scolabs

Preliminary Comments

PINTU TOKEN

Scolabs Assessed
on Apr 18th, 2024

Daftar Isi

SUMMARY

REVIEW NOTES

FINDINGS

OPTIMIZATION

SUMMARY 1

scolabs | Scolabs Assessed on Apr 18th, 2024

Pintu Token

These preliminary comments were prepared by Scolabs, the leader in Web3.0 security.

Executive Summary

TYPES
Fundraising

ECOSYSTEM
Ethereum Blockchain (ERC20)

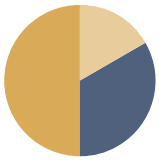
METHODS
Formal Verification, Manual Review, Static Analysis, Solution and Recomendation

LANGUAGE
Solidity

TIMELINE
Delivered on 04/18/2024

CODEBASE
https://vscode.blockscan.com/ethereum/0xC229c69eB3BB51828D-0cAA3509A05a51083898dd

Vulnerability Summary



18	0	0	0	0	0	18
Total Finding	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Pending

---

0 Critical

0 High

High severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.

3 Medium — 3 Pending

Medium severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.

6 Low — 6 Pending

Low severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.

9 NC — 9 Pending

No Compliance issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.

Codebase

SOURCE / REPOSITORY

https://vscode.blockscan.com/ethereum/0xC229c69eB3BB51828D-0cAA3509A05a51083898dd

COMMIT

base: -

Audit Scope

FILES DETAILS

Filepath
src/PintuTokenProxy/PintuTokenProxy.sol
src/PintuTokenV1/PintuTokenV1.sol

Approach & Method

This report has been prepared for Pintu Token to discover issues and vulnerabilities in the source code of the Pintu Token

- New Deal project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Formal Verification, Manual Review, Static Analysis, Recommendation and Solution techniques.

The auditing process pays special attention to the following considerations:

- 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.
- Testing the smart contracts against both common and uncommon attack vectors.
- Get Recommendation and Solution for vulnerabilities.

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.

Overview

Pintu Token (PTU) is an ERC-20 token that is designed to drive and support the long term growth and vision of the Pintu ecosystem and community. The launch of PTU is aimed at rewarding loyal users of the platform and incentivising new members to join Pintu's expanding ecosystem.

FINDINGS 3

This report has been prepared to discover issues and vulnerabilities for PINTU Token. Through this audit, we have uncovered 20 issues ranging from different severity levels. Utilizing the techniques of Formal Verification, Manual/ Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings :

ID	TITLE	SEVERITY	INSTANCES	STATUS
PTU 01	Constant functions using assembly code	Medium	3	Pending
PTU 02	Centralization Risk for trusted owners	Medium	6	Pending
PTU 03	Library function isn't 'internal' or 'private'	Medium	27	Pending
PTU 04	Local variable shadowing	Low	8	Pending
PTU 05	Incorrect modifier	Low	1	Pending
PTU 06	Do not leave an implementation contract uninitialized	Low	3	Pending
PTU 07	Initializers could be front-run	Low	3	Pending
PTU 08	Use 'Ownable2Step. transferOwnership' instead of 'Ownable. transferOwnership'	Low	1	Pending

PTU 09	Use ‘initializer’ for public-facing functions only	Low	1	Pending
PTU 10	Assembly usage	NC	6	Pending
PTU 11	Dead code	NC	6	Pending
PTU 12	Incorrect versions of Solidity	NC	2	Pending
PTU 13	Low-level calls	NC	1	Pending
PTU 14	Redundant Statements	NC	1	Pending
PTU 15	Public function that could be declared external	NC	1	Pending
PTU 16	Missing checks for ‘address(0)’ when assigning values to address state variables	NC	1	Pending
PTU 17	Event missing indexed field	NC	2	Pending
PTU 18	Function ordering does not follow the Solidity style guide	NC	1	Pending

PTU 01 Constant Functions Using Assembly Code

Pintu Token (PTU) is an ERC-20 token that is designed to drive and support the long term growth and vision of the Pintu ecosystem and community. The launch of PTU is aimed at rewarding loyal users of the platform and incentivising new members to join Pintu's expanding ecosystem.

CATEGORY	SEVERITY	LOCATION	STATUS
Volatile Code	Medium	PintuTokenProxy.sol: 94-100, 143-148, 287-292	Pending

Description:

Functions declared as constant/pure/view using assembly code.

Constant/pure/view was not enforced prior to Solidity 0.5. Starting from Solidity 0.5, a call to a constant/pure/view function uses the STATICCALL opcode, which reverts in case of state modification. As a result, a call to an incorrectly labeled function may trap a contract compiled with Solidity 0.5.

Source: PintuTokenProxy.sol

```
94    function isContract(address addr) internal view returns (bool) {
95        uint256 size;
96        assembly {
97            size := extcodesize(addr)
98        }
99        return size > 0;
100   }
```

```
143   function _implementation() internal view returns (address impl) {
144       bytes32 slot = IMPLEMENTATION_SLOT;
145       assembly {
146           impl := sload(slot)
147       }
148   }
```

```
287   function _admin() internal view returns (address adm) {
288       bytes32 slot = ADMIN_SLOT;
289       assembly {
290           adm := sload(slot)
291       }
292   }
```

Recommendation:

Ensure the attributes of contracts compiled prior to Solidity 0.5.0 are correct.

PTU 02 Centralization Risk for Trusted Owners

CATEGORY	SEVERITY	LOCATION	STATUS
Centralization	Medium	PintuTokenV1.sol: 379-381, 387-393, 411-464, 452-455, 460-463, 787-793	Pending

Description:

Contracts have owners with privileged rights to perform admin tasks and need to be trusted to not perform malicious updates or drain funds.

Source: PintuTokenV1.sol

```
379     function renounceOwnership() public onlyOwner {
380         _setOwner(address(0));
381     }
```

```
387     function transferOwnership(address newOwner) public onlyOwner {
388         require(
389             newOwner != address(0),
390             "Ownable: new owner is the zero address"
391         );
392         _setOwner(newOwner);
393     }
```

```
411 > contract Pausable is Ownable {…
464 }
```

```
452     function pause() public onlyOwner whenNotPaused {
453         _paused = true;
454         emit Paused(_msgSender());
455     }
```

```
460     function unpause() public onlyOwner whenPaused {
461         _paused = false;
462         emit Unpaused(_msgSender());
463     }
```

```
787     function mint(
788         address account,
789         uint256 amount
790     ) public whenNotPaused onlyOwner returns (bool) {
791         _mint(account, amount);
792         return true;
793     }
```

Recommendation:
Implement decentralized governance mechanisms that allow stakeholders to participate in decision-making processes related to the smart contract, such as protocol upgrades, parameter adjustments, or fund management.

PTU 03 Library Function Isn't 'Internal' Or 'Private'

CATEGORY	SEVERITY	LOCATION	STATUS
Volatile Code	Medium	PintuTokenV1.sol: 166, 171, 180, 189, 205, 216, 328, 347, 355, 394, 417, 425, 467, 474, 486, 494, 510, 533, 555, 574, 690, 697, 710, 723, 740, 748, 764	Pending

Description:

Marking library functions as internal or private provides encapsulation and prevents unintended access from external contracts, enhancing security and readability of your code.

Source: PintuTokenV1.sol

```
166:     function totalSupply() external view returns (uint256);

171:     function balanceOf(address account) external view returns (uint256);

180:     function transfer(

189:     function allowance(

205:     function approve(address spender, uint256 amount) external returns (bool);

216:     function transferFrom(

328:     function owner() public view returns (address) {

347:     function renounceOwnership() public onlyOwner {

355:     function transferOwnership(address newOwner) public onlyOwner {

394:     function paused() public view returns (bool) {

417:     function pause() public onlyOwner whenNotPaused {

425:     function unpause() public onlyOwner whenPaused {

467:     function totalSupply() public view returns (uint256) {

474:     function balanceOf(address account) public view returns (uint256) {
```

```
486:     function transfer(

494:     function allowance(

510:     function approve(

533:     function transferFrom(

555:     function increaseAllowance(

574:     function decreaseAllowance(

690:     function burn(uint256 amount) public whenNotPaused {

697:     function burnFrom(address account, uint256 amount) public whenNotPaused {

710:     function mint(

723:     function initialize(

740:     function name() public view returns (string memory) {

748:     function symbol() public view returns (string memory) {

764:     function decimals() public view returns (uint8) {
```

Recommendation:
Mark it as internal or private. This ensures that the function cannot be called externally and is only accessible within the library or contract.

PTU 04 Local Variable Shadowing

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	Low	PintuTokenProxy.sol: 130, 222, 326 PintuTokenV1.sol: 533, 729, 804, 805, 806	Pending

Description:

Local variable shadowing can have significant implications due to the deterministic nature of blockchain execution and the potential for unexpected behavior.

Source: PintuTokenProxy.sol

```
21    /**
22     * @return The Address of the implementation.
23     */
24    function _implementation() internal view returns (address);
```

```
143   function _implementation() internal view returns (address impl) {
144       bytes32 slot = IMPLEMENTATION_SLOT;
145       assembly {
146           impl := sload(slot)
147       }
148   }
```

```
130   constructor(address _implementation) public {
131       assert(
132           IMPLEMENTATION_SLOT ==
133               keccak256("org.zeppelinos.proxy.implementation")
134       );
135
136       _setImplementation(_implementation);
137   }
```

```
221   constructor(
222       address _implementation
223   ) public UpgradeabilityProxy(_implementation) {
224       assert(ADMIN_SLOT == keccak256("org.zeppelinos.proxy.admin"));
225
226       _setAdmin(msg.sender);
227   }
```

```
325     constructor(
326         address _implementation
327     ) public AdminUpgradeabilityProxy(_implementation) {}
```

Source: PintuTokenV1.sol

```
360     function owner() public view returns (address) {
361         return _owner;
362     }
```

```
532     function allowance(
533         address owner,
534         address spender
535     ) public view returns (uint256) {
536         return _allowances[owner][spender];
537     }
```

```
729 function _approve(address owner, address spender, uint256 amount) internal {
730     require(owner != address(0), "ERC20: approve from the zero address");
731     require(spender != address(0), "ERC20: approve to the zero address");
732
733     _allowances[owner][spender] = amount;
734     emit Approval(owner, spender, amount);
735 }
```

```
796 contract PintuTokenV1 is ERC20Burnable, ERC20Mintable {
797     string private _name;
798     string private _symbol;
799     uint8 private _decimals;
800
801     constructor() public {}
802
803     function initialize(
804         string memory name,
805         string memory symbol,
806         uint8 decimals,
807         uint256 initialSupply
808     ) public initializer {
809         Ownable.initialize(_msgSender());
810         _name = name;
811         _symbol = symbol;
812         _decimals = decimals;
813
814         _mint(_msgSender(), initialSupply);
815     }
```

```
820     function name() public view returns (string memory) {
821         return _name;
822     }
```

```
828     function symbol() public view returns (string memory) {
829         return _symbol;
830     }
```

```
844     function decimals() public view returns (uint8) {
845         return _decimals;
846     }
```

Recommendation:
Rename the local variables that shadow another component.

PTU 05 Incorrect Modifier

CATEGORY	SEVERITY	LOCATION	STATUS
Coding Style	Low	PintuTokenProxy.sol: 208-214	Pending

Description:

If a modifier does not execute _ or revert, the execution of the function will return the default value, which can be misleading for the caller.

Source: PintuTokenProxy.sol

```
208     modifier ifAdmin() {
209         if (msg.sender == _admin()) {
210             _;
211         } else {
212             _fallback();
213         }
214     }
```

Recommendation:
All the paths in a modifier must execute or revert.

PTU 06 Do Not Leave An Implementation Contract Uninitialized

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	Medium	PintuTokenV1.sol: 246, 389, 803	Pending

Description:

An uninitialized implementation contract can be taken over by an attacker, which may impact the proxy.

Source: PintuTokenV1.sol

```
246:     constructor() internal {}

389:     constructor() internal {}

803:     constructor() public {}
```

Recommendation:

To prevent the implementation contract from being used, it's advisable to invoke the '_disableInitializers' function in the constructor to automatically lock it when it is deployed. This should look similar to this:

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
    _disableInitializers();
}
```

PTU 07 Initializers Could Be Front-Run

CATEGORY	SEVERITY	LOCATION	STATUS
Volatile Code	Low	PintuTokenV1.sol: 246, 389, 803	Pending

Description:

An uninitialized implementation contract can be taken over by an attacker, which may impact the proxy.

Source: PintuTokenV1.sol

```
284:     modifier initializer() {

321:     function initialize(address sender) internal initializer {

723:     function initialize(
```

Recommendation:

One way to mitigate this risk is by using timelocks or delay mechanisms. These mechanisms introduce a delay between the deployment of the contract and the execution of critical functions, including initializers. During this delay period, users have the opportunity to review the contract and ensure that it behaves as expected before any critical functions are executed.

PTU 08 Use 'Ownable2Step.transferOwnership' Instead of 'Ownable.transferOwnership'

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	Medium	PintuTokenV1.sol: 355	Pending

Description:

This extension of the {Ownable} contract includes a two-step mechanism to transfer ownership, where the new owner must call {acceptOwnership} in order to replace the old one. This can help prevent common mistakes, such as transfers of ownership to incorrect accounts.

Source: PintuTokenV1.sol

```
355     function transferOwnership(address newOwner) public onlyOwner {
356         require(newOwner != address(0), "Ownable: new owner is the zero address");
357         _setOwner(newOwner);
358     }
```

Recommendation:

Use (Ownable2Step.transferOwnership) which is safer. Use it as it is more secure due to 2-stage ownership transfer

PTU 09 Use 'Initializer' for Public-Facing Functions Only

CATEGORY	SEVERITY	LOCATION	STATUS
Volatile Code	Low	PintuTokenV1.sol: 321	Pending

Description:

See the difference between onlyInitializing and initializer.

Source: PintuTokenV1.sol

```
321     function initialize(address sender) internal initializer {
322         _setOwner(sender);
323     }
```

Recommendation:
Use 'initializer' for public-facing functions only. Replace with 'onlyInitializing' on internal functions.

PTU 10 Assembly Usage

CATEGORY	SEVERITY	LOCATION	STATUS
Coding Style	NC	PintuTokenProxy.sol: 33-61, 96-99, 145-147, 171-173, 289-291, 301-303	Pending

Description:

Using assembly in this scenario allows for more efficient gas usage compared to using higher-level Solidity code for the same purpose. However, it's important to use assembly judiciously and only when necessary, as it can introduce complexities and potential security risks.

Source: PintuTokenProxy.sol

```
33          assembly {
34              // Copy msg.data. We take full control of memory in this inline assembly
35              // block because it will not return to Solidity code. We overwrite the
36              // Solidity scratch pad at memory position 0.
37              calldatacopy(0, 0, calldatasize)
38
39              // Call the implementation.
40              // out and outsize are 0 because we don't know the size yet.
41              let result := delegatecall(
42                  gas,
43                  implementation,
44                  0,
45                  calldatasize,
46                  0,
47                  0
48              )
49
50              // Copy the returned data.
51              returndatacopy(0, 0, returndatasize)
52
53              switch result
54              // delegatecall returns 0 on error.
55              case 0 {
56                  revert(0, returndatasize)
57              }
58              default {
59                  return(0, returndatasize)
60              }
61          }
```

```
96          assembly {
97              size := extcodesize(addr)
98          }
99          return size > 0;
100     }
```

```
145         assembly {
146             impl := sload(slot)
147         }
```

```
171         assembly {
172             sstore(slot, newImplementation)
173         }
```

```
289         assembly {
290             adm := sload(slot)
291         }
```

```
301         assembly {
302             sstore(slot, newAdmin)
303         }
```

Recommendation:
Instead of resorting to Assembly, it's generally recommended to leverage the built-in features and constructs of Solidity whenever possible

PTU 11 Dead Code

CATEGORY	SEVERITY	LOCATION	STATUS
Gas Optimization	NC	PintuTokenProxy.sol: 278-281, 102-104, 119-130, 143-145, 160-167, 77-89	Pending

Description:

Function is never called from outside the contract, so it is considered dead code. It doesn't affect the functionality of the contract, but it adds unnecessary complexity and potentially increases the gas cost of deploying and executing the contract.

Source: PintuTokenProxy.sol

```
278     function _msgData() internal view returns (bytes memory) {
279         this; // silence state mutability warning without generating
280         return msg.data;
281     }
```

```
102     function div(uint256 a, uint256 b) internal pure returns (uint256) {
103         return div(a, b, "SafeMath: division by zero");
104     }
```

```
119     function div(
120         uint256 a,
121         uint256 b,
122         string memory errorMessage
123     ) internal pure returns (uint256) {
124         // Solidity only automatically asserts when dividing by 0
125         require(b > 0, errorMessage);
126         uint256 c = a / b;
127         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
128
129         return c;
130     }
```

```
143     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
144         return mod(a, b, "SafeMath: modulo by zero");
145     }
```

```
160     function mod(
161         uint256 a,
162         uint256 b,
163         string memory errorMessage
164     ) internal pure returns (uint256) {
165         require(b != 0, errorMessage);
166         return a % b;
167     }
```

```
77  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
78      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
79      // benefit is lost if 'b' is also tested.
80      // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
81      if (a == 0) {
82          return 0;
83      }
84
85      uint256 c = a * b;
86      require(c / a == b, "SafeMath: multiplication overflow");
87
88      return c;
89  }
```

Recommendation:
Instead of resorting to Assembly, it's generally recommended to leverage the built-in features and constructs of Solidity whenever possible.

PTU 12 Incorrect Versions of Solidity

CATEGORY	SEVERITY	LOCATION	STATUS
Gas Optimization	NC	PintuTokenProxy.sol: 278-281, 102-104, 119-130, 143-145, 160-167, 77-89	Pending

Description:

Solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statement. Issues :

- DirtyBytesArrayToStorage
- ABIDecodeTwoDimensionalArrayMemory
- KeccakCaching
- EmptyByteArrayCopy
- DynamicArrayCleanup
- ImplicitConstructorCallvalueCheck
- TupleAssignmentMultiStackSlotComponents
- MemoryArrayCreationOverflow
- privateCanBeOverridden
- SignedArrayStorageCopy
- ABIEncoderV2StorageArrayWithMultiSlotElement
- DynamicConstructorArgumentsClippedABIV2
- UninitializedFunctionPointerInConstructor_0.4.x
- IncorrectEventSignatureInLibraries_0.4.x
- ABIEncoderV2PackedStorage_0.4.x
- ExpExponentCleanup
- EventStructWrongData

Source: PintuTokenProxy.sol

```
1   pragma solidity 0.4.24;
2
```

```
1   pragma solidity 0.5.17;
2
```

Recommendation:

Use a more recent version (at least 0.8.0), if possible.

PTU 13 Low-level Calls

CATEGORY	SEVERITY	LOCATION	STATUS
Coding Style	NC	PintuTokenProxy.sol: 276-282	Pending

Description:

It's important to handle the return values and check the success status when using low-level calls to handle potential errors.

Source: PintuTokenProxy.sol

```
276     function upgradeToAndCall(
277         address newImplementation,
278         bytes data
279     ) external payable ifAdmin {
280         _upgradeTo(newImplementation);
281         require(address(this).call.value(msg.value)(data));
282     }
```

Recommendation:

Check the call success. If the call is meant for a contract, check for code existence.

PTU 14 Redundant Statements

CATEGORY	SEVERITY	LOCATION	STATUS
Coding Style	NC	PintuTokenV1.sol: 279	Pending

Description:

Each commented line references types/identifiers, but performs no action with them, so no code will be generated for such statements and they can be removed.

Source: PintuTokenV1.sol

```
278     function _msgData() internal view returns (bytes memory) {
279         this; // silence state mutability warning without generating bytecode
280         return msg.data;
281     }
```

Recommendation:

Remove redundant statements if they congest code but offer no value.

PTU 15 Public Function That Could Be Declared External

CATEGORY	SEVERITY	LOCATION	STATUS
Coding Style	NC	PintuTokenV1.sol: 803-815	Pending

Description:

Public functions that are never called by the contract should be declared external, and its immutable parameters should be located in calldata to save gas.

Source: PintuTokenV1.sol

```
803     function initialize(
804         string memory name,
805         string memory symbol,
806         uint8 decimals,
807         uint256 initialSupply
808     ) public initializer {
809         Ownable.initialize(_msgSender());
810         _name = name;
811         _symbol = symbol;
812         _decimals = decimals;
813 
814         _mint(_msgSender(), initialSupply);
815     }
```

Recommendation:

Check the call success. If the call is meant for a contract, check for code existence. the following function parameters should change its data location:

- name location should be calldata
- symbol location should be calldata

PTU 16 Missing Checks For 'Address(0)' When Assigning Values to Address State Variables

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	NC	PintuTokenV1.sol: 362	Pending

Description:

Include checks for address(0) (the zero address) when assigning values to address state variables, especially if those variables should not be set to zero.

Source: PintuTokenV1.sol

```
360     function _setOwner(address newOwner) private {
361         address oldOwner = _owner;
362         _owner = newOwner;
363         emit OwnershipTransferred(oldOwner, newOwner);
364     }
```

Recommendation:

Before assigning the new value, it checks if _owner is not equal to address(0) (the zero address) using a require statement. If the condition is not met, it reverts the transaction with an error message.

```
360     function _setOwner(address newOwner) private {
361         require(_owner != address(0), "Old owner address cannot be zero");
362         address oldOwner = _owner;
363         _owner = newOwner;
364         emit OwnershipTransferred(oldOwner, newOwner);
365     }
```

PTU 17 Event Missing Indexed Field

CATEGORY	SEVERITY	LOCATION	STATUS
Coding Style	NC	PintuTokenV1.sol: 381, 386	Pending

Description:

Index event fields make the field more quickly accessible (concept-of-event-indexing) that parse events. This is especially useful when it comes to filtering based on an address. However, note that each index field costs extra gas during emission, so it's not necessarily best to index the maximum allowed per event (three fields). Where applicable, each `event` should use three `indexed` fields if there are three or more fields, and gas usage is not particularly of concern for the events in question. If there are fewer than three applicable fields, all of the applicable fields should be indexed.

Source: PintuTokenV1.sol

```
378     /**
379      * @dev Emitted when the pause is triggered by the owner (`account`).
380      */
381     event Paused(address account);
382
383     /**
384      * @dev Emitted when the pause is lifted by the owner (`account`).
385      */
386     event Unpaused(address account);
```

Recommendation:
Add an indexed field

```
378     /**
379      * @dev Emitted when the pause is triggered by the owner (`account`).
380      */
381     event Paused(address indexed account);
382
383     /**
384      * @dev Emitted when the pause is lifted by the owner (`account`).
385      */
386     event Unpaused(address indexed account);
387
```

PTU 18 Function Ordering Does Not Follow The Solidity Style Guide

CATEGORY	SEVERITY	LOCATION	STATUS
Coding Style	NC	PintuTokenV1.sol: all	Pending

Description:

According to the Solidity style guide, functions should be laid out in the following order: 'constructor()', 'receive()', 'fallback()', 'external', 'public', 'internal', 'private', but the cases below do not follow this patter

Current Order:

```
internal add
internal sub
internal sub
internal mul
internal div
internal div
internal mod
internal mod
external totalSupply
external balanceOf
external transfer
external allowance
external approve
external transferFrom
internal _msgSender
internal _msgData
internal initialize
public owner
public renounceOwnership
public transferOwnership
private _setOwner
public paused
public pause
public unpause
public totalSupply
public balanceOf
public transfer
public allowance
public approve
public transferFrom
public increaseAllowance
public decreaseAllowance
internal _transfer
internal _mint
internal _burn
internal _approve
internal _burnFrom
public burn
public burnFrom
public mint
public initialize
public name
public symbol
public decimals
```

Recommendation Order:

```
external totalSupply
external balanceOf
external transfer
external allowance
external approve
external transferFrom
public owner
public renounceOwnership
public transferOwnership
public paused
public pause
public unpause
public totalSupply
public balanceOf
public transfer
public allowance
public approve
public transferFrom
public increaseAllowance
public decreaseAllowance
public burn
public burnFrom
public mint
public initialize
public name
public symbol
public decimals
internal add
internal sub
internal sub
internal mul
internal div
internal div
internal mod
internal mod
internal _msgSender
internal _msgData
internal initialize
internal _transfer
internal _mint
internal _burn
internal _approve
internal _burnFrom
private _setOwner
```

OPTIMIZATION 4

Essential to profile your contract's gas usage and prioritize optimizations based on the most significant gas consumers.:

ID	Gas Optimizations	INSTANCES	STATUS
PTU-OPT 01	Don't use '_msgSender()' if not supporting **EIP-2771**	18	**Pending**
PTU-OPT 02	Use assembly to check for 'address(0)'	7	**Pending**
PTU-OPT 03	Using bools for storage incurs overhead	3	**Pending**
PTU-OPT 04	Use calldata instead of memory for function arguments that do not get mutated	2	**Pending**
PTU-OPT 05	For Operations that will not overflow, you could use unchecked	7	**Pending**
PTU-OPT 06	Use != 0 instead of > 0 for unsigned integer comparisonv	1	

PTU-OPT 01
Don't Use '_msgSender()' If Not Supporting EIP-2771

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	Optimization	PintuTokenV1.sol: 248, 336, 419, 427, 487, 512, 516, 537, 538, 556, 556, 576, 578, 673, 674, 691, 729, 734	Pending

Description:

Use `msg.sender` if the code does not implement [EIP-2771 trusted ethereum EIPS 2771 support.

Source: PintuTokenV1.sol

```
248     function _msgSender() internal view returns (address payable) {
249         return msg.sender;
250     }
```

```
335     modifier onlyOwner() {
336         require(owner() == _msgSender(), "Ownable: caller is not the owner");
337         _;
338     }
```

```
417     function pause() public onlyOwner whenNotPaused {
418         _paused = true;
419         emit Paused(_msgSender());
420     }
421
422     /**
423      * @dev Called by the owner to unpause, returns to normal state.
424      */
425     function unpause() public onlyOwner whenPaused {
426         _paused = false;
427         emit Unpaused(_msgSender());
428     }
```

```
486     function transfer(address recipient, uint256 amount) public whenNotPaused returns (bool) {
487         _transfer(_msgSender(), recipient, amount);
488         return true;
489     }
```

```
510     function approve(address spender, uint256 amount) public whenNotPaused returns (bool) {
511         require(
512             (amount == 0) || (_allowances[_msgSender()][spender] == 0),
513             "Can not approve from non-zero to non-zero allowance"
514         );
515
516         _approve(_msgSender(), spender, amount);
517         return true;
518     }
```

```
533     function transferFrom(address sender, address recipient, uint256 amount) public whenNotPaused returns (bool) {
534         _transfer(sender, recipient, amount);
535         _approve(
536             sender,
537             _msgSender(),
538             _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance")
539         );
540         return true;
541     }
```

```
555     function increaseAllowance(address spender, uint256 addedValue) public whenNotPaused returns (bool) {
556         _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
557         return true;
558     }
```

```
574     function decreaseAllowance(address spender, uint256 subtractedValue) public whenNotPaused returns (bool) {
575         _approve(
576             _msgSender(),
577             spender,
578             _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero")
579         );
580         return true;
581     }
```

```
669     function _burnFrom(address account, uint256 amount) internal {
670         _burn(account, amount);
671         _approve(
672             account,
673             _msgSender(),
674             _allowances[account][_msgSender()].sub(amount, "ERC20: burn amount exceeds allowance")
675         );
676     }
```

```
690     function burn(uint256 amount) public whenNotPaused {
691         _burn(_msgSender(), amount);
692     }
```

```
723     function initialize(
724         string memory name,
725         string memory symbol,
726         uint8 decimals,
727         uint256 initialSupply
728     ) public initializer {
729         Ownable.initialize(_msgSender());
730         _name = name;
731         _symbol = symbol;
732         _decimals = decimals;
733 
734         _mint(_msgSender(), initialSupply);
735     }
```

Recommendation:

It is recommended to have the list of trusted forwarders be immutable, and if this is not feasible, then only trusted contract owners should be able to modify it.

PTU-OPT 02
Use Assembly to Check for 'address(0)'

CATEGORY	SEVERITY	LOCATION	STATUS
Coding Style	Optimization	PintuTokenV1.sol: 356, 598, 599, 616, 635, 656,657	Pending

Description:

Assembly code is more gas-efficient compared to a Solidity condition (if (_addr == address(0))) because it doesn't involve the overhead of a high-level language comparison operation. However, remember to thoroughly test the assembly code to ensure its correctness and efficiency.

Saves 6 gas per instance

Source: PintuTokenV1.sol

```
355     function transferOwnership(address newOwner) public onlyOwner {
356         require(newOwner != address(0), "Ownable: new owner is the zero address");
357         _setOwner(newOwner);
358     }
```

```
597     function _transfer(address sender, address recipient, uint256 amount) internal {
598         require(sender != address(0), "ERC20: transfer from the zero address");
599         require(recipient != address(0), "ERC20: transfer to the zero address");
600
601         _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");
602         _balances[recipient] = _balances[recipient].add(amount);
603         emit Transfer(sender, recipient, amount);
604     }
```

```
615     function _mint(address account, uint256 amount) internal {
616         require(account != address(0), "ERC20: mint to the zero address");
617
618         _totalSupply = _totalSupply.add(amount);
619         _balances[account] = _balances[account].add(amount);
620         emit Transfer(address(0), account, amount);
621     }
```

```
634     function _burn(address account, uint256 amount) internal {
635         require(account != address(0), "ERC20: burn from the zero address");
636
637         _balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds balance");
638         _totalSupply = _totalSupply.sub(amount);
639         emit Transfer(account, address(0), amount);
640     }
```

```
655     function _approve(address owner, address spender, uint256 amount) internal {
656         require(owner != address(0), "ERC20: approve from the zero address");
657         require(spender != address(0), "ERC20: approve to the zero address");
658 
659         _allowances[owner][spender] = amount;
660         emit Approval(owner, spender, amount);
661     }
```

Recommendation:

Using assembly can sometimes lead to more gas-efficient code. Here's how you can use assembly to check for the zero address (address(0)).

```
function isZeroAddress(address _addr) external pure returns (bool) {
    // Load the address into memory
    assembly {
        // Compare the address with zero
        if iszero(_addr) {
            // If the address is zero, return 1 (true)
            mstore(0, 1)
            return(0, 32)
        }
    }
    // If the address is not zero, return 0 (false)
    return false;
}
```

PTU-OPT 03
Using Bools for Storage Incurs Overhead

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	Optimization	PintuTokenV1.sol: 274, 278, 387	Pending

Description:

Use uint256(1) and uint256(2) for true/false to avoid a Gwarmaccess (100 gas), and to avoid Gsset (20000 gas) when changing from 'false' to 'true', after having been 'true' in the past. See Info : OpenZeppelin - ReentrancyGuard

Source: PintuTokenV1.sol

```
270  contract Initializable {
271      /**
272       * @dev Indicates that the contract has been initialized.
273       */
274      bool private _initialized;
275
276      /**
277       * @dev Indicates that the contract is in the process of being initialized.
278       */
279      bool private _initializing;
280
```

```
386
387      bool private _paused;
388
```

Recommendation:

Using uint256 than bool.

PTU-OPT 04
Use Calldata Instead of Memory for Function Arguments That Do Not Get Mutated

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	Optimization	PintuTokenV1.sol: 724, 725	Pending

Description:

When a function with a `memory` array is called externally, the `abi.decode()` step has to use a for-loop to copy each index of the `calldata` to the `memory` index. Each iteration of this for-loop costs at least 60 gas (i.e. `60 * <mem_array>.length`). Using `calldata` directly bypasses this loop.

If the array is passed to an `internal` function which passes the array to another internal function where the array is modified and therefore `memory` is used in the `external` call, it's still more gas-efficient to use `calldata` when the `external` function uses modifiers, since the modifiers may prevent the internal functions from being called. Structs have the same overhead as an array of length one.

Saves 60 gas per instance

Source: PintuTokenV1.sol

```
723     function initialize(
724         string memory name,
725         string memory symbol,
726         uint8 decimals,
727         uint256 initialSupply
728     ) public initializer {
729         Ownable.initialize(_msgSender());
730         _name = name;
731         _symbol = symbol;
732         _decimals = decimals;
733 
734         _mint(_msgSender(), initialSupply);
735     }
```

Recommendation:

Update contract with the name and symbol parameters changed to calldata in the constructor.

```
interface IERC20 {
    function name() external view returns (string memory);
    function symbol() external view returns (string memory);
```

```
723     function initialize(
724         string calldata name,
725         string calldata symbol,
726         uint8 decimals,
727         uint256 initialSupply
728     ) public initializer {
729         Ownable.initialize(_msgSender());
730         _name = name;
731         _symbol = symbol;
732         _decimals = decimals;
733 
734         _mint(_msgSender(), initialSupply);
735     }
```

PTU-OPT 05
For Operations That Will Not Overflow, You Could Use Unchecked

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	Optimization	PintuTokenV1.sol: 27, 59, 81, 118	Pending

Description:

The 'unchecked' block in Solidity can be used to suppress arithmetic overflow and underflow checks for specific operations where you are certain that overflow/underflow will not occur. This can potentially save gas costs.

Source: PintuTokenV1.sol

```
26      function add(uint256 a, uint256 b) internal pure returns (uint256) {
27          uint256 c = a + b;
28          require(c >= a, "SafeMath: addition overflow");
29
30          return c;
31      }
```

```
57      function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
58          require(b <= a, errorMessage);
59          uint256 c = a - b;
60
61          return c;
62      }
```

```
80
81          uint256 c = a * b;
82          require(c / a == b, "SafeMath: multiplication overflow");
83
```

```
115     function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
116         // Solidity only automatically asserts when dividing by 0
117         require(b > 0, errorMessage);
118         uint256 c = a / b;
119         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
120
121         return c;
122     }
```

Recommendation:

Try Using `unchecked`

```
26      function add(uint256 a, uint256 b) internal pure returns (uint256) {
27          unchecked {
28              return a + b;
29          }
30      }
```

Function is wrapped in an unchecked block, indicating that arithmetic overflow and underflow checks should be disabled for the enclosed operation.

PTU-OPT 06
Use != 0 instead of > 0 for Unsigned Integer Comparison

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Issue	Optimization	PintuTokenV1.sol: 116	Pending

Description:

Using != 0 instead of > 0 for unsigned integer comparison can sometimes make the code clearer and more explicit, especially when you want to check if a value is non-zero.

Source: PintuTokenV1.sol

```
114     function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
115         // Solidity only automatically asserts when dividing by 0
116         require(b > 0, errorMessage);
117         uint256 c = a / b;
118         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
119
120         return c;
121     }
```

Recommendation:

Using != 0 instead of > 0 can improve code readability and reduce the chances of misunderstanding the intention of the comparison.

DISCLAIMER 5

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 Scolabs 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 Scolabs 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. Scolabs position is that each company and individual are responsible for their own due diligence and continuous security. Scolabs 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 Scolabs 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, SCOLABS HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, SCOLABS 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, SCOLABS 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, SCOLABS 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, MEETANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THATANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER SCOLABS NOR ANY OF SCOLABS 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. SCOLABS WILLASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENTAND 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 SCOLABS PRIOR WRITTEN CONSENT IN EACH INSTANCE.

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

THE REPRESENTATIONS AND WARRANTIES OF SCOLABS CONTAINED IN THIS AGREEMENTARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE ATHIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST SCOLABS 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 ANYASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

scolabs

Founded in 2022 in Indonesia, Scolabs is a pioneer in blockchain security, utilizing best-in-class Formal Verification and AI technology to secure and monitor blockchains, smart contracts, and Web3 apps.