileOwner` has the authority to set up and edit subscription profiles and relevant fee structures.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Furthermore, any plan to invoke the aforementioned functions should also be considered to move to the execution queue of the `Timelock` contract.

FINDINGS | BACKEDBY - AUDIT



4	0	2	0	0	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for BackedBy - Audit. Through this audit, we have uncovered 4 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
BBS-01	Incorrect Handling Of _ownedProfiles Mapping Update In `editProfile()`	Logical Issue	Major	● Resolved
CON-01	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	● **Acknowledged**
BBF-01	Lack Of Checks For Types Of Token Contracts Accepted	Logical Issue	Informational	● Resolved
BSP-01	Logic Can Be Consolidated For `checkUpkeep()`	Coding Style	Informational	● Resolved

BBS-01 INCORRECT HANDLING OF _OWNEDPROFILES MAPPING UPDATE IN `editProfile()`

Category	Severity	Location	Status
Logical Issue	● Major	contracts/BBProfiles.sol: 97	● Resolved

Description

There is an issue with the current logic for updating the `_ownedProfiles` mapping within the `editProfile()` code for the original owner when ownership transfers occur.

The current logic sets the final index of the `_ownedProfiles` mapping of the original owner to the `profileId` of the profile that is being set to change owners:

```
97  _ownedProfiles[msg.sender][_ownersTotalProfiles[msg.sender] - 1] =
_ownedProfiles[msg.sender][_ownedProfilesIndexes[msg.sender][profileId]];
```

This line of code overwrites the last profile index of the caller's `_ownedProfiles` mapping to the profile that is being transferred to the new owner. This means that the profile that is supposed to be transferred remains in the caller's mapping and even overwrites the caller's last active profile.

This would also have an impact when `getOwnersProfiles()` is run since the `profileId` that is being transferred is still being shown under the previous owner in the `_ownedProfiles` mapping.

Recommendation

It is advised that the team review the logic for this replacement to ensure that the logic works as expected without overwriting any valid records.

As per the project design, it seems that the intended implementation is to replace the mapping related to the profile to be transferred with the last profile within the `_ownedProfiles` mapping and to remove the last index to prevent having any empty `profileId`. This method will effectively remove the profile from the `ownedProfiles` mapping for the caller while also adjusting the `profileIds` accordingly. For example,

```
97  _ownedProfiles[msg.sender][_ownedProfilesIndexes[msg.sender][profileId]] =
_ownedProfiles[msg.sender][_ownersTotalProfiles[msg.sender] - 1];
98  _ownedProfiles[msg.sender][_ownersTotalProfiles[msg.sender] - 1] = 0;
```

Alleviation

[BackedBy, 10/10/2022]: The team heeded the advice and resolved the finding in the commit hash <6c4efc9e7ec0e7ac43a663211d62189fcbb6e2f1>. The team made the suggested changes to properly replace the transferred profile with the last profile within the `_ownedProfiles` mapping and subsequently cleared the final item in the `_ownedProfiles` mapping to ensure that there are no empty slots within the mapping as a result of transferred profiles.

CON-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/BBPosts.sol: 57, 75; contracts/BBProfiles.sol: 94; contracts/BBSubscriptions.sol: 278; contracts/BBSubscriptionsFactory.sol: 135, 202, 216, 257; contracts/BBTiers.sol: 96, 116, 150**	● **Acknowledged**

Description

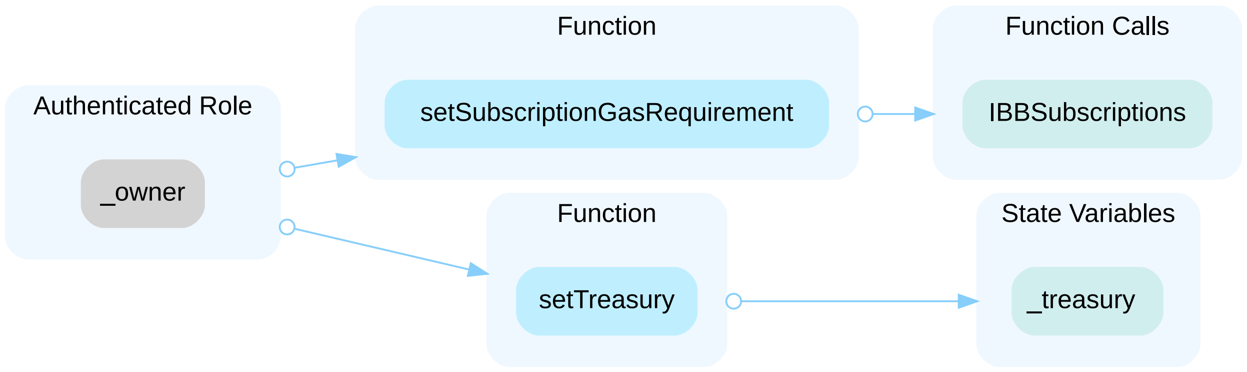
There are several privileged roles within the contracts that control access to significant administrative functions.

Owner

In the contract `BBSubscriptionsFactory` the role `_owner` has authority over the functions shown in the diagram and the list below:

- setTreasury() - Set treasury address
- setSubscriptionGasRequirement() - Set the subscription gas requirement for a specific currency

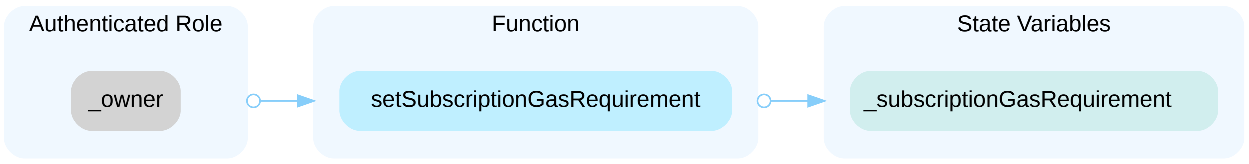
Any compromise to the `_owner` account may allow the hacker to take advantage of this authority to set important contract parameters.



In the contract `BBSubscriptions` the role `_owner` has authority over the functions shown in the diagram and the list below:

- setSubscriptionGasRequirement() - Set the subscription gas requirement

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority to set important contract parameters.

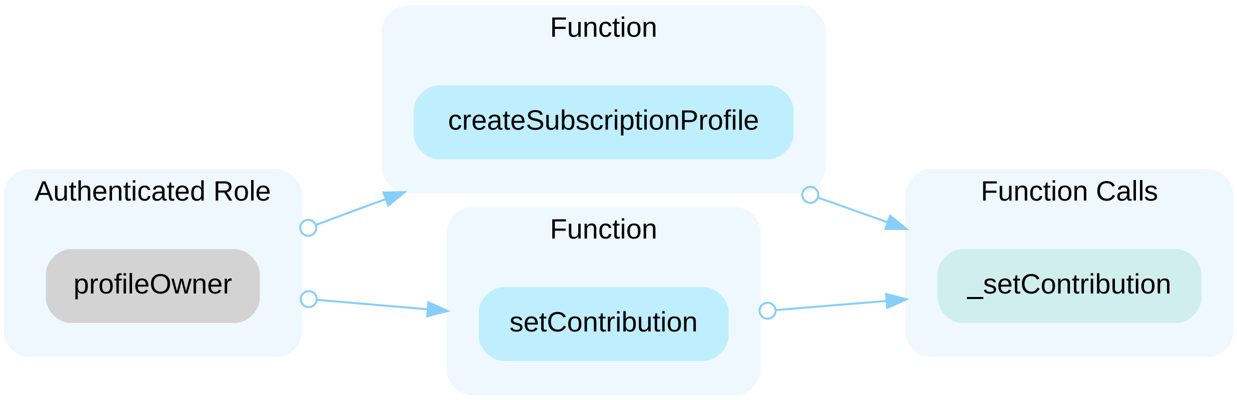


Profile Owner

In the contract `BBSubscriptionsFactory` the role `profileOwner` has authority over the functions shown in the diagram and the list below:

- createSubscriptionProfile() - Set up a new subscription profile
- setContribution() - Set a subscription profile's treasury contribution percentage

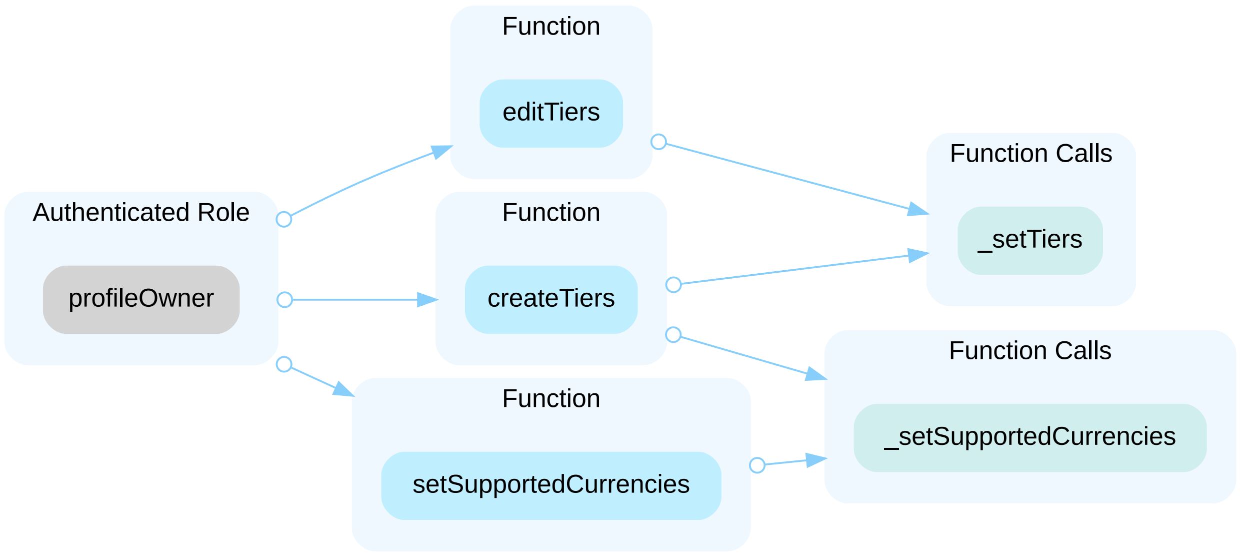
Any compromise to the `profileOwner` account may allow the hacker to take advantage of this authority to set important contract parameters.



In the contract `BBTiers` the role `profileOwner` has authority over the functions shown in the diagram and the list below:

- createTiers() - Create new tier level
- editTiers() - Edit existing tier levels
- setSupportedCurrencies() - Set supported ERC20 tokens for payments

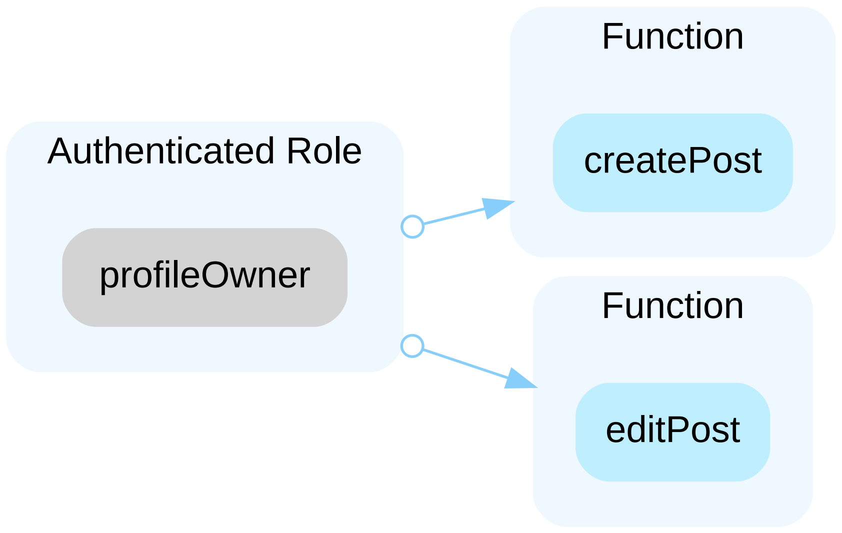
Any compromise to the `profileOwner` account may allow the hacker to take advantage of this authority to set important contract parameters.



In the contract `BBPosts` the role `profileOwner` has authority over the functions shown in the diagram and the list below:

- createPost() - Create a new post
- editPost() - Edit existing post variables

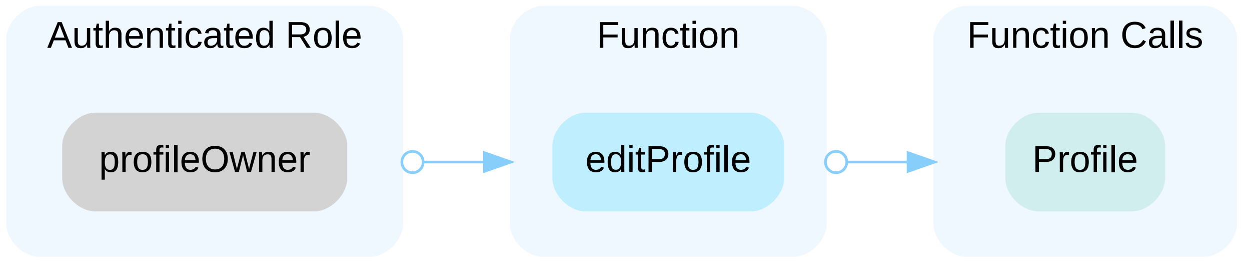
Any compromise to the `profileOwner` account may allow the hacker to take advantage of this authority to create and edit posts.



In the contract `BBProfiles` the role `profileOwner` has authority over the functions shown in the diagram and the list below:

- editProfile() - Edit existing profile variables

Any compromise to the `profileOwner` account may allow the hacker to take advantage of this authority to edit profiles.



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. The client is advised to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, it is strongly recommended that 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

[BackedBy, 12/20/2022]: The team updated the roles for the `BBSubscriptionsFactory` in commit hash <b54513fd86ac4770c8b7f707c5bed7cdf72ce9d1>.

In order to reduce the risk of a single address being compromised, the `_owner` role has been split into the following roles:

- `_treasuryOwner` - has the authority to run `setTreasuryOwner()` & `setTreasury()`
- `_gasOracleOwner` - has the authority to run `setGasOracleOwner()` & `setGasOracle()`
- `_subscriptionFeeOwner` - has the authority to run `setSubscriptionFeeOwner()` & `setSubscriptionFee()`

The `_owner` role was also removed from the `BBSubscriptions` contract since gas prices are extracted from gas price oracles instead.

It is important to note that these roles are initially set to the deployer address through the constructor() at deployment and requires the individual setting up of the addresses at a later point in time through the running of setter functions.

In terms of the `profileOwner` the BackedBy team explained that these roles were assumed and controlled by the users of the BB contract to allow them to responsible for their own security/decentralization.

[CertiK, 01/03/2023]: The splitting of privileged roles can avoid a single-point failure to some extent, however it is highly recommended to adopt a multi-sig wallet solution.

BBF-01 | LACK OF CHECKS FOR TYPES OF TOKEN CONTRACTS ACCEPTED

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/BBSubscriptionsFactory.sol: 96	● Resolved

Description

The design currently allows anyone to run the `deploySubscriptions()` function and set a subscription to allow any valid currency contract address. This might allow attackers to use malicious contracts included as a `currency` contract.

Recommendation

Recommend that a whitelist of accepted token contract addresses be included so that malicious token contracts which might negatively impact the platform cannot be included and used.

Alleviation

[BackedBy, 10/10/2022] The team decided to leave the code as is and provided the following explanation.

The architecture of the BackedBy subscriptions payments is designed to allow any ERC20 standard token to be supported.

The BackedBy team has reduced the risk of code injected into the `transfer()`, or `transferFrom()` function of an ERC20 token being used maliciously, by deploying a separate `BBSubscriptions` contract for each ERC20 token, which do not interact with each other.

Since users have to opt-in to each `BBSubscriptions` contract, if a user only uses `BBSubscriptions` contracts associated with well known ERC20 tokens, with standard `transfer()` and `transferFrom()` functions, there is no risk of any sort of exploit.

In production, the frontend will only support a handful of audited, and well-known ERC20 tokens for subscription payments. The team did not want developers building their own frontend to be forced into using only certain approved tokens.

[CertiK, 01/03/2023] The team confirmed that when deploying token contracts, the team will only support a handful of audited and well-known ERC20 tokens for subscription payments. The tokens serve as an external dependency outside the current audit scope.

BSP-01 | LOGIC CAN BE CONSOLIDATED FOR `checkUpkeep()`

Category	Severity	Location	Status
Coding Style	● Informational	contracts/BBSubscriptions.sol: 160~171, 178~190	● Resolved

Description

The two loops within function `checkUpKeep()` are essentially looping through the same values twice.

Recommendation

Recommend combining the two loops into one to save on gas and improve readability of code. The code should loop through all the values just once and return the appropriate values based on the results.

```
uint256 renewalIndex;
for(uint256 i; i < checkLength; i++) {
            uint256 subscriptionIndex = lowerBound + i;

            // If subscription has expired, add the subscription ID to the array of
IDs to renew
            if(_subscriptions[subscriptionIndex].expiration < block.timestamp &&
_subscriptions[subscriptionIndex].cancelled == false) {
                renewIndexes[renewalIndex] = subscriptionIndex;
                renewalIndex++;
            }
}

// If subscriptions to renew is zero or less than minimum required renewals, return
false
if(renewalIndex == 0 || renewalIndex < minRenews) {
    return (false, "");
}

return (true, abi.encode(renewIndexes, refundReceiver));
```

Alleviation

[BackedBy, 10/10/2022]: The team heeded the advice and resolved the finding in the commit hash <6c4efc9e7ec0e7ac43a663211d62189fcbb6e2f1>. The `checkUpkeep()` renewal check logic has been consolidated, and a loop to resize the renewal indexes array has been added.

OPTIMIZATIONS | BACKEDBY - AUDIT

ID	Title	Category	Severity	Status
BSP-02	Lacks Checks Of Input Parameters Decoded From Input Bytes Data	Logical Issue	Optimization	● Resolved
BSP-03	Change Logic To Skip Irrelevant `renewIndexes`	Coding Style	Optimization	● Resolved

BSP-02 LACKS CHECKS OF INPUT PARAMETERS DECODED FROM INPUT BYTES DATA

Category	Severity	Location	Status
Logical Issue	● Optimization	contracts/BBSubscriptions.sol: 91, 147	● Resolved

Description

For both functions `checkUpkeep()` and `performUpkeep()` there is no checking of the parameters decoded from the input data. Specifically for sensitive information like the `renewIndexes` and `refundReceiver` addresses, it would be useful to have a check to ensure that valid input values have been submitted into the function.

Recommendation

Recommend including checks with the functions to check that the input parameters are correct.

In the case of `refundReceiver` , a whitelist can be made to ensure that only registered addresses are able to receive funds and incoming input parameters can check with that list to ensure that legitimate addresses have been used for this value.

For `renewIndexes` , checks can be added within the logic to check that only valid entries are used, perhaps by adding require statements checking that `profileId` and `tierId` have been populated within the `_subscriptions[renewIndexes[i]]` entry as well.

Alleviation

[BackedBy, 10/10/2022]: The team heeded the advice and resolved the finding in the commit hash <3ea261df8a532b4b3d86bb53324ffac6a202879e>. The team included a check to ensure that the inputted `renewIndexes[i]` falls within the range of used subscription IDs.

In terms of the `refundReceiver` treatment, the team explained that the gas refund is designed to incentivize users to call the `performUpkeep()` function and keep subscription payments being made, therefore the `refundReceiver` must allow any address set by the function caller.

BSP-03 | CHANGE LOGIC TO SKIP IRRELEVANT `renewIndexes`

Category	Severity	Location	Status
Coding Style	● Optimization	contracts/BBSubscriptions.sol: 95~96	● Resolved

Description

The current logic for `performUpKeep()` might result in the function reverting after a series of updates and payments, which will be a waste of gas, and updates will need to be executed again from the beginning.

The issue is mainly related to the two require statements as shown below:

```
  95  require(_subscriptions[renewIndexes[i]].expiration < block.timestamp,
BBErrorCodesV01.SUBSCRIPTION_NOT_EXPIRED);
  96  require(_subscriptions[renewIndexes[i]].cancelled == false,
BBErrorCodesV01.SUBSCRIPTION_CANCELLED);
```

Recommendation

Recommend replacing the require statements with if statements and implementing a method to skip (and perhaps record) any skipped entries to prevent the function from reverting after a series of updates and payments have been made. This will improve the flow of the function and reduce the gas used when incorrect input data causes the function to fail.

An edge case that could cause the function to fail is when a user runs the `subscribe()` function for a valid expired subscription while or just before the `performUpKeep()` function is run. The addition of a skipping mechanism would ensure that all other entries are successfully updated.

Alleviation

[BackedBy, 10/10/2022]: The team heeded the advice and resolved the finding in the commit hash <6c4efc9e7ec0e7ac43a663211d62189fcbb6e2f1>. The team replaced `require` statements with `if` statements as recommended. The team also updated the gas refunded handling to a mechanism based on the number of indexes renewed or cancelled.

APPENDIX | BACKEDBY - AUDIT

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

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

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.

CertiK | **Securing** the **Web3** World

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.