CERTIK

Preliminary Comments

Wasabix-yum

Jun 11th, 2021

Table of Contents

Summary

This report has been prepared for Wasabix-yum smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Wasabix-yum
Description	wasabix
Platform	BSC
Language	Solidity
Codebase	https://github.com/wasabi-swap-team/wasabix-yum/tree/main/contracts
Commit	31a023d325b799542940402891bc13dc20a7b864

Audit Summary

Delivery Date	Jun 11, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

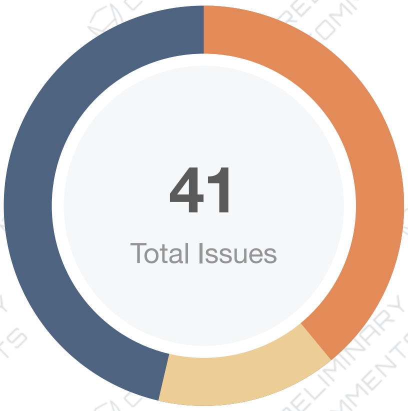
Vulnerability Summary

Total Issues	41
● Critical	0
● Major	16
● Medium	0
● Minor	6
● Informational	19
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
ALC	Alchemist.sol	903e8cf4665ff9d882d23806bad4484469e3c4d98151fdb0cb1f2dec37e24897
MSW	MultiSigWallet.sol	10e5e7781958a4cc734bae848ce89f551e33da513f7cfc6e8958691170295ba2
MST	MultiSigWalletWithTimelock.sol	f92db429228a4427660af4d61575f78f8795d6c89f5f9a3baa615bf84b39611a
SPM	StakingPools.sol	48a534676a5c8e5f536e42627767dd1ddae3fd52b3f7142d2b5691d10c0c3bce
TRA	Transmuter.sol	645beae9857bd11f006ea1901fa2e8e95c3f4ade7ebc47f0360eeadc0ee3e846
TDM	TransmuterD8.sol	824cdb1bb684b51076a6e2814e3f601dab9e29bceaccf8a9d588cc0673a1e30c
WIT	WIT.sol	9370d6e23936988a5e8fb1582ff0323eb46beecde8895a298a8636f6ad85ae57
WIZ	WIZT.sol	58e592e44512ee9b72af8e6ad3683a1f5d11a20a21d3847ab5cd308dde8c68a3
WVV	WVVT.sol	b4e4f14c963c1a3a66ef1a0bdf0398a7c305d17a3c1bd2d70f2632e4484d3451
WBT	WaBtcToken.sol	8344d95deb77f954f1a93059263fc7a69e6dad16d565c84ba47ba1dc45f8f854
WTM	WaToken.sol	25687b49a4263218a4a5c8b01eb39fd5daf5f8c57d5044a4a954ab45864eda56
WTA	WasabiToken.sol	1c1349ae15e35c9455ddda55a4b724a4406bb3c8531f209e6c4f845903ee0213
YVA	adapters/YearnVaultAdapter.sol	3584b2d84e1d984a59e06f60d671d7d2f1deffba79724105bfc19cf436b446aa

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**16** (39.02%)
🟨	**Medium**	**0** (0.00%)
🟨	**Minor**	**6** (14.63%)
🟦	**Informational**	**19** (46.34%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
ALC-01	Single Source of Price Oracle	**Centralization / Privilege**	● **Informational**	ⓘ **Pending**
ALC-02	Missing Emit Event	Coding Style	● Informational	ⓘ Pending
ALC-03	Recommended Explicit Vault Validity Checks	Logical Issue	● Informational	ⓘ Pending
ALC-04	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
IVA-01	Lack of Input Validation	Volatile Code	● Informational	ⓘ Pending
IVA-02	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
MSW-01	Lack of Input Validation	Logical Issue	● Minor	ⓘ Pending
PVA-01	Lack of Input Validation	Volatile Code	● Informational	ⓘ Pending
PVA-02	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
SPM-01	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
TDM-01	No Log In `require()` Check	Coding Style	● Informational	ⓘ Pending
TDM-02	TBD: Emit event	Volatile Code	● Minor	ⓘ Pending
TDM-03	Minimize The Scope of Access To The Function	Control Flow	● Minor	ⓘ Pending

ID	Title	Category	Severity	Status
TDM-04	Missing Emit Event	Coding Style	● Informational	ⓘ Pending
TRA-01	No Log In `require()` Check	Coding Style	● Informational	ⓘ Pending
TRA-02	Minimize The Scope of Access To The Function	Control Flow	● Minor	ⓘ Pending
VVA-01	Lack of Input Validation	Volatile Code	● Informational	ⓘ Pending
VVA-02	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
WBT-01	Inaccurate Comment	Inconsistency	● Minor	ⓘ Pending
WBT-02	Centralized Risk to Sensitive Functions	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
WIT-01	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
WIZ-01	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
WTA-01	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
WTM-01	Inaccurate Comment	Inconsistency	● Minor	ⓘ Pending
WTM-02	Centralized Risk to Sensitive Functions	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
WVV-01	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
YIV-01	Single Source of Price Oracle	**Centralization / Privilege**	● **Informational**	ⓘ **Pending**
YIV-02	Missing Emit Event	Coding Style	● Informational	ⓘ Pending
YIV-03	Recommended Explicit Vault Validity Checks	Logical Issue	● Informational	ⓘ Pending
YIV-04	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
YPV-01	Single Source of Price Oracle	**Centralization / Privilege**	● **Informational**	ⓘ **Pending**

ID	Title	Category	Severity	Status
YPV-02	Missing Emit Event	Coding Style	● Informational	ⓘ Pending
YPV-03	Recommended Explicit Vault Validity Checks	Logical Issue	● Informational	ⓘ Pending
YPV-04	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
YPV-05	Safemath library is not used	Logical Issue	● Major	ⓘ Pending
YVA-01	Lack of Input Validation	Volatile Code	● Informational	ⓘ Pending
YVA-02	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**
YVV-01	Single Source of Price Oracle	**Centralization / Privilege**	● **Informational**	ⓘ **Pending**
YVV-02	Missing Emit Event	Coding Style	● Informational	ⓘ Pending
YVV-03	Recommended Explicit Vault Validity Checks	Logical Issue	● Informational	ⓘ Pending
YVV-04	Centralized Risk	**Centralization / Privilege**	● **Major**	ⓘ **Pending**

ALC-01 | Single Source of Price Oracle

Category	Severity	Location	Status
Centralization / Privilege	● **Informational**	Alchemist.sol: 676	ⓘ **Pending**

Description

Chainlink is the only price oracle that provides the price. If the single price oracle provides an incorrect price, this error will dominate the price and cause single point of failure by affecting the token price.

Recommendation

In order to prevent the single point of failure issue and protect from the fluctuation of the price caused by price oracle, we advise the client to adopt multiple price oracles as token price references.

ALC-02 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	● Informational	Alchemist.sol: 279, 332	ⓘ Pending

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `setOracleAddress()`
- `setFlushActivator()`

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

ALC-03 | Recommended Explicit Vault Validity Checks

Category	Severity	Location	Status
Logical Issue	● Informational	Alchemist.sol: 716	ⓘ Pending

Description

There's no sanity check to validate if a vault is existing. If the same vault at address `_adapter` were added multiple times, the total amount of `totalDeposited` of a specific token will be mistakenly calculated.

Recommendation

We advise the client to detect whether the given vault for addition is a duplicate of an existing vault. The vault addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

ALC-04 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	Alchemist.sol: 706	ⓘ Pending

Description

The owner of the account with the `governance` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example,

- User who is granted a `governance` role can update the address of chainlink price oracle and minimum value for Peggy, to update the price of the token.
- `governance` user can set the threshold `flushActivator` to indirectly decide when to invoke the vaults flushing functionality in functions like `deposit()` and `withdraw()`

Hackers who compromise the account with a `governance` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

IVA-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	adapters/IdleVaultAdapter.sol: 45	ⓘ Pending

Description

The assigned values to `vault` and `admin` in the constructor of adapters should be verified as a non-zero value to prevent error.

Recommendation

Check that the passed-in values are non-zero values. Example:

```
require(address(_vault) != address(0), "_vault address is a zero address");
require(_admin != address(0), "_admin is a zero address");
```

IVA-02 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	adapters/IdleVaultAdapter.sol: 122	ⓘ **Pending**

Description

The owner of the account `owner` can withdraw an arbitrary amount of token from vault to an arbitrary address `_recipient` by calling function `withdraw()`

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

MSW-01 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	● Minor	MultiSigWallet.sol: 157	ⓘ Pending

Description

The value of `newOwner` argument is not validated as non-zero value. An invalid owner address will prevent any fund withdraw from its wallet.

Recommendation

We advise the client to add a argument validator to check if the value of `newOwner` is set as address(0)

PVA-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	adapters/PickleVaultAdapter.sol: 56	ⓘ Pending

Description

The assigned values to `vault` and `admin` in the constructor of adapters should be verified as a non-zero value to prevent error.

Recommendation

Check that the passed-in values are non-zero values. Example:

```
require(address(_vault) != address(0), "_vault address is a zero address");
require(_admin != address(0), "_admin is a zero address");
```

PVA-02 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	adapters/PickleVaultAdapter.sol: 135	ⓘ **Pending**

Description

The owner of the account `owner` can withdraw an arbitrary amount of token from vault to an arbitrary address `_recipient` by calling function `withdraw()`

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

SPM-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	StakingPools.sol: 117	ⓘ **Pending**

Description

The owner of the account with the `governance` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example,

- User who is granted a `governance` role can update the address of chainlink price oracle and minimum value for Peggy, to update the price of the token.
- `governance` user can set the threshold `flushActivator` to indirectly decide when to invoke the vaults flushing functionality in functions like `deposit()` and `withdraw()`

Hackers who compromise the account with a `governance` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

TDM-01 | No Log In `require()` Check

Category	Severity	Location	Status
Coding Style	● Informational	TransmuterD8.sol: 190	ⓘ Pending

Description

No log message is added in the `require()` check. Log is essential message for debugging purpose and tracking the transaction. Adding log to `require()` can also increase the readability and overall quality of the codebase.

Recommendation

We advise the client to add log message to the `require()` check with similar snippet as following:

```
require(realisedTokens[sender] > 0, "no realisedToken balance for sender");
```

TDM-02 | TBD: Emit event

Category	Severity	Location	Status
Volatile Code	● Minor	TransmuterD8.sol	ⓘ Pending

Description

Emit event

Recommendation

Emit event

TDM-03 | Minimize The Scope of Access To The Function

Category	Severity	Location	Status
Control Flow	● Minor	TransmuterD8.sol: 351	ⓘ Pending

Description

As the comment indicates in LXX `This function is meant to be called by the Alchemist contract for when it is sending the yield to the transmuter.` , the function `distribute()` should only be called by `Alchemist` contract. However, currently a whitelist is adopted to restrict the accesses to the `distribute()` function, which may have the potential to add non-Alchemist address into it.

Recommendation

We advise the client to stored the `Alchemist` contract addresses in immutable variables and initialized them in the constructor of e.g the `TransmuterD8` contract.

TDM-04 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	● Informational	TransmuterD8.sol: 188, 212, 230, 201, 270, 351, 478	ⓘ Pending

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers.

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

TRA-01 | No Log In `require()` Check

Category	Severity	Location	Status
Coding Style	● Informational	Transmuter.sol: 203	ⓘ Pending

Description

No log message is added in the `require()` check. Log is essential message for debugging purpose and tracking the transaction. Adding log to `require()` can also increase the readability and overall quality of the codebase.

Recommendation

We advise the client to add log message to the `require()` check with similar snippet as following:

```
require(realisedTokens[sender] > 0, "no realisedToken balance for sender");
```

TRA-02 | Minimize The Scope of Access To The Function

Category	Severity	Location	Status
Control Flow	● Minor	Transmuter.sol: 364	ⓘ Pending

Description

As the comment indicates in LXX `This function is meant to be called by the Alchemist contract for when it is sending the yield to the transmuter.` , the function `distribute()` should only be called by `Alchemist` contract. However, currently a whitelist is adopted to restrict the accesses to the `distribute()` function, which may have the potential to add non-Alchemist address into it.

Recommendation

We advise the client to stored the `Alchemist` contract addresses in immutable variables and initialized them in the constructor of e.g the `TransmuterD8` contract.

VVA-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	adapters/VesperVaultAdapter.sol: 43	ⓘ Pending

Description

The assigned values to `vault` and `admin` in the constructor of adapters should be verified as a non-zero value to prevent error.

Recommendation

Check that the passed-in values are non-zero values. Example:

```
require(address(_vault) != address(0), "_vault address is a zero address");
require(_admin != address(0), "_admin is a zero address");
```

VVA-02 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	adapters/VesperVaultAdapter.sol: 115	ⓘ Pending

Description

The owner of the account `owner` can withdraw an arbitrary amount of token from vault to an arbitrary address `_recipient` by calling function `withdraw()`

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

WBT-01 | Inaccurate Comment

Category	Severity	Location	Status
Inconsistency	● Minor	WaBtcToken.sol: 64, 68	ⓘ Pending

Description

The comment in mentioned lines shows that only the caller that has the minter role can call the function `mint()`, which is not accurate as there's no `Minter_Role` in the contract `AlToken`.

Recommendation

We advise the client to add `Minter_Role` and corresponding modifier to restrict the access to the function `mint()`.

WBT-02 | Centralized Risk to Sensitive Functions

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	WaBtcToken.sol: 81, 88, 106	ⓘ **Pending**

Description

The owner of the account `owner` can update the ceiling of a token that is allowed to mint, add the account to which the minted token can be transferred, and grant `SENTINEL_ROLE` to any address in the contract `WaToken()` and `WaBtcToken()`.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

WIT-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	WIT.sol: 45	ⓘ **Pending**

Description

The owner of the account that is assigned as `MINTER_ROLE` can mint an arbitrary amount of token to an arbitrary address by calling function `mint()`

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

WIZ-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	WIZT.sol: 45	ⓘ **Pending**

Description

The owner of the account that is assigned as `MINTER_ROLE` can mint an arbitrary amount of token to an arbitrary address by calling function `mint()`

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

WTA-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	WasabiToken.sol: 45	Pending

Description

The owner of the account that is assigned as `MINTER_ROLE` can mint an arbitrary amount of token to an arbitrary address by calling function `mint()`

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

WTM-01 | Inaccurate Comment

Category	Severity	Location	Status
Inconsistency	● Minor	WaToken.sol: 60, 64	ⓘ Pending

Description

The comment in mentioned lines shows that only the caller that has the minter role can call the function `mint()`, which is not accurate as there's no `Minter_Role` in the contract `AlToken`.

Recommendation

We advise the client to add `Minter_Role` and corresponding modifier to restrict the access to the function `mint()`.

WTM-02 | Centralized Risk to Sensitive Functions

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	WaToken.sol: 77, 84, 102	ⓘ **Pending**

Description

The owner of the account `owner` can update the ceiling of a token that is allowed to mint, add the account to which the minted token can be transferred, and grant `SENTINEL_ROLE` to any address in the contract `WaToken()` and `WaBtcToken()`.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

WVV-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	WVVT.sol: 45	ⓘ **Pending**

Description

The owner of the account that is assigned as `MINTER_ROLE` can mint an arbitrary amount of token to an arbitrary address by calling function `mint()`

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

YIV-01 | Single Source of Price Oracle

Category	Severity	Location	Status
Centralization / Privilege	● **Informational**	YumIdleVault.sol: 657	ⓘ **Pending**

Description

Chainlink is the only price oracle that provides the price. If the single price oracle provides an incorrect price, this error will dominate the price and cause single point of failure by affecting the token price.

Recommendation

In order to prevent the single point of failure issue and protect from the fluctuation of the price caused by price oracle, we advise the client to adopt multiple price oracles as token price references.

YIV-02 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	● Informational	YumIdleVault.sol: 318, 265	ⓘ Pending

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `setOracleAddress()`
- `setFlushActivator()`

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

YIV-03 | Recommended Explicit Vault Validity Checks

Category	Severity	Location	Status
Logical Issue	● Informational	YumIdleVault.sol: 697	ⓘ Pending

Description

There's no sanity check to validate if a vault is existing. If the same vault at address `_adapter` were added multiple times, the total amount of `totalDeposited` of a specific token will be mistakenly calculated.

Recommendation

We advise the client to detect whether the given vault for addition is a duplicate of an existing vault. The vault addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

YIV-04 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	YumIdleVault.sol: 687	ⓘ **Pending**

Description

The owner of the account with the `governance` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example,

- User who is granted a `governance` role can update the address of chainlink price oracle and minimum value for Peggy, to update the price of the token.
- `governance` user can set the threshold `flushActivator` to indirectly decide when to invoke the vaults flushing functionality in functions like `deposit()` and `withdraw()`

Hackers who compromise the account with a `governance` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

YPV-01 | Single Source of Price Oracle

Category	Severity	Location	Status
Centralization / Privilege	● **Informational**	YumPickleVault.sol: 659	ⓘ **Pending**

Description

Chainlink is the only price oracle that provides the price. If the single price oracle provides an incorrect price, this error will dominate the price and cause single point of failure by affecting the token price.

Recommendation

In order to prevent the single point of failure issue and protect from the fluctuation of the price caused by price oracle, we advise the client to adopt multiple price oracles as token price references.

YPV-02 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	● Informational	YumPickleVault.sol: 266, 319	ⓘ Pending

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `setOracleAddress()`
- `setFlushActivator()`

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

YPV-03 | Recommended Explicit Vault Validity Checks

Category	Severity	Location	Status
Logical Issue	● Informational	YumPickleVault.sol: 699	ⓘ Pending

Description

There's no sanity check to validate if a vault is existing. If the same vault at address `_adapter` were added multiple times, the total amount of `totalDeposited` of a specific token will be mistakenly calculated.

Recommendation

We advise the client to detect whether the given vault for addition is a duplicate of an existing vault. The vault addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

YPV-04 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	YumPickleVault.sol: 689	ⓘ **Pending**

Description

The owner of the account with the `governance` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example,

- User who is granted a `governance` role can update the address of chainlink price oracle and minimum value for Peggy, to update the price of the token.
- `governance` user can set the threshold `flushActivator` to indirectly decide when to invoke the vaults flushing functionality in functions like `deposit()` and `withdraw()`

Hackers who compromise the account with a `governance` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

YPV-05 | Safemath library is not used

Category	Severity	Location	Status
Logical Issue	● Major	YumPickleVault.sol: 423	ⓘ Pending

Description

Safemath library is not used in line 423.

```
return _recallFunds(_vaultId, _vault.totalDeposited * (1000-slippage) / 1000);
```

Recommendation

We strongly recommend to use safemath library for any calculation.

YVA-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	adapters/YearnVaultAdapter.sol: 33~34	ⓘ Pending

Description

The assigned values to `vault` and `admin` in the constructor of adapters should be verified as a non-zero value to prevent error.

Recommendation

Check that the passed-in values are non-zero values. Example:

```
require(address(_vault) != address(0), "_vault address is a zero address");
require(_admin != address(0), "_admin is a zero address");
```

YVA-02 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	adapters/YearnVaultAdapter.sol: 72	ⓘ **Pending**

Description

The owner of the account `owner` can withdraw an arbitrary amount of token from vault to an arbitrary address `_recipient` by calling function `withdraw()`

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

YVV-01 | Single Source of Price Oracle

Category	Severity	Location	Status
Centralization / Privilege	● **Informational**	YumVesperVaultD8.sol: 649	ⓘ **Pending**

Description

Chainlink is the only price oracle that provides the price. If the single price oracle provides an incorrect price, this error will dominate the price and cause single point of failure by affecting the token price.

Recommendation

In order to prevent the single point of failure issue and protect from the fluctuation of the price caused by price oracle, we advise the client to adopt multiple price oracles as token price references.

YVV-02 | Missing Emit Event

Category	Severity	Location	Status
Coding Style	● Informational	YumVesperVaultD8.sol: 312, 259	ⓘ Pending

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `setOracleAddress()`
- `setFlushActivator()`

Recommendation

We advise the client to consider adding events for sensitive actions and emit them in the corresponding functions.

YVV-03 | Recommended Explicit Vault Validity Checks

Category	Severity	Location	Status
Logical Issue	● Informational	YumVesperVaultD8.sol: 689	ⓘ Pending

Description

There's no sanity check to validate if a vault is existing. If the same vault at address `_adapter` were added multiple times, the total amount of `totalDeposited` of a specific token will be mistakenly calculated.

Recommendation

We advise the client to detect whether the given vault for addition is a duplicate of an existing vault. The vault addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

YVV-04 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	YumVesperVaultD8.sol: 679	ⓘ Pending

Description

The owner of the account with the `governance` role has the privilege to update the sensitive variables and conduct sensitive operations in the project. For example,

- User who is granted a `governance` role can update the address of chainlink price oracle and minimum value for Peggy, to update the price of the token.
- `governance` user can set the threshold `flushActivator` to indirectly decide when to invoke the vaults flushing functionality in functions like `deposit()` and `withdraw()`

Hackers who compromise the account with a `governance` role may take advantage of these centralized privileges and manipulate the project for profits.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

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

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

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to 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 without CertiK's prior written consent.

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.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK