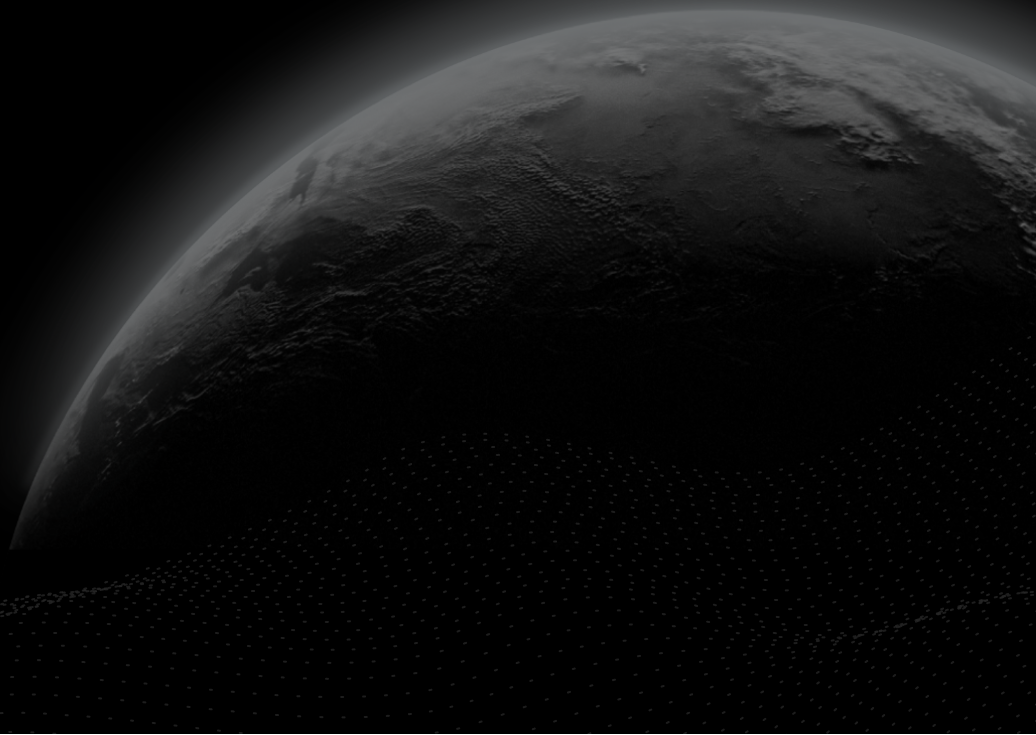




Security Assessment

Flash 3.0

CertiK Assessed on Jul 12th, 2023





Certik Assessed on Jul 12th, 2023

Flash 3.0

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

Executive Summary

TYPES

Others

ECOSYSTEM

Other

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 07/12/2023

KEY COMPONENTS

Token

CODEBASE

- <https://testnet.bscscan.com/token/0xa1B5819263FCeeb68b09B053b6A4471E596b594A>
- <https://etherscan.io/address/0xbb19da2482308ec02a242aced>

[View All in Codebase Page](#)

Vulnerability Summary



8

Total Findings

4

Resolved

1

Mitigated

1

Partially Resolved

2

Acknowledged

0

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 Mitigated, 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.



6

Minor

4 Resolved, 1 Partially 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.



0

Informational

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 | FLASH 3.0

I **Summary**

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

I **Review Notes**

[Overview](#)

[External Dependencies](#)

[Decentralization Efforts - Initial Token Distribution](#)

[Decentralization Efforts - Privileged Roles](#)

[FLASH 3.0](#)

[Recommendations](#)

[Short Term:](#)

[Long Term:](#)

[Permanent:](#)

I **Findings**

[FLA-06 : Initial Token Distribution](#)

[FLA-16 : Centralization Risks in FLASH.sol](#)

[FLA-17 : Missing Zero Address Validation](#)

[FLA-21 : Potential Denial-of-Service Situation](#)

[FLA-22 : Usage of `transfer`/`send` for Sending Native Tokens](#)

[FLA-23 : Incompatible Type Cast](#)

[FLA-24 : Unsafe Integer Cast](#)

[FLA-26 : Redundant Statements](#)

I **Appendix**

I **Disclaimer**



CODEBASE | FLASH 3.0

Repository

- <https://testnet.bscscan.com/token/0xa1B5819263FCeeb68b09B053b6A4471E596b594A>
- <https://etherscan.io/address/0xbb19da2482308ec02a242aced4fe0f09d06b12a7>

AUDIT SCOPE | FLASH 3.0

2 files audited ● 2 files without findings

ID	Repo	File	SHA256 Checksum
● FLS	CertiKProject/certik-audit-projects	 FLASH.sol	ecfc9cea2b9681841a19d31ad80f8976055d27272802d8dc4fa8d2a2ee830373
● FLA	CertiKProject/certik-audit-projects	 FLASH.sol	148a4676bf55d19e13e7f28821bbb0d24017265a79b6364127b54a6ac4ecbd01

APPROACH & METHODS | FLASH 3.0

This report has been prepared for Flash to discover issues and vulnerabilities in the source code of the Flash 3.0 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 | FLASH 3.0

Overview

The project `Flash 3.0`, is an BEP20 compatible burnable token running on BSC with an initial total supply set to 100,000,000 FLASH 3.0. The token contract also included functionality to control trading until the initial liquidity and other user protection measures.

External Dependencies

The following are external dependencies found in the contract codebase:

- A DEX router in `dexRouter` state variable.
- A Antisniper or protection mechanism in `initializer` state variable.

Supported DEX routers will depend on the deployment chain.

It is assumed that these contracts and libraries are valid and are implemented properly within the current project.

Decentralization Efforts - Initial Token Distribution

All FLASH 3.0 are sent to the deployer when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Decentralization Efforts - Privileged Roles

In the `FLASH 3.0` project, to limit and access control the usage of the FLASH 3.0 contract, it contains one modifier defined:

- `onlyOwner()` : makes sure that only an authorized admin or manager can execute the requested operation.

Apart from the previously defined modifiers, the contract supports a comprehensive list of functions that allow ownership management, tax management, contract settings modifications, and token swapping.

FLASH 3.0

In the contract `FLASH 3.0` the role `owner` has authority over the functions shown in the list below.

- `transferOwner(address newOwner)` - transfers the ownership of the contract to a new address
- `renounceOwnership()` - renounces contract ownership and set its to address zero
- `setInitializer(address initializer)` - initializes contract state variables
- `setExcludedFromProtection(address account, bool enabled)` - updates the exclusion status of an specific address from protection mechanism
- `removeSniper(address account)` - removes an address from the sniper

- `setProtectionSettings(bool _antiSnipe, bool _antiBlock)` - updates protection status
- `excludePresaleAddresses(address router, address presale)` - excludes a given address from fees during presale mode
- `sweepContingency()` - transfers native tokens from the contract to the owner
- `sweepExternalTokens(address token)` - transfers BEP20 tokens stuck in the contract to the owner

Any compromise to the `owner` account may allow the attacker to take advantage of privileged functions allowing it to take full control over the token contract, withdraw project funds, and change important settings.

Recommendations

The advantage of privileged roles in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth noting the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private keys of the privileged accounts are compromised, it could lead to devastating consequences for the project.

The risk describes the current project design and potentially makes iterations to improve the security operations 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., multi-signature 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:

The Timelock and Multi sign (2/3, 3/5) combination *mitigates* this risk 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 being compromised;
AND
- A medium/blog link for sharing the timelock contract and multi-signature addresses with the public audience.

Long Term:

The combination of Timelock and a DAO *mitigates* this risk 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-signature addresses, and DAO information with the public audience.

Permanent:

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

- Renouncing the ownership and never claiming back the privileged roles.
OR
- Remove the risky functionality.

Note: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

FINDINGS | FLASH 3.0



8

Total Findings

0

Critical

2

Major

0

Medium

6

Minor

0

Informational

This report has been prepared to discover issues and vulnerabilities for Flash 3.0. Through this audit, we have uncovered 8 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
FLA-06	Initial Token Distribution	Centralization	Major	● Mitigated
FLA-16	Centralization Risks In FLASH.Sol	Centralization	Major	● Acknowledged
FLA-17	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
FLA-21	Potential Denial-Of-Service Situation	Logical Issue	Minor	● Resolved
FLA-22	Usage Of <code>transfer</code> / <code>send</code> For Sending Native Tokens	Volatile Code	Minor	● Resolved
FLA-23	Incompatible Type Cast	Inconsistency	Minor	● Acknowledged
FLA-24	Unsafe Integer Cast	Incorrect Calculation	Minor	● Resolved
FLA-26	Redundant Statements	Volatile Code	Minor	● Partially Resolved

FLA-06 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	● Major	FLASH.sol (06/23): 124	● Mitigated

Description

All FLASH 3.0 tokens are sent to the deployer when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and that the team make sufficient efforts to restrict the access of the private key.

Alleviation

[FLASH Technology, 07/11/2023]: The team mitigated this issue by heeding the advice.

[CertiK, 07/11/2023]: The FLASH team has provided the info of initial token distribution. Link to the token distribution plan: <https://www.pinksale.finance/launchpad/0x9E5685074372a4BE03D31e47198B579396D7094f?chain=ETH>

Multi-sig wallet address: [0x40CA317C575ADCAf6619E5d3370477A3e0B54bC5](#)

- Signer 1: [0xd8d242663d03337397b7FAEDE4fFADeCccE0F514](#)
- Signer 2: [0x15f5aC23FEA29c0f8be082EB490D3a18d7d08892](#)
- Signer 3: [0xAA0af5f4f5C0Ff8b385d0eCa9D162749Fd4D1332](#)

2 out of 3 signers are required.

It is noted that the [0x15f5aC23FEA29c0f8be082EB490D3a18d7d08892](#) received the initial funds from [0xd8d242663d03337397b7FAEDE4fFADeCccE0F514](#) in the transaction [0xaf97243f6f84056e70b52e7e3dc5f85de009f61018eaa6cea3062c03c6164a6f](#).

Here is the link to the balances of current token holders:

<https://etherscan.io/token/0xBb19DA2482308ec02a242ACED4Fe0f09D06b12A7#balances>.

As of July 11, 2023, the holders are as follows:

- [0x9E5685074372a4BE03D31e47198B579396D7094f](#) (Pinksale Launchpad contract) holds 88,440,000 or 88.44%
- [0x40CA317C575ADCAf6619E5d3370477A3e0B54bC5](#) (Gnosis safe multisig) holds 6,474,875.153 or 6.47%
- [0x90Fd19108AC3084EF3bE770f0f51c5f8d6e70fed](#) holds 1,688,301.1 or 1.69%

- 0x3c522541055e535cEa5D84B4f076248AFb201153 holds 1,500,000 or 1.5%
- 0x970C653204b6a3563C0273555d15c4008D1Bd9b6 holds 1,216,975 or 1.22%
- 0x2Fc5023faC08527056868bD9b6bAE3bFF3909146 holds 440,235 or 0.44%
- 0xDBB3fB4f353Fde03717585cEC0Ad273394CC430f holds 239,613.747 or 0.24%

The team mentioned that for the remaining amounts:

- 11.56% will be airdropped to users
- 6% for a private sale
- and the rest is for marketing and team the team, which will not be sold for at least 1 month

Link to verify multi-sig wallet: <https://app.safe.global/new-safe/load?chain=eth>.

FLA-16 | CENTRALIZATION RISKS IN FLASH.SOL

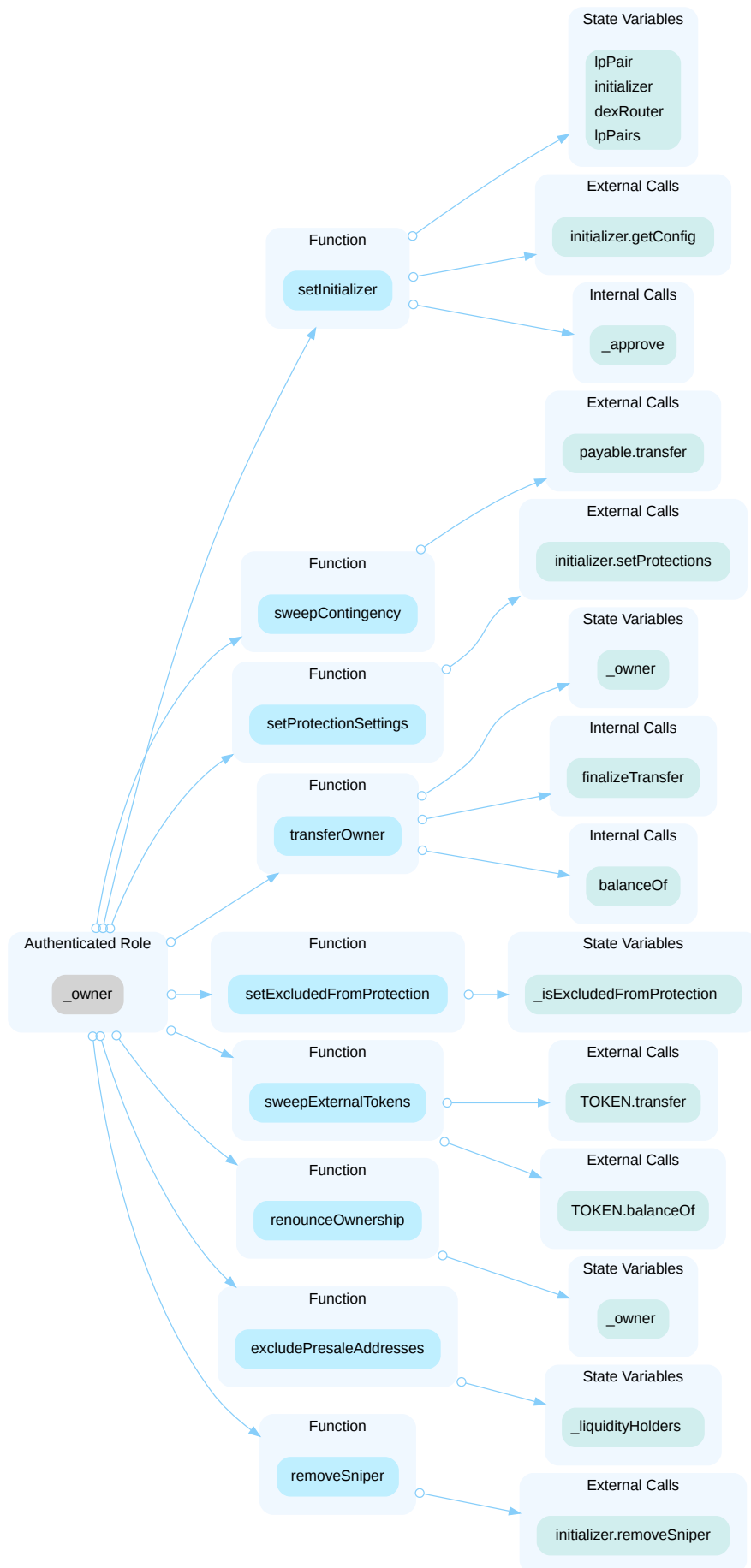
Category	Severity	Location	Status
Centralization	● Major	FLASH.sol (06/23): 149 , 162 , 209 , 225 , 231 , 237 , 244 , 343 , 347	● Acknowledged

Description

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

- `transferOwner(address newOwner)` - transfers the ownership of the contract to a new address
- `renounceOwnership()` - renounces contract ownership and set its to address zero
- `setInitializer(address initializer)` - initializes contract state variables
- `setExcludedFromProtection(address account, bool enabled)` - updates the exclusion status of an specific address from protection mechanism
- `removeSniper(address account)` - removes an address from the sniper
- `setProtectionSettings(bool _antiSnipe, bool _antiBlock)` - updates protection status
- `excludePresaleAddresses(address router, address presale)` - excludes a given address from fees during presale mode
- `sweepContingency()` - transfers native tokens from the contract to the owner
- `sweepExternalTokens(address token)` - transfers ERC20 tokens stuck in the contract to the owner

Any compromise to the `owner` account may allow the attacker to take advantage of privileged functions allowing it to take full control over the token contract, withdraw project funds, and change important settings.



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 (2/3, 3/5) 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

[FLASH Technology, 07/03/2023]: The team acknowledged this issue and will not update the codebase at current stage.

[CertiK, 07/03/2023]: The team changed some privileged functions listed below in the latest audit version:

- Removed `removeSniper(address account)`
- Removed `setProtectionSettings(bool _antiSnipe, bool _antiBlock)`
- Renamed `sweepExternalTokens(address token)` To `sweepTokens(address token)`

[CertiK, 07/12/2023]: It was noted that the FLASH contract was deployed on the mainnet and the code matches the latest commit on record: [4a2a6e7923ae14869b8b672225fe0b2291103075](https://github.com/flash-crypto/flash-contract/commit/4a2a6e7923ae14869b8b672225fe0b2291103075). The address is as follows:

<https://etherscan.io/address/0xbb19da2482308ec02a242aced4fe0f09d06b12a7>.

FLA-17 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	FLASH.sol (06/23): 214	Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

```
214 dexRouter = IRouter02(router); lpPair = constructorLP; lpPairs[
constructorLP] = true;
```

- `constructorLP` is not zero-checked before being used.

Recommendation

It is recommended to add zero-checks for the passed-in address values to prevent unexpected errors.

Alleviation

[FLASH Technology, 07/03/2023]: The team resolved this issue by adding the missing zero-checks in the latest private version.

FLA-21 | POTENTIAL DENIAL-OF-SERVICE SITUATION

Category	Severity	Location	Status
Logical Issue	● Minor	FLASH.sol (06/23): 303~305	● Resolved

Description

In the `_checkLiquidityAdd()` function, if the initializer contract is not set, the initializer contract will be set to `address(this)`. In function `finalizeTransfer`, executed at the end of every transfer, there is a check to ensure that the sender and receiver are whitelisted.

```
316         if (_hasLimits(from, to)) { bool checked;  
317             try initializer.checkUser(from, to, amount) returns (bool check) {  
318                 checked = check; } catch { revert(); }  
319             if(!checked) { revert(); }  
320         }
```

The concern is that if the `initializer` contract is not set and there isn't any implementation of the `checkUser()` function, it will cause the transfer to revert on L318 and thus block users from making transfers. On another note, this will also impact the `removeSniper()` and `setProtectionSettings()` functions since the external calls in initializer functions are also not implemented will cause these functions to revert.

Additionally, the `initializer` contract is a third-party dependency contract and not within the current audit scope. Mishandling of the initializer contract will lead to a potential DoS situation.

Recommendation

We recommend setting the initializer contract correctly or removing the incorrect assignment.

Alleviation

[FLASH Technology, 07/03/2023]: The team resolved this issue by removing the aforementioned function call in the latest private version.

FLA-22 | USAGE OF `transfer` / `send` FOR SENDING NATIVE TOKENS

Category	Severity	Location	Status
Volatile Code	● Minor	FLASH.sol (06/23): 344	● Resolved

Description

It is not recommended to use Solidity's `transfer()` and `send()` functions for transferring native tokens, since some contracts may not be able to receive the funds. Those functions forward only a fixed amount of gas (2300 specifically) and the receiving contracts may run out of gas before finishing the transfer. Also, EVM instructions' gas costs may increase in the future. Thus, some contracts that can receive now may stop working in the future due to the gas limitation.

```
344 payable(_owner).transfer(address(this).balance);
```

- FLASH.sweepContingency uses `transfer()`.

Recommendation

We recommend using the `Address.sendValue()` function from OpenZeppelin.

Since `Address.sendValue()` may allow reentrancy, we also recommend guarding against reentrancy attacks by utilizing the [Checks-Effects-Interactions Pattern](#) or applying OpenZeppelin [ReentrancyGuard](#).

Alleviation

[FLASH Technology, 07/03/2023]: The team resolved this issue by using `call()` instead of `transfer()` to transfer native tokens in the latest private version.

FLA-23 | INCOMPATIBLE TYPE CAST

Category	Severity	Location	Status
Inconsistency	Minor	FLASH.sol (06/23): 304 , 304	Acknowledged

Description

Any mismatched type or interface casting would cause failures in the contract execution.

`this` is cast to `Initializer`, however it is declared as `FLASH`.

```
304         initializer = Initializer(address(this));
```

- `this` is cast from `FLASH` to `address`.
- `address(this)` is cast from `address` to `Initializer`.

Recommendation

It is advisable to ensure that the source address incorporates all the functionalities of the target type to which it is converted.

Alleviation

[FLASH Technology, 07/04/2023]: The team acknowledged this issue and stated this is an intended design.

FLA-24 | UNSAFE INTEGER CAST

Category	Severity	Location	Status
Incorrect Calculation	● Minor	FLASH.sol (06/23): 308	● Resolved

Description

Type casting refers to changing an variable of one data type into another. The code contains an unsafe cast between integer types, which may result in unexpected truncation or sign flipping of the value.

```
308         try initializer.setLaunch(lpPair, uint32(block.number), uint64(
block.timestamp), _decimals) {} catch {}
```

Casted expression `block.number` has estimated range [0, 700800000000] but target type `uint32` has range [0, 4294967295].

Recommendation

It is recommended to check the bounds of integer values before casting. Alternatively, consider using the `SafeCast` library from OpenZeppelin to perform safe type casting and prevent undesired behavior.

Reference: <https://github.com/OpenZeppelin/openzeppelin-contracts/blob/cf86fd9962701396457e50ab0d6cc78aa29a5ebc/contracts/utils/math/SafeCast.sol>

Alleviation

[FLASH Technology, 07/03/2023]: The team resolved this issue by removing the linked function call in the latest private version.

FLA-26 | REDUNDANT STATEMENTS

Category	Severity	Location	Status
Volatile Code	● Minor	FLASH.sol (06/23): 129 , 214 , 215~216 , 277~278 , 303~305	● Partially Resolved

Description

The linked statements do not affect the functionality of the codebase and appear to be either remnants of test code or older functionality.

Recommendation

We recommend that the team review the mentioned lines of code to determine if they will be used and to remove any unused code to better prepare the code for production environments and improve readability.

Alleviation

[FLASH Technology, 07/04/2023]: The team partially resolved this issue by removing some of the linked statements in the latest private version and stated that some of the unchanged statements are intended and therefore left unchanged.

APPENDIX | FLASH 3.0

Finding Categories

Categories	Description
Incorrect Calculation	Incorrect Calculation findings are about issues in numeric computation such as rounding errors, overflows, out-of-bounds and any computation that is not intended.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
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.

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.

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.

