CERTIK

Security Assessment

# Venus - XVS Token Bridge

CertiK Assessed on Dec 26th, 2023

CertiK Assessed on Dec 26th, 2023

# Venus - XVS Token Bridge

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

## Executive Summary

| TYPES | ECOSYSTEM | METHODS |
|---|---|---|
| DeFi | Binance Smart Chain (BSC) | Manual Review, Static Analysis |
| **LANGUAGE** | **TIMELINE** | **KEY COMPONENTS** |
| Solidity | Delivered on 12/26/2023 | N/A |

CODEBASE
https://github.com/VenusProtocol/token-bridge/
View All in Codebase Page

COMMITS
base: af1661c65247b05b596b324b4ed8973add250d01
update1: 4ef1a4c27fd824d8d4915f44a0c6c37d714a07c0
update2: 66e0995d302b91621362d18d58db2cea634b4c2b
View All in Codebase Page

## Highlighted Centralization Risks

- Privileged role can mint tokens
- Has blacklist/whitelist
- Privileged role can remove users' tokens

## Vulnerability Summary

| 15 | 14 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|
| Total Findings | Resolved | Mitigated | Partially Resolved | Acknowledged | Declined |

| | | | |
|---|---|---|---|
| 0 | Critical | | Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks. |
| 1 | Major | 1 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. |
| 1 | Medium | 1 Resolved | Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform. |
| 6 | Minor | 6 Resolved | 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. |

| | | | |
|---|---|---|---|
| 7 | Informational | 7 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 | VENUS - XVS TOKEN BRIDGE

# CODEBASE | VENUS - XVS TOKEN BRIDGE

## Repository

https://github.com/VenusProtocol/token-bridge/

## Commit

base: af1661c65247b05b596b324b4ed8973add250d01

update1: 4ef1a4c27fd824d8d4915f44a0c6c37d714a07c0

update2: 66e0995d302b91621362d18d58db2cea634b4c2b

update3: da1310f1c2c5514eea111df238cfd7aadf9100b3

update4: e65ada340d44c9fd93f7a88a63bdee7421a55673

update5: 8b8dbce9f75029203e6011f34a58a64684fc3379

# AUDIT SCOPE | VENUS - XVS TOKEN BRIDGE

8 files audited ● 6 files with Acknowledged findings ● 2 files without findings

| ID | Repo | Commit | File | SHA256 Checksum |
|---|---|---|---|---|
| BXV | VenusProtocol/token-bridge | af1661c | BaseXVSProxyOFT.sol | f3570b5fc9af047e9284d0c7b8a4a96c6533414ff3e609739b5c4bfafeafe41a |
| XVS | VenusProtocol/token-bridge | af1661c | XVSBridgeAdmin.sol | 5b63915b73aeb692d1aa9e7378bf94e92b291475aaa03b971a84ee5d96149946 |
| XVP | VenusProtocol/token-bridge | af1661c | XVSProxyOFTDest.sol | a2ef0adc20eecde6e7854c97d7f6f1a0fd73d4b73a0387e79bed04595e3b7a2c |
| XVO | VenusProtocol/token-bridge | af1661c | XVSProxyOFTSrc.sol | 783d1f166fbba14af6ded76b7a077870b7184a6c045332596b157d19b0921476 |
| TCB | VenusProtocol/token-bridge | af1661c | token/TokenController.sol | 71150cb0e2450bcdd1511778bfb103895aee70ad596fd4c758b7231b01c1ad8b |
| XVB | VenusProtocol/token-bridge | af1661c | token/XVS.sol | eb14d9d3fc8404e38f128a1f7edd4e61bef98925dac542de408a516536afea20 |
| IXV | VenusProtocol/token-bridge | af1661c | interfaces/IXVS.sol | 78315aec9118f464babb9b54e4daea262637645cc2364d727000e1300836f515 |
| IXS | VenusProtocol/token-bridge | af1661c | interfaces/IXVSProxyOFT.sol | 484b3e1e1f67189dfb9406e1219b75e485153a638585f7b1ed1a20ff383c539b |

# APPROACH & METHODS | VENUS - XVS TOKEN BRIDGE

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - XVS Token Bridge 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.

# DEPENDENCIES | VENUS - XVS TOKEN BRIDGE

## Third Party Dependencies

The protocol is serving as the underlying entity to interact with third party protocols. The third parties that the contracts interact with are:

- Oracles through the use of the protocol's Resilient Oracle
- Layer Zero endpoint relayers
- ERC20 Tokens

The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. Moreover, updates to the state of a project contract that are dependent on the read of the state of external third party contracts may make the project vulnerable to read-only reentrancy. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

## Out Of Scope Dependencies

The protocol is serving as the underlying entity to interact with out-of-scope dependencies. The out-of-scope dependencies that the contracts interact with are:

- Layer Zero contract inheritance `BaseOFTV2`.
- Updates made to the Resilient Oracle to accommodate use on other chains.

The scope of the audit treats out-of-scope dependencies as black boxes and assumes their functional correctness.

## Assumptions

Within the scope of the audit, assumptions are made about the intended behavior of the protocol in order to inspect consequences based on those behaviors. Assumptions made within the scope of this audit include:

- Only the `XVS` token currently deployed at 0xcf6bb5389c92bdda8a3747ddb454cb7a64626c63 will be used as the `innerToken` of the `XVSProxyOFTSrc` contract.
- Only the `XVS` token contract in-scope of this audit will be used as the `innerToken` of the `XVSProxyOFTDest` contract.

## Recommendations

We recommend constantly monitoring the third parties involved to mitigate any side effects that may occur when unexpected changes are introduced, as well as vetting any third party contracts used to ensure no external calls can be made before updates to its state. Additionally, we recommend all out-of-scope dependencies are carefully vetted to ensure they function

as intended. Last, we recommend all assumptions about the behavior of the project are thoroughly reviewed and, if the assumptions do not match the intention of the protocol, documenting the intended behavior for review.

# FINDINGS | VENUS - XVS TOKEN BRIDGE

| 15 | 0 | 1 | 1 | 6 | 7 |
|---|---|---|---|---|---|
| Total Findings | Critical | Major | Medium | Minor | Informational |

This report has been prepared to discover issues and vulnerabilities for Venus - XVS Token Bridge. Through this audit, we have uncovered 15 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 |
|---|---|---|---|---|
| **VPB-01** | **Centralization Related Risks** | **Centralization** | **Major** | **Acknowledged** |
| TCB-02 | Function _increaseMintLimit() Does Not Handle Possible Update To `minterToCap` | Logical Issue | Medium | Resolved |
| BRI-07 | Discrepancy With Blacklisting In XVS | Logical Issue | Minor | Resolved |
| VPB-02 | Missing Input Validation | Volatile Code | Minor | Resolved |
| XVB-02 | Pausing Behavior In `XVS` | Design Issue | Minor | Resolved |
| XVF-01 | Check-Effect-Interaction Pattern Violated | Concurrency | Minor | Resolved |
| XVF-02 | Inconsistency With `cap` Check | Logical Issue | Minor | Resolved |
| XVO-02 | Tokens Become Locked In The Contract When `toAddress_` Is `address(this)` | Logical Issue | Minor | Resolved |
| BRI-01 | Discussion On `XVSProxyOFTDest` Vs. `XVSProxyOFTSrc` | Logical Issue | Informational | Resolved |
| BRI-02 | Typos And Inconsistencies | Inconsistency | Informational | Resolved |
| BRI-03 | Missing Or Incomplete NatSpec | Inconsistency | Informational | Resolved |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| BRI-08 | Consider Reverting As Opposed To Having Empty Implementaton | Coding Style | Informational | Resolved |
| BVP-01 | `minterToCap` Must Be Updated Consistently With Access To `mint()` And `burn()` Functions | Inconsistency | Informational | Resolved |
| GLOBAL-01 | Discussion On Design | Inconsistency | Informational | Resolved |
| XVS-03 | Cases Not Explicitly Handled | Inconsistency | Informational | Resolved |

# VPB-01 | CENTRALIZATION RELATED RISKS

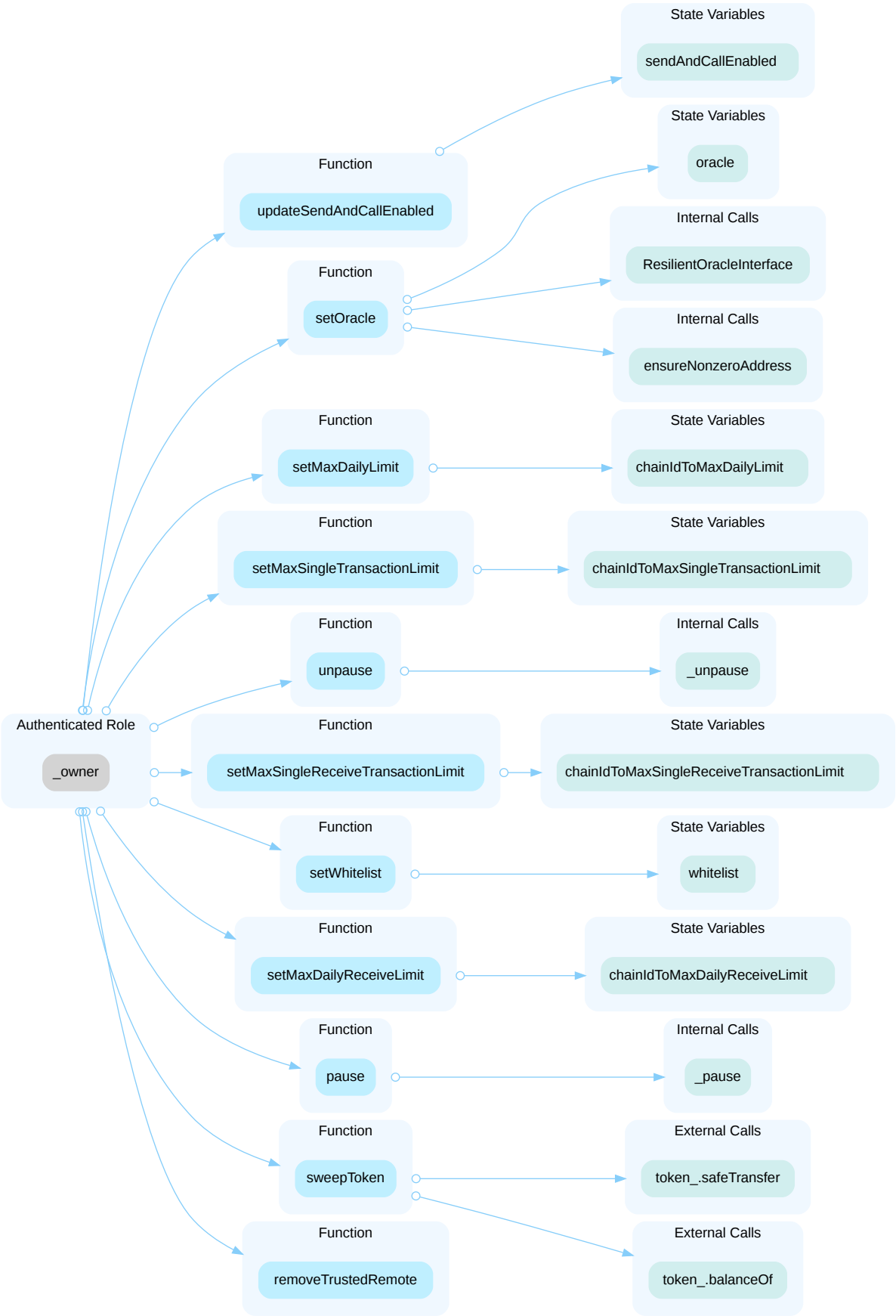
| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization | Major | BaseXVSProxyOFT.sol (base): 143, 156, 169, 182, 195, 211, 220, 228, 236; XVSBridgeAdmin.sol (base): 50, 62, 73, 95; XVSProxyOFTDest.sol (base): 33; XVSProxyOFTSrc.sol (base): 44, 55; token/TokenController.sol (base): 81~82, 90~91, 102~103, 115~116, 129; token/XVS.sol (base): 28~29, 42~43; BaseXVSProxyOFT.sol (update3): 255~256; XVSProxyOFTSrc.sol (update3): 67~68; token/TokenController.sol (update3): 157~158 | Acknowledged |

## Description

### BaseXVSProxyOFT

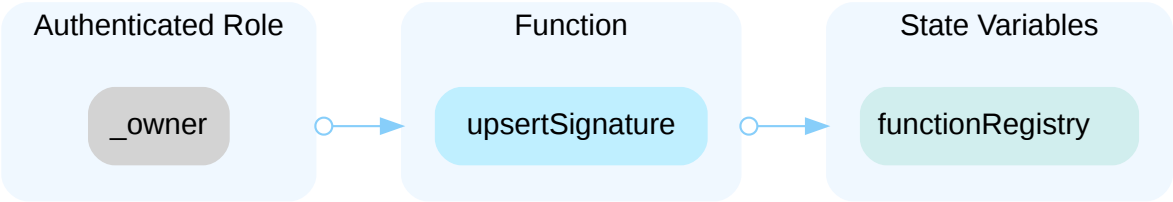
In the contract `BaseXVSProxyOFT` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and

- Set the oracle to an address that does not provide accurate pricing information.
- Pause or unpause the movement of tokens through the contract at critical periods.
- Set the max daily limit, the max daily receive limit, the max single transaction limit, and the max single receive transaction limit to values that either are advantageous to the hacker moving funds across chains, or the hacker may set these values to 0 to prevent users from moving funds out of a vulnerable environment.
- Set an address they control to be whitelisted and circumvent the bridged amount checks.
- Set or remove the trusted remote address, either allowing a malicious remote address or denying service to a trusted remote address.
- Remove tokens that correspond to the `outboundAmount` without updating the `outboundAmount`, creating a discrepancy where it appears more tokens are locked in the source bridge than actually are.
- Enable or disable the use of `sendAndCall()`.




## XVSBridgeAdmin

In the contract `XVSBridgeAdmin` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and make a function signature active within the `functionRegistry` mapping in order to give their account access to this bridge logic through the privilege outlined below.




In the contract `XVSBridgeAdmin` the role `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the following functions:

- Functions within the XVS Bridge via the `fallback()` function in `XVSBridgeAdmin`
- `setTrustedRemoteAddress()`
- `transferBridgeOwnership()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or accounts granted this privilege may allow the hacker to take advantage of this authority and do the following:

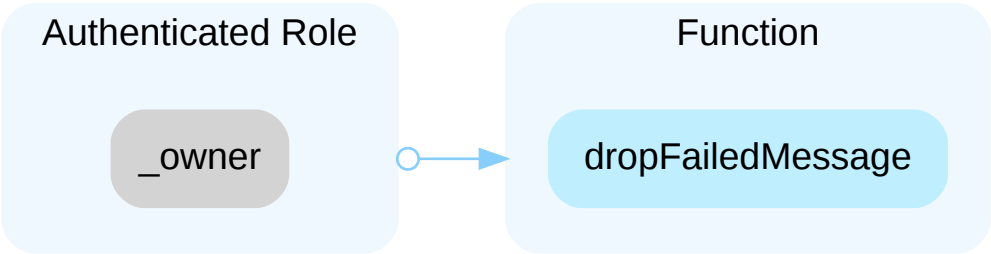
- Call privileged functions that the `XVSBridgeAdmin` controls within the XVS Bridge, potentially allowing the hacker to steal funds locked in the bridge or using the bridge minting privilege to steal funds from a pool on one of the bridged chains.
- Add a malicious trusted remote contract address in order to take steps to do the above.
- Give an account they access the owner privilege of the bridge in order to access other functions so they can take steps to perform the above.

---

### XVSProxyOFTDest

In the contract `XVSProxyOFTDest` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and

- Remove a failed message from the system so that it cannot be retried.

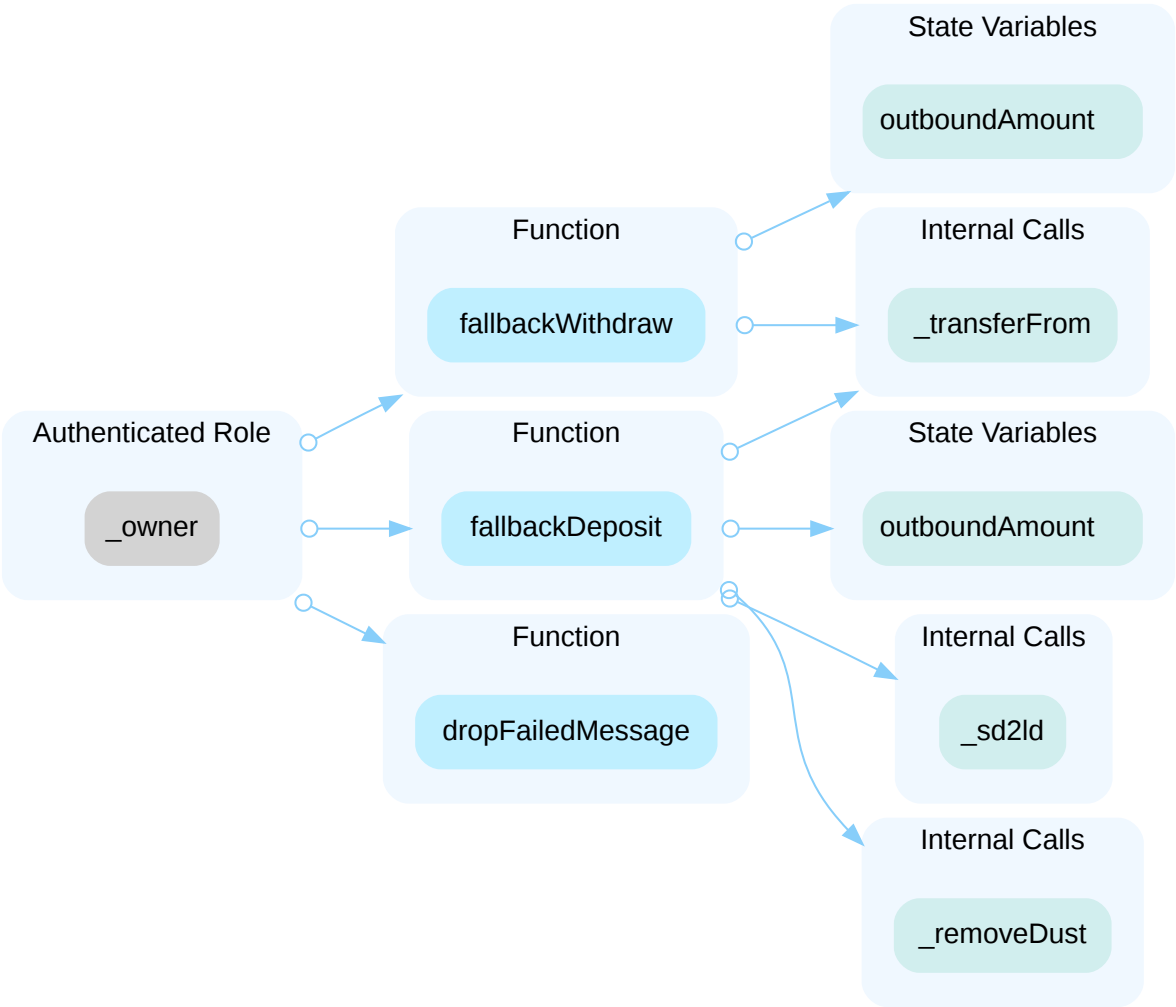



---

### XVSProxyOFTSrc

In the contract `XVSProxyOFTSrc` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and

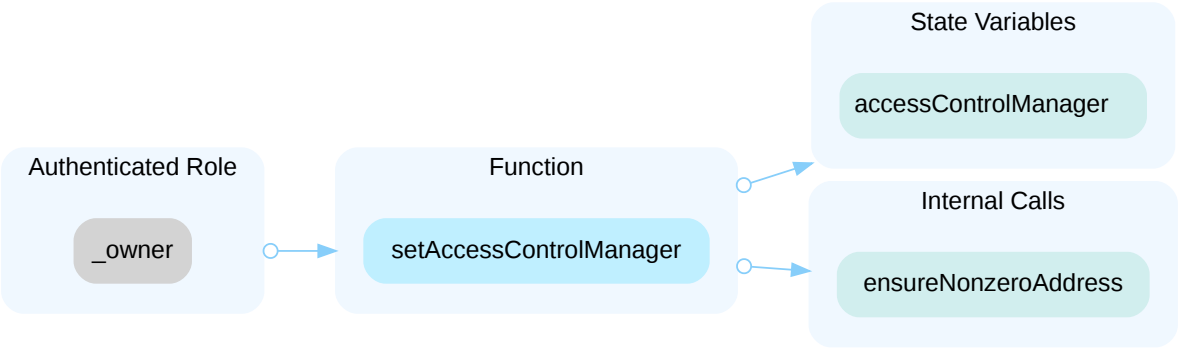
- Remove a failed message from the system so that it cannot be retried.
- Remove any amount of XVS funds currently held by the contract.
- Use the function to inaccurately update the `outboundAmount` through a combination of calls to `sweepToken()` and `fallbackDeposit()`. This may cause a discrepancy where the `outboundAmount` is far larger than the balance the contract actually holds.




---

### TokenController

In the contract `TokenController` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set the `accessControlManager` to an address they control so that they can make updates via the functions outlined below.




In the contract `TokenController`, the role `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the following functions:

- `pause()`
- `unpause()`
- `updateBlacklist()`
- `setMintCap()`
- `migrateMinterTokens()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or accounts granted this privilege may allow a hacker to take advantage of this authority and

- Pause or unpause the contract at a critical moment.
- Blacklist users so that funds bridged over cannot be minted to their accounts.
- Update the mint cap of the bridge to prevent funds from being bridged over, or give an account they control a large mint cap in order to mint funds within the chain.
- Transfer records of previously minted tokens to a malicious contract address in order to allow the address to burn tokens belonging to any account. With control of additional privileges, this could potentially be used to maliciously bridge tokens to other chains.

---

## XVS

In the contract `XVS`, the role `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the following functions:

- `mint()`
- `burn()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or accounts granted this privilege may allow a hacker to take advantage of this authority and

- Mint funds to an account they control.

- Burn the tokens of other users without a corresponding bridging action on another chain.

## Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

### Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

### Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

### Permanent:

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

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

## Alleviation

`[Venus, 12/15/2023]`: "The owner of the bridge contracts (XVSProxyOFTSrc, XVSProxyOFTDest) will be an instance of the contract XVSBridgeAdmin. The owner of XVSBridgeAdmin will be the Normal Timelock contract [1]. The privilege functions in the proxy contract will be executable only by the Normal, Fast-track and Critical timelock contracts [2]. These permissions will be granted via the AccessControlManager contract [3].

Default Admin role of ACM is granted to Normal Timelock only.

The owner of the TokenController contract will be the Normal Timelock contract.

[1] on BNB chain, the Normal Timelock is deployed at https://bscscan.com/address/0x939bD8d64c0A9583A7Dcea9933f7b21697ab6396

[2] on BNB chain the Fast-track timelock is deployed at https://bscscan.com/address/0x555ba73dB1b006F3f2C7dB7126d6e4343aDBce02 and the Critical timelock is deployed at https://bscscan.com/address/0x213c446ec11e45b15a6E29C1C1b402B8897f606d

[3] on BNB chain the ACM contract is deployed at https://bscscan.com/address/0x4788629abc6cfca10f9f969efdeaa1cf70c23555"

`[CertiK, 12/18/2023]`: The client has provided all steps towards mitigation on the BSC chain. In order to mitigate the finding completely, please provide the relevant information corresponding to the destination chains in which the bridge will initially be deployed.

# TCB-02 FUNCTION _INCREASEMINTLIMIT() DOES NOT HANDLE POSSIBLE UPDATE TO `minterToCap`

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Medium | token/TokenController.sol (base): 117~118, 175~176 | Resolved |

## Description

Function `_increaseMintLimit()` is used within the `burn()` function of the in-scope `XVS` token contract as a way to document the increase in available funds to mint after burning. The function logic assumes that the `minterToCap` mapping for the source address `from` will be larger than the calculated `totalMintedNew` value after subtracting away the new `amount_` that is burned.

If in between minting and burning, the `minterToCap` corresponding value is decreased for the minter address, then it is possible that `minterToCap` is less than the calculated `totalMintedNew` value at the time of burning. In that case, the logic of `_increaseMintLimit()` will cause a revert in the `burn()` logic, preventing burning from functioning properly.

## Scenario

1. A `minter` address has a `minterToCap` value of 100, and a `minterToMintedAmount` of 90.
2. The function `setMintCap()` is used to update the `minterToCap` for the `minter` to 80.
3. Function `burn()` is called on an amount of 5 so that `totalMintedNew` is `90 - 5 = 85`.
4. The calculation `availableLimit = 80 - 85` will revert due to underflow.

## Recommendation

We recommend preventing this occurrence directly within function `setMintCap()` by checking that the input `amount_` is not less than the current `minterToMintedAmount` entry for the `minter_` address.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commit 4ef1a4c27fd824d8d4915f44a0c6c37d714a07c0.

# BRI-07 | DISCREPANCY WITH BLACKLISTING IN XVS

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | XVSProxyOFTSrc.sol (base): 74~75; token/XVS.sol (base): 30~31, 42 | Resolved |

## Description

In the `XVS` contract, a blacklisted account cannot be minted tokens. However, on the chain in which the account is blacklisted, anyone can still transfer tokens to the blacklisted account, transfer tokens from the blacklisted account, the account can be approved to move others' tokens, and the account can have their tokens burned. As a consequence, the blacklisted account can easily acquire tokens, and they can still bridge these tokens to another chain where they may not be blacklisted.

Moreover, in `XVSProxyOFTSrc`, there is no check from this side of whether the account bridging tokens is blacklisted on other chains, and there is no way to blacklist the user from the source chain. As a result, a user can transfer in their tokens into the `XVSProxyOFTSrc` contract, but if they are blacklisted on the destination chain, there is no equivalent amount given out. In this way, these tokens become stuck within the protocol and cannot be removed without intervention via centralized accounts.

## Recommendation

We recommend preventing the transfer of tokens to and from a blacklisted account, and preventing burning by a blacklisted account.

Additionally, within the `XVSProxyOFTSrc`, we recommend implementing a method for blacklisting accounts to prevent the initial transfer of funds to this contract, and blacklisting any account consistently across all chains in which the token can be bridged.

## Alleviation

`[CertiK, 12/19/2023]`: The client made changes resolving the finding in commits

- 52ff1120d828b50ff110acb2df939abe28a3dd28
- f659dee63bd5a3713e7ad0f81fa9bc515b0e8e62
- 51168baa96a6ce64b5448726af26c3cdc105618e
- 66e0995d302b91621362d18d58db2cea634b

`[Venus, 12/15/2023]`: "Regarding the second recommendations (on XVSProxyOFTSrc), In case the user is blacklisted in the destination chain, the tokens will stay locked in the src chain and then we have control on how we want to manage the funds locked"

# VPB-02 | MISSING INPUT VALIDATION

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Minor | XVSProxyOFTSrc.sol (base): 44, 87~88; token/TokenController.sol (base): 173~174; token/TokenController.sol (update3): 157~158 | Resolved |

## Description

### TokenController.sol

- In function `_increaseMintLimit()` there is a missing check that the input `amount_` is no larger than the `minterToMintedAmount` for the input `from_` address. This may cause an ambiguous error message due to underflow in the calculation of `totalMintedNew` in the function.
- In function `migrateMinterTokens()`, there is no check that the `source_` and `destination_` addresses are not the same. If they are the same address, the update to mapping `minterToMintedAmount` for that address will not accurately reflect the minted amount (e.g. an address with `mintedToMinted` amount of 100 originally will double to 200 as long as the `destinationCap` is not exceeded).

### XVSProxyOFTSrc.sol

- In the function `fallbackWithdraw()` there is no check that `amount_` is no larger than `outboundAmount`. In this case, an ambiguous error message due to underflow will be thrown.
- In the function `_creditTo()` there is no check that `amount_` is no larger than `outboundAmount`. In this case, an ambiguous error message due to underflow will be thrown.

## Recommendation

We recommend adding the missing checks outlined above.

## Alleviation

`[Venus, 12/15/2023]`:

- XVSProxyOFTSrc.sol: creditTo Once the tokens are bridged from src chain the outbound amount will always be greater than amount which is bridged from destination chain

`[CertiK, 12/18/2023]`: The client clarified or made changes resolving the finding in commits

- 85a718c299a1e65034b52c9494bc8479adfb3a87;

- 981401f49eef5427643eb5158d1edbae680bc358.

# XVB-02 | PAUSING BEHAVIOR IN `XVS`

| Category | Severity | Location | Status |
|---|---|---|---|
| Design Issue | Minor | token/XVS.sol (base): 28, 42 | Resolved |

## Description

`mint()` and `burn()` can be paused, however users can still transfer tokens when the contract is paused. We would like to ensure that this is the intended behavior.

## Recommendation

We recommend clarifying whether the above case is intended behavior.

## Alleviation

`[CertiK, 12/18/2023]`: The client clarified the behavior outlined above is not intended and made changes resolving the finding in commit f57a9ab2d4688ba7269abcaaeced316aad1ccbc4.

# XVF-01 | CHECK-EFFECT-INTERACTION PATTERN VIOLATED

| Category | Severity | Location | Status |
|---|---|---|---|
| Concurrency | Minor | XVSProxyOFTSrc.sol (update3): 69~75 | Resolved |

## Description

An external call to transfer the `innerToken` is made first, and then the `outboundAmount` is updated and checked. While it is assumed that the `innerToken` is the team's `XVS` token, following the check-effect-interaction pattern can reduce any unforeseen risk.

## Recommendation

We recommend following the check-effect-interaction pattern and calling `_transferFrom()` after necessary updates to state and checks have been performed within the protocol.

## Alleviation

`[CertiK, 12/22/2023]`: The client made changes resolving the finding in commit: e65ada340d44c9fd93f7a88a63bdee7421a55673.

# XVF-02 | INCONSISTENCY WITH `cap` CHECK

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | XVSProxyOFTSrc.sol (update3): 72~74 | Resolved |

## Description

In the function `fallbackDeposit()` it is checked that the new `outboundAmount` after the deposit will not exceed the `_sd2ld(type(uint64).max)`. As some chains like Aptos use `uint64` to represent balances, the maximum balance on their chain would be `type(uint64).max` and thus `_sd2ld(type(uint64).max)` represents this maximum amount using the local decimals.

However, this check is not performed in `_debitFrom()`, so that it is possible to exceed this cap using normal chain operations and then preventing `fallbackDeposit()` from being called.

## Recommendation

We recommend checking the `outboundAmount` does not exceed the cap in the `_debitFrom()` function.

## Alleviation

`[CertiK, 12/26/2023]`: The client made the recommended changes in commit 8b8dbce9f75029203e6011f34a58a64684fc3379.

Note that this ensures that the amount bridged from this bridge will not exceed the cap. However, if multiple bridges are used, it may be possible for the total amount bridged across all bridges to exceed this cap.

# XVO-02 | TOKENS BECOME LOCKED IN THE CONTRACT WHEN `toAddress_` IS `address(this)`

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | XVSProxyOFTSrc.sol (base): 45~46, 87~91 | Resolved |

## Description

In the case where XVS tokens are bridged to the `XVSProxyOFTSrc` contract itself, the `_creditTo()` logic will subtract the `amount_` from the `outboundAmount` while keeping the corresponding `XVS` in the contract, in order to account for the fact that it is no longer bridged to one of the destination chains, and is now technically part of the `circulatingSupply()` on the source chain.

However, the only way to remove tokens bridged to the `XVSProxyOFTSrc` contract itself would be to use the `fallbackWithdraw()` function, in which case, the `amount_` would be subtracted from the `outboundAmount` a second time, causing a discrepancy in the accounting of the variable.

Consequently, any funds a user accidentally sends directly to the bridge rather than sending to their own account will cause the tokens to be permanently locked in the bridge, or else will cause an issue with `outboundAmount` where eventually someone else's tokens will be permanently locked in the bridge.

Additionally, if anyone directly transfers tokens to the bridge, they will not be accounted for in the `outboundAmount`, so they also cannot be removed with `fallbackWithdraw()` without causing an issue with this variable.

## Recommendation

We recommend providing a privileged method for removal of tokens from the `XVSProxyOFTSrc` contract which are not accounted for in the `outboundAmount`.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commit 06c6009e01411182d738b2249cbbb06019926b54.

# BRI-01 | DISCUSSION ON `XVSProxyOFTDest` VS. `XVSProxyOFTSrc`

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Informational | XVSProxyOFTDest.sol (base): 15; XVSProxyOFTSrc.sol (base): 16 | Resolved |

## Description

There are currently two bridge contracts `XVSProxyOFTSrc` and `XVSProxyOFTDest`. The main difference being that `XVSProxyOFTSrc` locks tokens if transferring to another chain and gives locked tokens when receiving tokens from another chain, while `XVSProxyOFTDest` burns or mints tokens when transffering to another chain or receiving from another chain respectively.

As such we assume that the design is to deploy a single `XVSProxyOFTSrc` on BSC as it holds the current supply of XVS and all other chains should only receive XVS that has been bridged out of BSC. Furthermore we assume that all other chains will have a single `XVSProxyOFTDest` contract deployed to bridge XVS tokens.

Last, it is assumed that only one address (the bridge contract) will utilize a nonzero `minterToCap` value per chain, except in extenuating circumstances in which funds become stuck in the protocol based upon the restrictions of the design (such as the bridged amount exceeding the mint cap of the bridge for the chain). Furthermore, it is assumed that any use of this minting capability beyond the bridge associated with the chain will be carefully assessed in order to maintain the invariant that `totalSupply()` of XVS on BSC is equivalent to the `circulatingSupply()` of the BSC bridge contract, plus the amount `outboundAmount`. A similar assumption is made for the `fallbackWithdraw()` function in `XVSProxyOFTSrc`.

Please let us know if these assumptions are correct and provide further information if they do not align with the design intent.

## Recommendation

We recommend providing documentation on the design of the contracts to ensure proper understanding.

## Alleviation

`[CertiK, 12/18/2023]`: The client confirms that all assumptions outlined above are correct. They further state that they will publish the technical documentation found at PR 77 to the public documentation on their website, https://docs.venus.io.

# BRI-02 | TYPOS AND INCONSISTENCIES

| Category | Severity | Location | Status |
|---|---|---|---|
| Inconsistency | Informational | BaseXVSProxyOFT.sol (base): 64, 69, 103, 206; XVSBridgeAdmin.sol (base): 69, 90; XVSProxyOFTDest.sol (base): 17; XVSProxyOFTSrc.sol (base): 19, 28; token/TokenController.sol (base): 30~31, 109~110, 167~168; token/XVS.sol (base): 21~22, 25~26, 26~27 | Resolved |

## Description

### XVSProxyOFTSrc

- The comment above the variable `outboundAmount` states "total amount is transferred from this chain to other chains." which should be "Total amount that is transferred from this chain to other chains."
- In the comment above the event `DropFailedMessage`, the word "successful" is misspelled as "successfull."

### XVSProxyOFTDest

- In the comment above the event `DropFailedMessage`, the word "successful" is misspelled as "successfull."

### XVSBridgeAdmin

- The comment above the function `upsertSignature()` does not describe that this is a setter function for the registry.
- In the comment above function `transferBridgeOwnership()`, "transfer" should be "transfers".

### BaseXVSProxyOFT

- In the comment above the `whitelist` mapping, the word "applicable" is misspelled as "appicable."
- In the comment above event `SetWhitelist`, the word "emitted" is misspelled as "emmited"
- In the comment above function `setWhitelist()`, the word "address" is misspelled as "Adress"
- In the comment above the `constructor()`, "No" should either be "No." or "Number".

### TokenController.sol

- In the comment above the `minterToMintedAmount` mapping, the word "minted" is misspelled as "m inted."

- In the comment above functions `setMintCap()` and `_increaseMintLimit()` the word "minting" is misspelled as "miniting."

### XVS.sol

- In the comment above `mint()` function the description "@param account_ Address to which tokens be assigned" reads correctly as "account_ Address to which tokens are assigned."
- In the comment above `mint()`, there is an incorrect description of error `MintNotAllowed`. The error should say it is thrown when minting is not allowed to `to_` address; instead it says the error is thrown when minting is not allowed to the `from_` address.
- In the comment above `mint()`, the error description "MintLimitExceed is thrown when minting amount exceed the maximum cap" reads correctly as "MintLimitExceed is thrown when minting amount exceeds the maximum cap"

## Recommendation

We recommend fixing the typos and inconsistencies mentioned above.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commit 8ce11e679649956014e700aac2e7b622e1886970.

# BRI-03 | MISSING OR INCOMPLETE NATSPEC

| Category | Severity | Location | Status |
|---|---|---|---|
| Inconsistency | Informational | BaseXVSProxyOFT.sol (base): 101~109, 232~235, 246~248, 253, 291, 330, 344, 361; XVSBridgeAdmin.sol (base): 43~46, 56~60, 68~72, 89~93, 100~105, 117~119, 124; XVSProxyOFTDest.sol (base): 28~32, 38~40, 45, 57; XVSProxyOFTSrc.sol (base): 39~43, 50~54, 60~62, 67, 81; token/TokenController.sol (base): 144~149, 166~170, 179 | Resolved |

## Description

### XVSProxyOFTSrc

- The comments above `fallbackWithdraw()` do not include the emmited event or the access restriction.
- The comments above `dropFailedMessage()` do not include the emmited event or the access restriction.
- The comments above `circulatingSupply()` do not include the return value.
- There are no comments above the function `_debitFrom()`.
- There are no comments above the function `_creditTo()`.

### XVSProxyOFTDest

- The comments above `dropFailedMessage()` do not include the emmited event or the access restriction.
- The comments above `circulatingSupply()` do not include the return value.
- There are no comments above the function `_debitFrom()`.
- There are no comments above the function `_creditTo()`.

### XVSBridgeAdmin

- The comments above `fallback()` do not include the return value.
- The comments above `setTrustedRemoteAddress()` do not include the access restriction or a summary of the functionality.
- The comments above `upsertSignature()` do not include the access restriction or emitted events.
- The comments above `transferBridgeOwnership()` do not include the potential error.
- The comments above `isTrustedRemote()` do not include the return value.
- The comments above `_getFunctionName()` do not include the input parameter and return value.
- There are no comments above the function `bytesToAddress()`.

### BaseXVSProxyOFT

- The comments above the `constructor()` do not include the emitted events.
- The comments above `removeTrustedRemote()` do not include the access restriction or emitted event.
- The comments above `token()` do not include the return value.
- There are no comments above the function `_isEligibleToSend()`.
- There are no comments above the function `_isEligibleToReceive()`.
- There are no comments above the function `_transferFrom()`.
- There are no comments above the function `_nonblockingLzReceive()`.
- There are no comments above the function `_ld2sdRate()`.

### TokenController

- The comments above the function `_isEligibleToMint()` do not include the emitted event or potential error.
- The comments above the function `_increaseMintLimit()` do not include the emitted event.
- The comments above the function `_ensureAllowed()` do not include the input parameter or the potential error.

## Recommendation

We recommend adding the missing or incomplete NatSpec comments mentioned above.

## Alleviation

`[CertiK, 12/19/2023]`: The client made changes resolving the finding in commits

- 7c63f14cb6ebae8f97ddbba719be6d196c59593f
- 60d4c5c88fe4f6a83b5cedb615fd20ed5f055301

# BRI-08 CONSIDER REVERTING AS OPPOSED TO HAVING EMPTY IMPLEMENTATON

| Category | Severity | Location | Status |
|---|---|---|---|
| Coding Style | Informational | BaseXVSProxyOFT.sol (base): 241~244; XVSBridgeAdmin.sol (base): 112~115 | Resolved |

## Description

The function `renounceOwnership()` is overriden to have an empty implementation to prevent ownership from accidentally being revoked. We recommend considering having the function revert with a descriptive error message indicating this function has been disabled for the contract. This will help avoid any potential confusion in the future if someone was to call this function.

## Recommendation

We recommend reverting as opposed to leaving an empty implementation.

## Alleviation

`[Venus, 12/14/2023]` : "Issue acknowledged. I won't make any changes for the current version. We don't consider it needed and we prefer to reduce the number of changes"

# BVP-01 | `minterToCap` MUST BE UPDATED CONSISTENTLY WITH ACCESS TO `mint()` AND `burn()` FUNCTIONS

| Category | Severity | Location | Status |
|---|---|---|---|
| Inconsistency | Informational | token/TokenController.sol (base): 115~116; token/XVS.sol (base): 29~30, 43~44 | Resolved |

## Description

- An account given a nonzero `minterToCap` value is assumed to be given access to functions `mint()` and `burn()`.
- An account given access to functions `mint()` or `burn()` is assumed to be given a nonzero `minterToCap` value.
- An account given access to the `burn()` function is assumed to be given access to mint, and vice versa.

While these privileges are dependent on one another, each part of the privilege is given independently, increasing the chance for inconsistent update.

## Recommendation

We recommend ensuring consistent update to all depending privileges whenever a new account is given this kind of access.

## Alleviation

`[Venus, 12/14/2023]`: "Issue acknowledged. I won't make any changes for the current version. All kinds of limits and caps will be initially set via multisig TX and via VIP's shortly. Therefore, there will be opportunities to review the proposed changes and avoid the inconsistencies described"

# GLOBAL-01 | DISCUSSION ON DESIGN

| Category | Severity | Location | Status |
|---|---|---|---|
| Inconsistency | Informational | | Resolved |

## Description

It is our understanding that the design of these contracts are for `XVS`, however, they also include functionality to check balances and potentially handle deflationary tokens.

We would like to ensure these contracts are only to be used for `XVS` and that no other deflationary or token that implements hooks should be considered.

## Recommendation

We recommend providing contextual information on the tokens to be used with the bridge.

## Alleviation

`[Venus, 12/14/2023]`: "Initially, only XVS will be used with these contracts. We don't expect to use them with fee-on-transfer or rebase tokens"

# XVS-03 | CASES NOT EXPLICITLY HANDLED

| Category | Severity | Location | Status |
|---|---|---|---|
| Inconsistency | Informational | XVSBridgeAdmin.sol (base): 79~85 | Resolved |

## Description

If `active_[i] && signature.length != 0` or `!active_[i] && signature.length == 0` then none of the if-blocks will be executed. If either of these combinations are input by accident, then it can cause confusion as there is nothing to indicate this other than a lack of event emitted.

## Recommendation

We recommend considering handling these cases explicitly by emitting a specific event to avoid any potential confusion.

## Alleviation

`[Venus, 12/14/2023]`: "Issue acknowledged. I won't make any changes for the current version. We prefer not to emit any events when there is no change in the function registry to avoid any extra gas. Moreover, it is an onlyOwner function that will be taken care in VIP or multisig tx. So, we'll have time to review the TX before submitting it, and therefore avoid the described scenario"

# OPTIMIZATIONS | VENUS - XVS TOKEN BRIDGE

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| BXV-01 | Repeated Check If Transaction Is Succesful | Inconsistency, Code Optimization | Optimization | Resolved |
| BXV-03 | Checking The `oracle_` Input Is Nonzero Can Be Done Sooner | Gas Optimization | Optimization | Resolved |
| TCB-04 | Redundant Getter Functions | Gas Optimization | Optimization | Resolved |
| TCB-05 | Unchecked Blocks Can Optimize Contract | Gas Optimization | Optimization | Resolved |
| XVS-01 | Duplicate Zero Address Checks | Logical Issue | Optimization | Resolved |
| XVS-02 | `for` Loop Optimization | Gas Optimization | Optimization | Resolved |

# BXV-01 | REPEATED CHECK IF TRANSACTION IS SUCCESFUL

| Category | Severity | Location | Status |
|---|---|---|---|
| Inconsistency, Code Optimization | Optimization | BaseXVSProxyOFT.sol (base): 352~357 | Resolved |

## Description

The function `lzReceive()` in https://github.com/LayerZero-Labs/solidity-examples/blob/main/contracts/lzApp/LzApp.sol makes the check

```
require(
            _srcAddress.length == trustedRemote.length && trustedRemote.length > 0
&& keccak256(_srcAddress) == keccak256(trustedRemote),
            "LzApp: invalid source sending contract"
        );
```

If this succeeds then it calls `_blockingLzReceive()`, which is overriden in https://github.com/LayerZero-Labs/solidity-examples/blob/main/contracts/lzApp/NonblockingLzApp.sol to call `nonblockingLzReceive()`, which then calls `_nonblockingLzReceive()`. However, this contract overrides this to then perform the same check again:

```
 require(
            _srcAddress.length == trustedRemote.length &&
                trustedRemote.length > 0 &&
                keccak256(_srcAddress) == keccak256(trustedRemote),
            "LzApp: invalid source sending contract"
        );
```

While this will ensure if there are any updates to trusted remotes between when a message fails and is retried, a check can be added when retrying the message to prevent making duplicate checks on successful messages.

## Recommendation

We recommend overriding and adding this check to `retryMessage()` from https://github.com/LayerZero-Labs/solidity-examples/blob/main/contracts/lzApp/NonblockingLzApp.sol.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commit 5293d3e5c1d33e0aa41500431e1f529adf612902.

# BXV-03 CHECKING THE `oracle_` INPUT IS NONZERO CAN BE DONE SOONER

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Optimization | BaseXVSProxyOFT.sol (base): 128~129 | Resolved |

## Description

In `BaseXVSProxyOFT.sol`, the check that the `oracle_` address input on deployment is nonzero is made after setting state variables `ld2sdRate` and `innerToken`. The check with `ensureNonzeroAddress` can be made at the beginning of the constructor with the other zero address checks.

## Recommendation

We recommend making the check that the `oracle_` address input is nonzero at the beginning of the constructor logic.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commit 7c2193bd7e7d9de4dff59982b98050844a9b3e51.

# TCB-04 | REDUNDANT GETTER FUNCTIONS

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Optimization | token/TokenController.sol (base): 24~25, 140~142 | Resolved |

## Description

Function `isBlacklisted()` is a view function for mapping `_blacklist`, however, this mapping is public.

## Recommendation

We recommend making the `_blacklist` mapping internal to avoid duplicate getter functions.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commit b017f579abd2c5709075f581fe1168e4ed9e3d2e.

# TCB-05 | UNCHECKED BLOCKS CAN OPTIMIZE CONTRACT

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Optimization | token/TokenController.sol (base): 162 | Resolved |

## Description

In the function `_isEligibleToMint()` the logic:

```
if (totalMintedNew > mintingCap) {
            revert MintLimitExceed();
        }
```

ensures that `mintingCap >= totalMintedNew`. Thus it is impossible for `mintingCap - totalMintedNew` to underflow so it can be placed in an unchecked block.

## Recommendation

We recommend using unchecked blocks on operations where underflow/overflow is not possible.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commit 2ddf4c02544c010ca8d669ae02c71d8ea976f76e.

# XVS-01 | DUPLICATE ZERO ADDRESS CHECKS

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Optimization | XVSBridgeAdmin.sol (base): 39~40, 96 | Resolved |

## Description

The function `initialize()` in the contract `XVSBridgeAdmin` checks that the input `accessControlManager_` is not the zero address via `ensureNonzeroAddress`. However, it also calls `__AccessControlled_init()`, which will call `_setAccessControlManager()` that has the following check

```
        require(address(accessControlManager_) != address(0), "invalid acess control
manager address");
```

Thus it will check that the input `accessControlManager_` is not the `address(0)` twice.

---

The function `transferBridgeOwnership()` in the contract `XVSBridgeAdmin` checks that the input `newOwner_` is not the zero address via `ensureNonzeroAddress`. However, it calls `XVSBridge.transferOwnership()`, which calls `transferOwnership()` in OpenZeppelin's `Ownable` contract that also checks the input is not the zero address.

## Recommendation

We recommend removing the duplicate checks.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commit 6785b84e085eaf455e35f66a6d04f79570279e5f.

# XVS-02 | `for` LOOP OPTIMIZATION

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Optimization | XVSBridgeAdmin.sol (base): 76 | Resolved |

## Description

In general, the counter in a for loop can be incremented or decremented in an unchecked block as it cannot overflow or underflow, saving gas as it will not perform a check for overflow or underflow.

Additionally, it saves a small amount of gas to increment an index in a `for` loop from the left instead of from the right side as it performs fewer operations.

## Recommendation

We recommend incrementing the index of the for loop in an unchecked block with a prefix increment.

## Alleviation

`[CertiK, 12/18/2023]`: The client made changes resolving the finding in commits

- 6bacbf7b9a265afeb99188bfe3c3d864ae06c145
- 4ef1a4c27fd824d8d4915f44a0c6c37d714a07c0

# APPENDIX | VENUS - XVS TOKEN BRIDGE

## Finding Categories

| Categories | Description |
|---|---|
| Gas Optimization | Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction. |
| Coding Style | Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable. |
| Concurrency | Concurrency findings are about issues that cause unexpected or unsafe interleaving of code executions. |
| 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. |
| Design Issue | Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories. |

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

# DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.