Kambria GT Sync

Smart Contract Security Audit

Prepared by ShellBoxes

March 14th, 2023 - March 16th, 2023

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	Kambria
Version	1.0
Classification	Public

Scope

Contract Name	Contract Address
SyncGTLPModule	0x039c13ae33895D41eAAfD6640F1379d3d42080C9

Re-Audit

Contract Name	Contract Address
SyncGTLPModule	0xa9e23fae19a5c6b23be83369660b7369521b8f61

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

Kambria engaged ShellBoxes to conduct a security assessment on the Kambria GT Sync beginning on March 14th, 2023 and ending March 16th, 2023. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About Kambria

Kambria, an open innovation platform for Deep Tech.

Issuer	Kambria
Website	`https://kambria.io`
Type	Solidity Smart Contract
Documentation	Kambria GT Sync Module Document
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact				
	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

2 Findings Overview

2.1 Disclaimer

The SyncGTLPModule contract perform external calls to the XDAO contract. The Shellboxes team treated this contract as a black box and assumed that it will always behave correctly as it out of the audit scope.

2.2 Summary

The following is a synopsis of our conclusions from our analysis of the Kambria GT Sync implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.3 Key Findings

In general, the SyncGTLPModule Smart Contract is well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 2 medium-severity, 2 low-severity, 1 informational-severity vulnerabilities.

Vulnerabilities	Severity	Status
SHB.1. Uninitialized Token Can Lead To DOS	MEDIUM	Fixed
SHB.2. The ADMIN Can Manipulate The token Address	MEDIUM	Fixed
SHB.3. Missing Address Verification	LOW	Fixed
SHB.4. Missing Value Verification	LOW	Fixed
SHB.5. Lack of Role Verification in Granting and Revoking Roles Functionality	INFORMATIONAL	Fixed

3 Finding Details

SHB.1 Uninitialized Token Can Lead To DOS

- Severity : MEDIUM
- Status : Fixed
- Likelihood : 1
- Impact : 3

Description:

The token variable is supposed to be the address of the governance token, however, this variable is not initialized in the constructor. Therefore, all the functions that rely on this variable will have a Denial of Service, as it will be equal to the address(0) before the admin sets the token to the correct address.

Files Affected:

SHB.1.1: SyncGTLPModule.sol

```
14 Dao public token;
```

Recommendation:

Consider initializing the token variable to the governance token address in the constructor.

Updates

The Kambria team resolved the issue by initializing the token variable in the constructor.

SHB.1.2: SyncGTLPModule.sol

```
899 constructor(IDao _token) {
900     require(address(_token) != address(0x0), "Token must be different
        ↪ from address 0 ");
901     token = _token;
```

SHB.2 The ADMIN Can Manipulate The token Address

- Severity : MEDIUM
- Status : Fixed
- Likelihood : 1
- Impact : 3

Description:

The setGT function allows the admin to modify the token address at any time. However, the contract documentation already specifies one address related to the DAO token. this represents a significant centralization risk, as the admin can input any malicious token.

Files Affected:

SHB.2.1: SyncGTLPModule.sol

```
26 function setGT(Dao _token) public onlyRole(ADMIN_ROLE) returns (bool) {
27     token = _token;
28     return true;
29 }
```

Recommendation:

As the address is known before deployment, it is recommended to set the address only once in the constructor or to hard-code it in the contract.

Updates

The Kambria team resolved the issue by removing the setGT function to prevent any modification.

SHB.3 Missing Address Verification

- Severity : LOW
- Status : Fixed
- Likelihood : 1
- Impact : 2

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. The setGT function should verify the _token argument to be different from the address(0), Also, the mintList, burnList, and the sync functions should make sure that the address value of the array arguments is different from the address(0).

Files Affected:

SHB.3.1: SyncGTLPModule.sol

```
26 function setGT(Dao _token) public onlyRole(ADMIN_ROLE) returns (bool) {
27     token = _token;
28     return true;
29 }
```

SHB.3.2: SyncGTLPModule.sol

```
30 function mintList(address[] memory holders, uint256[] memory amounts)
       ↪ external onlyRole(SYNC_ROLE){
```

SHB.3.3: SyncGTLPModule.sol

```
43  function burnList(address[] memory holders, uint256[] memory amounts)
        ↪ external onlyRole(SYNC_ROLE){
```

SHB.3.4: SyncGTLPModule.sol

```
56  function sync(address[] memory minters, uint256[] memory mintAmounts,
        ↪ address[] memory burners, uint256[] memory burnAmounts) external
        ↪ onlyRole(SYNC_ROLE){
```

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Updates

The Kambria team resolved the issue by verifying the addresses provided in the arguments to be different from the address(0).

SHB.3.5: SyncGTLPModule.sol

```
919 for (uint256 i = 0; i < holders.length; ++i) {
920     require(amounts[i] > 0, "All amount must be greater than 0");
921     require(holders[i] != address(0x0), "Holder must be different from
        ↪ address 0 ");
```

SHB.4 Missing Value Verification

- Severity : LOW
- Status : Fixed
- Likelihood : 1
- Impact : 2

Description:

Certain functions lack a value safety check, the values of the arguments should be verified to allow only the ones that comply with the contract's logic. In the mintList, burnList, and sync functions, the contract must ensure that the values of the amounts array are different from zero, and that the holders and the amounts arrays have the same length, the same goes for the minters with the mintAmounts array and the burners with burnAmounts array.

Files Affected:

SHB.4.1: SyncGTLPModule.sol

```
30 function mintList(address[] memory holders, uint256[] memory amounts)
       ↪ external onlyRole(SYNC_ROLE){
```

SHB.4.2: SyncGTLPModule.sol

```
43  function burnList(address[] memory holders, uint256[] memory amounts)
        ↪ external onlyRole(SYNC_ROLE){
```

SHB.4.3: SyncGTLPModule.sol

```
56  function sync(address[] memory minters, uint256[] memory mintAmounts,
        ↪ address[] memory burners, uint256[] memory burnAmounts) external
        ↪ onlyRole(SYNC_ROLE){
```

Recommendation:

We recommend that you verify the values provided in the arguments. The issue can be addressed by utilizing require statements.

Updates

The Kambria team resolved the issue by verifying the arguments' values and lengths to match the contract's logic.

SHB.4.4: SyncGTLPModule.sol

```
914 function _mintOrBurn(string memory _selector, address[] memory holders,
       ↪ uint256[] memory amounts) private {
915     require(holders.length == amounts.length, "Arrays length mismatch");
916     require(holders.length != 0, "Holder array is empty");
917     require(amounts.length != 0, "Amount array is empty");
918     bytes memory selector = abi.encodeWithSignature(_selector);
919     for (uint256 i = 0; i < holders.length; ++i) {
920         require(amounts[i] > 0, "All amount must be greater than 0");
```

SHB.5 Lack of Role Verification in Granting and Revoking Roles Functionality

- Severity : INFORMATIONAL
- Status : Fixed
- Likelihood : 1
- Impact : 0

Description:

The contract's current implementation of role-based access control does not include verification checks when granting or revoking roles using the grantRole and revokeRole functions. This allows the ADMIN_ROLE to add any arbitrary role to the contract, which may not be used in the contract.

Files Affected:

SHB.5.1: SyncGTLPModule.sol

```
20 function grantRole(bytes32 role, address account) public virtual
       ↪ override onlyRole(ADMIN_ROLE) {
21     _grantRole(role, account);
22 }
23 function revokeRole(bytes32 role, address account) public virtual
       ↪ override onlyRole(ADMIN_ROLE){
24   _revokeRole(role, account);
25 }
```

Recommendation:

Consider adding a validation check to the grantRole and revokeRole functions to ensure that only predefined roles can be granted/revoked.

Updates

The Kambria team resolved the issue by verifying the role argument to be either the ADMIN_ROLE or the SYNC_ROLE.

SHB.5.2: SyncGTLPModule.sol

```
906 function grantRole(bytes32 role, address account) public virtual
        ↪ override onlyRole(ADMIN_ROLE) {
907         require((role == ADMIN_ROLE)(role == SYNC_ROLE), "Role is not
                ↪ predefined");
```

4 Best Practices

BP.1 Remove The Unused Ownable Contract

Description:

The Ownable contract is a module which provides a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions. However, the contract does not make use of the onlyOwner modifier, it is recommended to remove this contract as it is not used in the contract's logic.

Files Affected:

BP.1.1: SyncGTLPModule.sol

```
41 contract SyncGTLPModule is Ownable, AccessControl{
```

Status - Fixed

Updates

The Kambria team implemented the best practice by removing the unused Ownable contract.

BP.2 Use Pre-increment Instead Of Post-increment

Description:

i++ is generally more expensive because it must increment a value and "return" the old value, so it may require holding two numbers in memory. ++i only ever uses one number in memory, therefore, ++i consumes less gas than i++. Make sure to replace i++ with ++i in all for loops.

Status - Fixed

Updates

The Kambria team implemented the best practice by using the pre-increment instead of post-increment.

BP.3 Use The DAO Interface For External Calls

Description:

The contract imports the Dao.sol contract in order to perform an external call to the executePermitted function. It is recommended to make use of an interface of the XDao contract to perform external calls instead of importing all the contract's code.

Files Affected:

BP.3.1: SyncGTLPModule.sol

```
3 import "./Dao.sol";
```

Status - Fixed

Updates

The Kambria team implemented the best practice by only importing the interface of the XDao contract instead of importing all the contract code.

BP.4 Refactor Contract Functions for Improved Readability and Maintainability

Description:

The contract has three functions that perform similar operations, namely, mintList, burnList, and sync. Each function iterates over two arrays of addresses and amounts and calls

the executePermitted function on the governance token contract, which mints or burns tokens. However, the code inside each iteration is almost identical, except for the function selector used. This leads to code duplication, which makes the code harder to read and maintain.

A better approach is to create another function that takes as input an array of addresses and amounts and a string indicating whether the operation is a mint or a burn. This function should perform the iteration and the call to executePermitted, with the appropriate signature. Then, the three existing functions can call this internal function with the appropriate arguments. This approach reduces code duplication and makes the code easier to read and maintain.

Status - Fixed

Updates

The Kambria team followed the best practice by implementing the use of the recommended code.

BP.5 Remove Unused Imported Files

Description:

The contract imports the IERC20, SafeERC20, and ERC20 contracts from the openzeppelin repository. However, these contracts are not inherited or used anywhere in the SyncGTLPModule contract, consider removing those imports as they do not provide an additional value to the contract.

Files Affected:

BP.5.1: SyncGTLPModule.sol

```
6 import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
7 import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
8 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
```

Status - Fixed

Updates

The Kambria team implemented the best practice by removing the unused imported file.

5 Tests

→ Set GT

✓ Set the GT token Smartcontract Address

→ Read GT

✓ View the GT address and see if the address is contract …

→ Add permited list

✓ Add the sync module to the permited list on XDAO …

→ Mint list

✓ Enter the list of to be mint address and to be mint amount and execute the Mint function …

→ Burn list

✓ Enter the list of to be burn address and to be burn amount and execute the Mint function …

→ Sync function

✓ Do both Mint list and Burn list function …

6 Conclusion

In this audit, we examined the design and implementation of Kambria GT Sync contract and discovered several issues of varying severity. Kambria team addressed all the issues raised in the initial report and implemented the necessary fixes.

However Shellboxes' auditors advised Kambria Team to maintain a high level of vigilance and participate in bounty programs in order to avoid any future complications.

7 Scope Files

7.1 Audit

Files	MD5 Hash
SyncGTLPModule.sol	166238f6eec00a29a671a23e8e131a05

7.2 Re-Audit

Files	MD5 Hash
SyncGTLPModule2.sol	b80855ef7938bff3d5fcd4d58ab63a8d

8 Disclaimer

Shellboxes reports should not be construed as ”endorsements” or ”disapprovals” of particular teams or projects. These reports do not reflect the economics or value of any ”product” or ”asset” produced by any team or project that engages Shellboxes to do a security evaluation, nor should they be regarded as such. Shellboxes Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the examined technology, nor do they provide any indication of the technology's proprietors, business model, business or legal compliance. Shellboxes Reports should not be used in any way to decide whether to invest in or take part in a certain project. These reports don't offer any kind of investing advice and shouldn't be used that way. Shellboxes Reports are the result of a thorough auditing process designed to assist our clients in improving the quality of their code while lowering the significant risk posed by blockchain technology. According to Shellboxes, each business and person is in charge of their own due diligence and ongoing security. Shellboxes does not guarantee the security or functionality of the technology we agree to research; instead, our purpose is to assist in limiting the attack vectors and the high degree of variation associated with using new and evolving technologies.

For a Contract Audit, contact us at contact@shellboxes.com