Aug.09, 2023

ANTCHAIN | ZAN

Security Assessment Pioneer

—

Professional Service

Table of Contents

1. Overview

1.1. Executive Summary

Pioneer is a token project based on ERC20 standard. This report has been prepared for Pioneer project to discover issues and vulnerabilities in the source code of the Pioneer project as well as any contract dependencies that were not part of an officially recognized library. Based on the examination we have conducted by utilizing Static Analysis, Formal Verificaton and Manual Review, we have identified 1 medium vulnerability associated with centralization risks and 13 informational issues.

1.2. Project Summary

Project Name	Pioneer
Platform	Ethereum
Language	Solidity
Code Repository	https://github.com/Eleven711/Pioneer
Commit	f79133e1a07207fd3adf412c820893b31f840788

1.3. Assessment Summary

Delivery Date	Aug.09, 2023
Audit Methodology	Static Analysis, Formal Verification, Manual Review

1.4. Assessment Scope

ID	File	File Hash
1	/Pioneer/contracts/Prince.sol	e7811eba746c410e6fcae31869b0553b
2	/Pioneer/contracts/Whitelistable.sol	bb5c01b6e0e80eb8c329131be8c5a2bb

2. Checklist

2.1. Code Security

Reentrancy	DelegateCall	Integer Overflow
Input Validation	Unchecked this.call	Frozen Money
Arbitrary External Call	Unchecked Owner Transfer	Do-while Continue
Right-To-Left-Override Character	Unauthenticated Storage Access	Risk For Weak Randomness
TxOrigin	Missing Checks for Return Values	Diamond Inheritance
ThisBalance	VarType Deduction	Array Length Manipulation
Uninitialized Variable	Shadow Variable	Divide Before Multiply
Affected by Compiler Bug		

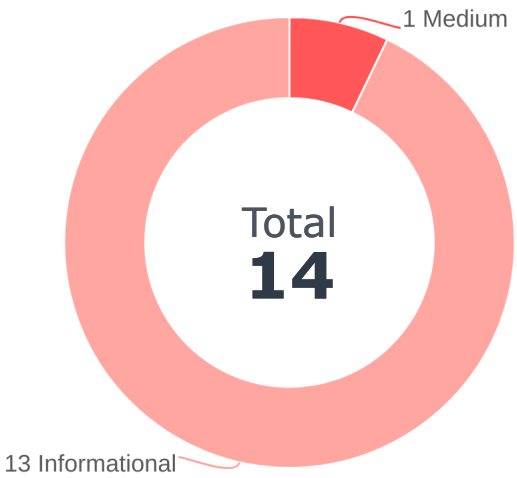
2.2. Optimization Suggestion

Compiler Version	Improper State Variable Modification
Function Visibility	Deprecated Function
Externally Controlled Variables	Code Style
Constant Specific	Event Specific
Return Value Unspecified	Inexistent Error Message
State Variable Defined Without Storage Location	Import Issue
Compare With Timestamp/Block Number/Blockhash	Constructor in Base Contract Not Implemented
Delete Struct Containing the Mapping Type	Usage of '=+'
Paths in the Modifier Not End with "_" or Revert	Non-payable Public Functions Use msg.value
Lack of SafeMath	Compiler Error/Warning
Tautology Issue	Loop Depends on Array Length
Redundant/Duplicated/Dead Code	Code Complexity/Code Inefficiency
Undeclared Resource	Optimizable Return Statement
Unused Resource	

2.3. Business Security

The Code Implementation is Consistent With Comments, Project White Papers and Other Materials
Permission Check
Address Check

3. Findings



Code Security

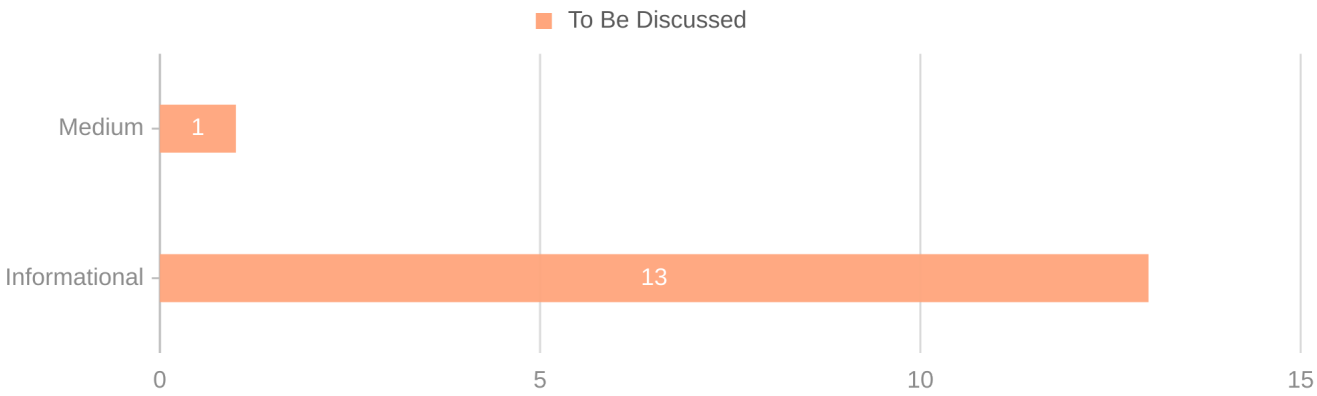
Total: 0, no vulnerabilities.

Optimization Suggestion

Total: 13, Informational: 13

Business Security

Total: 1, Medium: 1



ID	Title	Category	Severity	Status
M-01	Business Security - Centralization Risk	Business Security	Medium	To Be Discussed
I-02	Optimization Suggestion - Function Visibility Can Be External	Optimization Suggestion	Informational	To Be Discussed
I-03	Optimization Suggestion - Floating Pragma	Optimization Suggestion	Informational	To Be Discussed
I-04	Optimization Suggestion - Set the Constant to Private	Optimization Suggestion	Informational	To Be Discussed

ID	Title	Category	Severity	Status
I-05	Optimization Suggestion - Use CustomError Instead of String	Optimization Suggestion	Informational	To Be Discussed
I-06	Optimization Suggestion - Recommend to Follow Code Layout Conventions	Optimization Suggestion	Informational	To Be Discussed
I-07	Optimization Suggestion - No Check of Address Params with Zero Address	Optimization Suggestion	Informational	To Be Discussed
I-08	Optimization Suggestion - Use Assembly to Check Zero Address	Optimization Suggestion	Informational	To Be Discussed
I-09	Optimization Suggestion - Use ++i/--i Instead of i++/i--	Optimization Suggestion	Informational	To Be Discussed
I-10	Optimization Suggestion - Parameters Should Be Declared as Calldata	Optimization Suggestion	Informational	To Be Discussed
I-11	Optimization Suggestion - Variables Can Be Declared as Immutable	Optimization Suggestion	Informational	To Be Discussed
I-12	Optimization Suggestion - Functions with the Same Functionality Should Be Implemented Consistently	Optimization Suggestion	Informational	To Be Discussed
I-13	Optimization Suggestion - Redundant Function getManager	Optimization Suggestion	Informational	To Be Discussed
I-14	Optimization Suggestion - Inaccurate Code Comments and Error Messages	Optimization Suggestion	Informational	To Be Discussed

M-01|Business Security - Centralization Risk

Medium : Business Security

File Location : /Pioneer/contracts/Prince.sol:155-158,81-91,100-113,61-73

Description

The manager role possesses three unrestricted privileges.

Firstly, the manager can burn tokens at any address.

Secondly, the manager has the authorization to transfer tokens from one address within the whitelist to another whitelisted address.

Thirdly, only whitelisted addresses are authorized to invoke the transfer and transferFrom functions. However, the manager has the exclusive privilege to remove addresses from the whitelist. The removal of an address from the whitelist would result in the tokens associated with that address being effectively frozen.

Such operations have centralization risks. Users may not be willing to hold such tokens since they have good reasons to be skeptical about such centralized behaviors.

/Pioneer/contracts/Prince.sol

```
81      function transferByManager(address from, address to, uint256 amount)
82          public
83          virtual
84          onlyManager
85          inWhitelist(from)
86          inWhitelist(to)
87          returns (bool)
88      {
89          _transfer(from, to, amount);
90          return true;
91      }
```

/Pioneer/contracts/Prince.sol

```
100      function transferFrom(address from, address to, uint256 amount)
101          public
102          virtual
103          override
104          whenNotPaused()
105          inWhitelist(from)
106          inWhitelist(to)
107          returns (bool)
108      {
109          address spender = _msgSender();
110          _spendAllowance(from, spender, amount);
111          _transfer(from, to, amount);
112          return true;
113      }
```

/Pioneer/contracts/Prince.sol

```
61      function transfer(address to, uint256 amount)
62          public
63          virtual
64          override
65          whenNotPaused()
66          inWhitelist(msg.sender)
67          inWhitelist(to)
68          returns (bool)
69      {
70          address owner = _msgSender();
71          _transfer(owner, to, amount);
72          return true;
73      }
```

/Pioneer/contracts/Prince.sol

```
155
    function burn(address account, uint256 amount) external onlyManager returns (
    bool)
    {
156          _burn(account, amount);
157          return true;
158      }
```

Recommendation

It is recommended that the project party should refine the authority and pay attention to the losses caused by centralization risks to users.

Alleviation

I-02|Optimization Suggestion - Function Visibility Can Be External

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:195,81,211,180,218,187,/Pioneer/contracts/Whitelistable.sol:73

Description

Functions that are not called should be declared as external.

/Pioneer/contracts/Prince.sol

```
216       * @dev Returns the Uniform Resource Identifier (URI) for `tokenId` token.
217       */
218      function documentURI() public view returns(string memory){
219          return _documentURI;
220      }
```

/Pioneer/contracts/Prince.sol

```
193       * @dev called by the manager to unpause, returns to normal state
194       */
195      function unpause() public onlyManager{
196          _paused = false;
197          emit Unpause();
```

/Pioneer/contracts/Whitelistable.sol

```
71       * @param accounts batch of addresses to whitelist
72       */
73      function addBatchToWhitelist(address[] memory accounts) public onlyWhitelister{
74          for(uint i = 0; i < accounts.length; i++){
75              require(accounts[i] != address(0),
    "Whitelistable: account is the zero address");
```

/Pioneer/contracts/Prince.sol

```
178       * @dev Returns true if the contract is paused, and false otherwise.
179      */
180      function paused() public view virtual returns (bool){
181          return _paused;
182      }
```

/Pioneer/contracts/Prince.sol

```
209       * @param documentURI_ Uniform Resource Identifier (URI) of Document
210       */
211      function setDocumentURI(string memory documentURI_) public onlyManager{
212          _documentURI = documentURI_;
213      }
```

/Pioneer/contracts/Prince.sol

```
185         * @dev called by the manager to pause, triggers stopped state
186         */
187        function pause() public onlyManager{
188            _paused = true;
189            emit Pause();
```

/Pioneer/contracts/Prince.sol

```
79         * @return True if successful
80         */
81        function transferByManager(address from, address to, uint256 amount)
82            public
83            virtual
84
```

Recommendation

Functions that are not called in the contract should be declared as external.

Alleviation

I-03|Optimization Suggestion - Floating Pragma

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:2,/Pioneer/contracts/Whitelistable.sol:2

Description

Contracts should be deployed with the fixed compiler version that they have been tested thoroughly or make sure to pin the contract compiler version in the project config. Pinning the compiler version helps ensure that contracts are not compiled by untested compiler versions.

/Pioneer/contracts/Whitelistable.sol

```
1  //SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.18;
3
4  import "@openzeppelin/contracts/access/Ownable.sol";
```

/Pioneer/contracts/Prince.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.18;
3
4  import "@openzeppelin/contracts/access/Ownable.sol";
```

Recommendation

Use a fixed compiler version, and consider whether the bugs in the selected compiler version (https://github.com/ethereum/solidity/releases) will affect the contract.

Alleviation

I-04|Optimization Suggestion - Set the Constant to Private

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:16

Description

For constants, if the visibility is set to public, the compiler will automatically generate a getter function for it, which will consume more gas during deployment than private.

/Pioneer/contracts/Prince.sol

```
14      //using SafeMath for uint256;
15
16      uint256 public constant TOTAL_SUPPLY_MAX = 950000000000;
17      address public _manager;
18      bool private _paused;
```

Recommendation

It is recommended to set the visibility of constants to private instead of public.

Alleviation

I-05|Optimization Suggestion - Use CustomError Instead of String

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:31,43,172,143,51,/Pioneer/contracts/Whitelistable.sol:26-28,38-40,92-94,75-76

Description

When using require or revert, CustomError is more gas efficient than string description, as the error message described using CustomError is only compiled into four bytes. Especially when string exceeds 32 bytes, more gas will be consumed. Generally, around 250-270 gas can be saved for one CustomError replacement when compiler optimization is turned off, 60-80 gas can be saved even if compiler optimization is turned on.

/Pioneer/contracts/Whitelistable.sol

```
36        */
37       modifier inWhitelist(address account){
38           require(
39               _whitelisted[account],
40               "inWhitelist: account is not whitelisted"
```

/Pioneer/contracts/Prince.sol

```
41        */
42       modifier onlyManager(){
43           require(msg.sender == _manager,
 "JPETHStakingFundSP: caller is not the manager");
44           _;
45       }
```

/Pioneer/contracts/Prince.sol

```
49        */
50       modifier whenNotPaused(){
51           require(!_paused, "JPETHStakingFundSP transfer: paused");
52           _;
53       }
```

/Pioneer/contracts/Prince.sol

```
141           returns (bool)
142       {
143           require(amount + totalSupply() <= TOTAL_SUPPLY_MAX,
 "JPETHStakingFundSP: mint amount exceeds total supply");
144           _mint(to, amount);
145           addWhitelist(to);
```

/Pioneer/contracts/Whitelistable.sol

```
73
    function addBatchToWhitelist(address[] memory accounts) public onlyWhitelister{
74          for(uint i = 0; i < accounts.length; i++){
75              require(accounts[i] != address(0),
    "Whitelistable: account is the zero address");
76              require(!_whitelisted[accounts[i]],
    "Whitelistable: account is already whitelisted");
77              _whitelisted[accounts[i]] = true;
```

/Pioneer/contracts/Prince.sol

```
29          Whitelistable(manager)
30      {
31          require(manager != address(0),
    "JPETHStakingFundSP: manager is the zero address");
32          _manager = manager;
33          _decimals = tokenDecimals;
```

/Pioneer/contracts/Whitelistable.sol

```
90
91      function updateWhitelister(address newWhitelister) public onlyOwner{
92          require(
93              newWhitelister != address(0),
94              "Whitelistable: new whitelister is the zero address"
```

/Pioneer/contracts/Prince.sol

```
170         */
171       function updateManager(address newManager) external onlyOwner{
172           require(newManager != address(0),
     "JPETHStakingFundSP: new manager is the zero address");
173           _manager = newManager;
174           emit ManagerChanged(_manager);
```

/Pioneer/contracts/Whitelistable.sol

```
24        */
25      modifier onlyWhitelister(){
26          require(
27              msg.sender == _whitelister,
28              "Whitelistable: caller is not the whitelister"
```

Recommendation

When reverting, it is recommended to use CustomError instead of ordinary strings to describe the error message.

Alleviation

I-06|Optimization Suggestion - Recommend to Follow Code Layout Conventions

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:13,/Pioneer/contracts/Whitelistable.sol:10

Description

In the solidity document(https://docs.soliditylang.org/en/v0.8.17/style-guide.html), there are the following conventions for code layout:

Layout contract elements in the following order: 1. Pragma statements, 2. Import statements, 3. Interfaces, 4. Libraries, 5. Contracts.

Inside each contract, library or interface, use the following order: 1. Type declarations, 2. State variables, 3. Events, 4. Modifiers, 5. Functions.

Functions should be grouped according to their visibility and ordered: 1. constructor, 2. receive function (if exists), 3. fallback function (if exists), 4. external, 5. public, 6. internal, 7. private.

/Pioneer/contracts/Whitelistable.sol

```
8    * @dev Allows accounts to be Whitelisted by a "Whitelister" role
9    */
10  contract Whitelistable is Ownable{
11      address public _whitelister;
12      mapping(address => bool) internal _whitelisted;
```

/Pioneer/contracts/Prince.sol

```
11   * @dev ERC20 Token for JpEthStakingFundSp
12   */
13  contract Prince is Ownable, ERC20, Whitelistable{
14      //using SafeMath for uint256;
15
```

Recommendation

It is recommended to follow the above code layout conventions to improve code readability.

Alleviation

I-07|Optimization Suggestion - No Check of Address Params with Zero Address

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:27,/Pioneer/contracts/Whitelistable.sol:64,86

Description

The input parameter of the address type in the function does not use the zero address for verification.

/Pioneer/contracts/Prince.sol

```
25      event Unpause();
26
27
    constructor(
    string memory name, string memory symbol, address manager, address owner, uint8
    tokenDecimals
    )
28          ERC20(name, symbol)
29          Whitelistable(manager)
30
```

/Pioneer/contracts/Whitelistable.sol

```
84       * @param account The address to remove from the whitelist
85       */
86      function unWhitelist(address account) external onlyWhitelister{
87          _whitelisted[account] = false;
88          emit UnWhitelisted(account);
```

/Pioneer/contracts/Whitelistable.sol

```
62       * @param account The address to whitelist
63       */
64      function addWhitelist(address account) public onlyWhitelister{
65          _whitelisted[account] = true;
66          emit Whitelisted(account);
```

Recommendation

It is recommended to perform zero address verification on the input parameters of the address type.

Alleviation

I-08|Optimization Suggestion - Use Assembly to Check Zero Address

Informational : Optimization Suggestion

File Location :
/Pioneer/contracts/Prince.sol:31,172,/Pioneer/contracts/Whitelistable.sol:75,93

Description

Using assembly to check zero address can save about 18 gas in each call.

/Pioneer/contracts/Whitelistable.sol

```
91      function updateWhitelister(address newWhitelister) public onlyOwner{
92          require(
93              newWhitelister != address(0),
94              "Whitelistable: new whitelister is the zero address"
95          );
```

/Pioneer/contracts/Whitelistable.sol

```
73
    function addBatchToWhitelist(address[] memory accounts) public onlyWhitelister{
74          for(uint i = 0; i < accounts.length; i++){
75              require(accounts[i] != address(0),
    "Whitelistable: account is the zero address");
76              require(!_whitelisted[accounts[i]],
    "Whitelistable: account is already whitelisted");
77              _whitelisted[accounts[i]] = true;
```

/Pioneer/contracts/Prince.sol

```
29          Whitelistable(manager)
30      {
31          require(manager != address(0),
    "JPETHStakingFundSP: manager is the zero address");
32          _manager = manager;
33          _decimals = tokenDecimals;
```

/Pioneer/contracts/Prince.sol

```
170       */
171      function updateManager(address newManager) external onlyOwner{
172          require(newManager != address(0),
     "JPETHStakingFundSP: new manager is the zero address");
173          _manager = newManager;
174          emit ManagerChanged(_manager);
```

Recommendation

It is recommended to use assembly to check zero address.

Alleviation

I-09|Optimization Suggestion - Use ++i/--i Instead of i++/i--

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Whitelistable.sol:74

Description

Compared with i++, ++i can save about 5 gas per use.Compared with i--, --i can save about 3 gas per use in for loop.

/Pioneer/contracts/Whitelistable.sol

```
72        */
73
    function addBatchToWhitelist(address[] memory accounts) public onlyWhitelister{
74            for(uint i = 0; i < accounts.length; i++){
75                require(accounts[i] != address(0),
    "Whitelistable: account is the zero address");
76                require(!_whitelisted[accounts[i]],
    "Whitelistable: account is already whitelisted");
```

Recommendation

It is recommended to use ++i/--i instead of i++/i-- in for loop.

Alleviation

I-10|Optimization Suggestion - Parameters Should Be Declared as Calldata

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:211,/Pioneer/contracts/Whitelistable.sol:73

Description

When the compiler parses the external or public function, it can directly read the function parameters from calldata. Setting it to other storage locations may waste gas. About 300-400 gas can be saved with compiler optimization turned off while 120-150 gas can be saved vice versa.

/Pioneer/contracts/Whitelistable.sol

```
71        * @param accounts batch of addresses to whitelist
72        */
73       function addBatchToWhitelist(address[] memory accounts) public onlyWhitelister{
74           for(uint i = 0; i < accounts.length; i++){
75               require(accounts[i] != address(0),
    "Whitelistable: account is the zero address");
```

/Pioneer/contracts/Prince.sol

```
209        * @param documentURI_ Uniform Resource Identifier (URI) of Document
210        */
211       function setDocumentURI(string memory documentURI_) public onlyManager{
212           _documentURI = documentURI_;
213       }
```

Recommendation

In external or public functions, the storage location of function parameters should be set to calldata to save gas.

Alleviation

I-11|Optimization Suggestion - Variables Can Be Declared as Immutable

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:19

Description

The solidity compiler of version 0.6.5 introduces immutable to modify state variables that are only modified in the constructor. Using immutable can save gas.

/Pioneer/contracts/Prince.sol

```
17      address public _manager;
18      bool private _paused;
19      uint8 private _decimals;
20      string private _documentURI;
21
```

Recommendation

For contracts compiled with compiler of versions 0.6.5 and above, if the state variable is only modified in the constructor, it is recommended to modify the variable with immutable to save gas.

Alleviation

I-12|Optimization Suggestion - Functions with the Same Functionality Should Be Implemented Consistently

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Whitelistable.sol:64-67,75-76

Description

The functions addWhitelist and addBatchToWhitelist both add addresses to the whitelist. The difference between them is that addWhitelist adds a single address whereas addBatchToWhitelist adds a batch of addresses. In the addBatchToWhitelist function, each address being added is checked for being a zero address or already present in the whitelist before being added. However, addWhitelist function does not have such checks. The two implementations are inconsistent. Moreover, when adding an address to the whitelist, necessary zero-address validation should be conducted. However, if the address already exists in the whitelist, there is no need to revert the transaction.

/Pioneer/contracts/Whitelistable.sol

```
69      /**
70   * @dev Adds accounts to whitelist
71   * @param accounts batch of addresses to whitelist
72   */
73
     function addBatchToWhitelist(address[] memory accounts) public onlyWhitelister{
74          for(uint i = 0; i < accounts.length; i++){
75              require(accounts[i] != address(0),
     "Whitelistable: account is the zero address");
76              require(!_whitelisted[accounts[i]],
     "Whitelistable: account is already whitelisted");
77              _whitelisted[accounts[i]] = true;
78              emit Whitelisted(accounts[i]);
79          }
80      }
```

/Pioneer/contracts/Whitelistable.sol

```
60      /**
61   * @dev Adds account to whitelist
62   * @param account The address to whitelist
63   */
64      function addWhitelist(address account) public onlyWhitelister{
65          _whitelisted[account] = true;
66          emit Whitelisted(account);
67      }
```

Recommendation

It is recommended to implement the "adding addresses to the whitelist" functionality within a separate internal function, which can be called by both addWhitelist and addBatchToWhitelist functions. This approach ensures consistency in the implementation and avoids code redundancy.

Alleviation

I-13|Optimization Suggestion - Redundant Function getManager

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:163-165,17

Description

The visibility of the state variable _manager is public, which means that _manager has a default getter function to get its value, and there is no need for an additional getManager function to get the value of this variable. Redundant function will consume extra gas.

/Pioneer/contracts/Prince.sol

```
160      /**
161  * @dev getManager address
162  */
163      function getManager() external view returns (address){
164          return _manager;
165      }
```

/Pioneer/contracts/Prince.sol

```
17      address public _manager;
```

Recommendation

It is recommended to remove the function getManager or change the visibility of the state variable _manager to private.

Alleviation

I-14|Optimization Suggestion - Inaccurate Code Comments and Error Messages

Informational : Optimization Suggestion

File Location : /Pioneer/contracts/Prince.sol:143,31,43,48,51,172,11

Description

In the Prince.sol smart contract, the name 'JPETHStakingFundSP', which refers to another contract JpEthStakingFundSp.sol, appears multiple times within both the code comments and error messages. However, no inheritance or other connections between these two contracts were found. This discrepancy could potentially cause confusion.

/Pioneer/contracts/Prince.sol

```
1       /**
2    * @dev Throws if JPETHStakingFundSP transfer: paused
3    */
4       modifier whenNotPaused(){
5           require(!_paused, "JPETHStakingFundSP transfer: paused");
6           _;
7       }
```

/Pioneer/contracts/Prince.sol

```
171       function updateManager(address newManager) external onlyOwner{
172           require(newManager != address(0),
    "JPETHStakingFundSP: new manager is the zero address");
173           _manager = newManager;
174           emit ManagerChanged(_manager);
175       }
```

/Pioneer/contracts/Prince.sol

```
42       modifier onlyManager(){
43           require(msg.sender == _manager,
    "JPETHStakingFundSP: caller is not the manager");
44           _;
45       }
```

/Pioneer/contracts/Prince.sol

```
9    /**
10   * @title Prince Token
11   * @dev ERC20 Token for JpEthStakingFundSp
12   */
```

/Pioneer/contracts/Prince.sol

```
27
    constructor(
    string memory name, string memory symbol, address manager, address owner, uint8
    tokenDecimals
    )
28          ERC20(name, symbol)
29          Whitelistable(manager)
30      {
31          require(manager != address(0),
    "JPETHStakingFundSP: manager is the zero address");
32          _manager = manager;
33          _decimals = tokenDecimals;
34          _paused = true;
35          _transferOwnership(owner);
36
37      }
```

/Pioneer/contracts/Prince.sol

```
138      function mint(address to, uint256 amount)
139          external
140          onlyManager
141          returns (bool)
142      {
143          require(amount + totalSupply() <= TOTAL_SUPPLY_MAX,
     "JPETHStakingFundSP: mint amount exceeds total supply");
144          _mint(to, amount);
145          addWhitelist(to);
146          return true;
147      }
```

Recommendation

It is recommended that the project party thoroughly inspect these code comments and error messages to determine whether they have been introduced as a result of an erroneous code fork.

Alleviation

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

5.1 Visibility

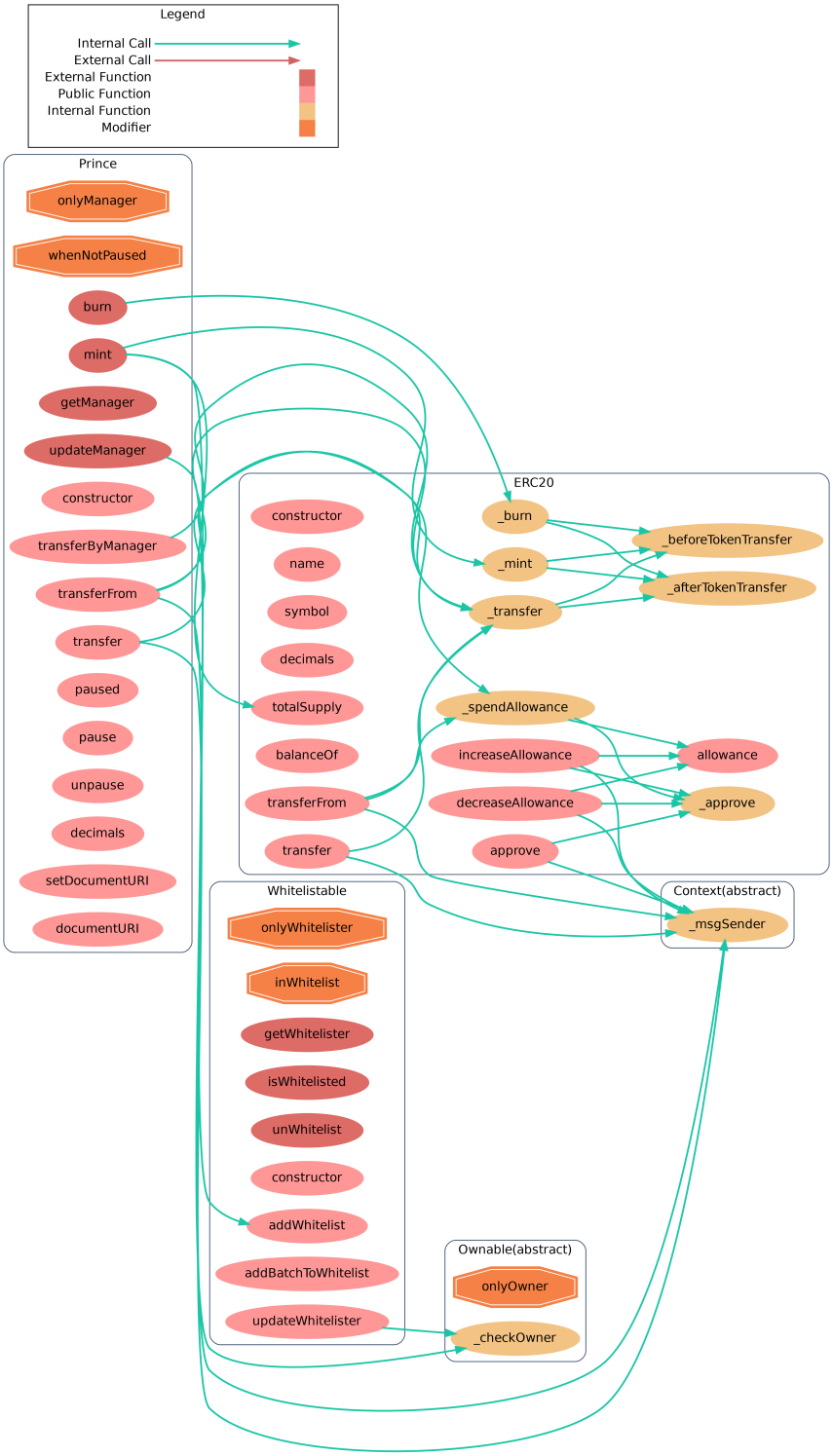
Contract	FuncName	Visibility	Mutability	Modifiers
Whitelistable	_CTOR_	public	Y	
Whitelistable	getWhitelister	external	N	
Whitelistable	isWhitelisted	external	N	
Whitelistable	addWhitelist	public	Y	onlyWhitelister
Whitelistable	addBatchToWhitelist	public	Y	onlyWhitelister
Whitelistable	unWhitelist	external	Y	onlyWhitelister
Whitelistable	updateWhitelister	public	Y	onlyOwner
Prince	_CTOR_	public	Y	ERC20,Whitelistable
Prince	transfer	public	Y	whenNotPaused,inWhitelist,inWhitelist
Prince	transferByManager	public	Y	onlyManager,inWhitelist,inWhitelist
Prince	transferFrom	public	Y	whenNotPaused,inWhitelist,inWhitelist
Prince	mint	external	Y	onlyManager
Prince	burn	external	Y	onlyManager
Prince	getManager	external	N	
Prince	updateManager	external	Y	onlyOwner
Prince	paused	public	N	

Contract	FuncName	Visibility	Mutability	Modifiers
Prince	pause	public	Y	onlyManager
Prince	unpause	public	Y	onlyManager
Prince	decimals	public	N	
Prince	setDocumentURI	public	Y	onlyManager
Prince	documentURI	public	N	

5. Appendix

5.2 Call Graph

Prince



5. Appendix

5.3 Inheritance Graph

Prince