Security Assessment

BLP

Sept 23rd, 2021

Table of Contents

Summary

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

- 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.

Overview

Project Summary

Project Name	BLP
Platform	BSC
Language	Solidity
Codebase	Private repository
Commit	

Audit Summary

Delivery Date	Sept 23, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	BullIdo, Bullpad, bullManager

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	2	0	0	1	1	0
Medium	3	0	0	1	0	2
Minor	1	0	0	1	0	0
Informational	1	0	0	0	0	1
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ACK	Address.sol	7a6f9acee77ad6aa4db616c473d507a5726698c3bd9005d6c54d8258c3cb6480
BCC	BullContract.sol	1bafb5e9e839ea21da88e4d6d492d11082a99a75efb0cdf9179db990a1e880d2
BPC	BullPad.sol	f2f85c6106e78b6065b37bd8994911de158f5cd696bf145e65916c702e426067
CCK	Context.sol	9a3d1e5be0f0ace13e2d9aa1d0a1c3a6574983983ad5de94fc412f878bf7fe89
DCK	Deployer.sol	1ed09381c6cd1edc5d8a03eefb062534da31167b76ca150ed80d6c8a6d031951
IBU	IBULL.sol	e3d1820047b408f708cdb4495a0fa57f076d94fed35da82cb7b51a9526ad6510
IER	IERC20.sol	0573c2961569aa4906845d0cd428b5b7394956170054ceeaa8f8af96cd44875c
ICK	Initializable.sol	10e65b6da82eb95f819bc3bf11c9b6d273bd5b04dab1e557ddac92d2aacb13cd
OCK	Ownable.sol	2b9fdaa1b13c3faab4d4edf2e58db7bae15aef70093e8095f5ff510e87a5f190
RGC	ReentrancyGuard.sol	3fc7968f4a1937caf3c96dffbac350398f86faad96288502e02c3a2b9f245e39
SER	SafeERC20.sol	80948ccae971a844a6c39336f3c55047dbc6e431f131861b7bf44a97c135e5fc
SMC	SafeMath.sol	38e47d1b5299ce0d5e48db837ed9248449043c50d90ffa0ee2ceb58ffde942c2

Overview

External Dependencies

There are a few depending injection contracts or addresses in the current project:

- `offeringToken` and `rasingToken` in the `BullIdo` contract
- `_offeringToken`, `_rasingToken` and `masterContract` in the `bullManager` contract

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

Privileged Functions

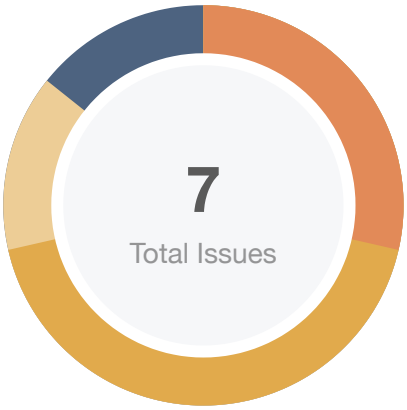
The contract contains the following privileged functions that are restricted by the `onlyOwner` modifier. They are used to modify the contract configurations and address attributes. We grouped these functions below.

The functions below have the `onlyOwner` modifier:

- `withdraw()`: The owner can transfer any amount from arbitrary `_token` or recipient.
- `setIdoBlock()`: The owner can update the ido or fcfs block time.
- `setReleaseBlock()`: The owner can update the release block and vesting block number.
- `setToken()`: The owner can change the token to an arbitrary address.
- `setFcfs()`: The owner can modify the remained allocation.
- `setAllocation()`: The owner can change any allocation, meaning that it might affect the price.
- `whitelistAddress()`: The owner allows to change arbitrary users as whitelist status to true.
- `removeAddress()`: The owner allows to change arbitrary users as whitelist status to false.
- `idoDeployment()`: The owner can set the `idoContract`.
- `newMasterContract()`: The owner can modify an arbitrary address as `masterContract` address.

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

Findings



Critical	**0** (0.00%)
Major	**2** (28.57%)
Medium	**3** (42.86%)
Minor	**1** (14.29%)
Informational	**1** (14.29%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
BCC-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
BCC-02	Incorrect Calculation	Mathematical Operations	Medium	Resolved
BCC-03	Discussion on Function `whiteListAllocation()`	Logical Issue	Medium	Acknowledged
BCC-04	Incompatibility With Deflationary Tokens	Volatile Code	Minor	Acknowledged
BPC-01	Missing Input Validation	Volatile Code	Informational	Resolved
DCK-01	Centralization Risk	**Centralization / Privilege**	**Major**	Partially Resolved
DCK-02	Lack of Access Control for Initialization	Volatile Code	Medium	Resolved

BCC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	BullContract.sol: 155, 161, 189, 194, 173, 183, 94, 103	Acknowledged

Description

In the contract `BullContract.sol`, the role `onlyOwner` has the authority over the following function:

- `withdraw()`: The owner can transfer any amount from arbitrary `_token` or recipient.
- `setIdoBlock()`: The owner can update the ido or fcfs block time.
- `setReleaseBlock()`: The owner can update the release block and vesting block number.
- `setToken()`: The owner can change the token to an arbitrary address.
- `setFcfs()`: The owner can modify the remained allocation.
- `setAllocation()`: The owner can change any allocation, meaning that it might affect the price.
- `whitelistAddress()`: The owner allows to change arbitrary users as whitelist status to true.
- `removeAddress()`: The owner allows to change arbitrary users as whitelist status to false.

Any compromise to the `onlyOwner` account may allow the hacker to take advantage of breaking the smart contract.

Recommendation

We advise the client to carefully manage the `onlyOwner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

BCC-02 | Incorrect Calculation

Category	Severity	Location	Status
Mathematical Operations	Medium	BullContract.sol: 118	Resolved

Description

The code implementation in L118 is designed to avoid users participating more than the share.

```
 115         User[_user]=user(userDetails.whitelist,
(userDetails.participationAmount).add(_amount),block.number,false,0);
 116         rasingToken.safeTransferFrom(_msgSender(),address(this),_amount);
 117         //avoid user participate more than share
 118
whiteListAllocation().sub(userDetails.participationAmount).sub(_amount,"BullPad:Participa
tion Amount Overflow");
```

However, the state `userDetails.participationAmount` has already updated and added `_amount` (which is the amount that the user participates in) in L115. Therefore, the calculation in L118 is incorrect since it minus `_amount` by twice.

Recommendation

We advise correct the calculation in L118:

```
 118
whiteListAllocation().sub(userDetails.participationAmount,"BullPad:Participation Amount
Overflow");
```

Alleviation

The development team heeded our advice and resolve this issue by updating `User[_user]` after the division.

```
 117
whiteListAllocation().sub(userDetails.participationAmount).sub(_amount,"BullPad:Participa
tion Amount Overflow");
 118         User[_user]=user(userDetails.whitelist,
(userDetails.participationAmount).add(_amount),block.number,false,0);
```

BCC-03 | Discussion on Function `whiteListAllocation()`

Category	Severity	Location	Status
Logical Issue	Medium	BullContract.sol: 206	Acknowledged

Description

According to the implementation of the function `whiteListAllocation()`, `whiteListAllocation()` would return the maximum tokens that the whitelisted user can participant.

```
206      function whiteListAllocation() public view returns(uint256){
207         return totalAllocation.div(totalWhitelist);
208      }
```

However, the return value of `whiteListAllocation()` might change dynamically since the owner of the contract can call `whitelistAddress()`/`removeAddress()` at any time, which would increase/decrease the value of `totalWhitelist`.

Recommendation

We advise the client to revisit the design and ensure it is intended.

BCC-04 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Volatile Code	Minor	BullContract.sol: 116, 132, 137	Acknowledged

Description

When transferring standard ERC20 deflationary tokens (i.e., `rasingToken` is a standard ERC20 deflationary token), the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user calls `_participate` to stake 100 deflationary tokens (with a 10% transaction fee) in `BullIdo` contract, only 90 tokens actually arrived in the contract. This might cause some tokenomic problems to the project.

Recommendation

We advise the client to regulate the set of `rasingToken` supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

BPC-01 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	BullPad.sol: 15, 18	Resolved

Description

The given input `_user` is missing the sanity check to validate the existence of the user.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error.

Alleviation

The development team heeded our advice and resolved this issue.

DCK-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Deployer.sol: 55, 23	Partially Resolved

Description

In the contract `Deployer.sol`, the role `onlyOwner` has the authority over the following function:

- `idoDeployment()`: The owner can set the `idoContract`.
- `newMasterContract()`: The owner can modify an arbitrary address as `masterContract` address.

Any compromise to the `onlyOwner` account may allow the hacker to take advantage of breaking the smart contract.

Recommendation

We advise the client to carefully manage the `onlyOwner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The development team heeded our advice and set the visibility of the function `idoDeployment()` as `internal.

(**CertiK**)

We advise the client to solve centralization risk by applying a decentralized mechanism or smart-contract-based accounts with enhanced security practices and provide corresponding transactions/proof to the community.

DCK-02 | Lack of Access Control for Initialization

Category	Severity	Location	Status
Volatile Code	Medium	Deployer.sol: 26~38	Resolved

Description

In the `bullManager` contract, the function `initialize()` can be called by anyone to initialize the contract. Although the project deployer can discard incorrectly initialized contracts, it might still bring errors if the deployment is not properly processed. One of the possible scenarios is described as below:

1. The deployer writes a script to deploy and initialize the contract.
2. The attacker noticed the deployment and initialized the contract before the initialization by the deployer is committed.
3. The deployment script mistakenly ignores the error of initializing the contract (because `initialized` is `true` now, the transaction of calling `init()` will be reverted), and continues executing other transactions in the script.

In this way, the attacker can inject suspicious addresses into the contracts.

Recommendation

We recommend adding proper access control to the `initialize()` function in the aforementioned contracts or checking the status of initialization in the deployment process.

Alleviation

The development team heeded our advice and resolve this issue.

Appendix

Finding Categories

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

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

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.

About

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.