北京链安
Chains Guard Technology

PHAToken Smart Contract Audit Report

Beijing Chains Guard Technology

Documentation

File name	PHAToken Smart Contract Audit Report		
Serial number	CG-YMSJ-202008023		
Confidentiality	Business secret	Version	V1.1
Author	ChainsGuard Security Center	Date	2020-08-31

Scope of application

This security assessment was authorized by the authorized party, and The Beijing ChainsGuard Network Technology Co., Ltd. (hereinafter referred to as "ChainsGuard") conducted an in-depth evaluation of the security risks of the PHAToken Smart Contract. security assessment and make recommendations to the reinforcement of the security situation in the main network is limited to Beijing ChainsGuard security and authorized party insiders circulated.

Version change record

Date	Version	Description	Modify by
2020-08-31	V1.1	Document creation	ChainsGuard Security Center

Catalogue

Disclaimer

The audit report is a technical security audit for the authorized party. The purpose of this audit is to provide the authorized party with a reference basis for conducting its business security assessment and optimization, The regulatory regime of the business model, or any other statement about the applicability of the application, as well as a statement or warranty that the application is in error-free behavior. This report cannot be used as a proof of that these tested systems and codes are absolutely secure and there are no other security risks.

The audit report only covers the code, installation packages and other materials provided by the authorized party, and its conclusion is only applicable to the corresponding version of the application. Once the relevant code, configuration, and operating environment change, the corresponding conclusion will no longer be applicable.

1 Introduction

1.1 Overview

This document includes the results of the audit performed by the Chains Guard Team on the PHAToken project, at the request of the PHAToken tea m. The goal of this audit is to review the smart contract code solidity impl ementation, study potential security vulnerabilities, its general designand arc hitecture, and uncover bugs that could compromise the software in product ion.

1.2 Audit Time

Evaluation test time	
Start time	2020-08-31
End time	2020-08-31

1.3 Audit Unit

Company Name	The Beijing ChainsGuard Network Technology Co., Ltd.
Web Site	https://www.chainsguard.com/

1.4 Audit Object

Name	Contract Address	Deploy Version
PHAToken	0x6c5bA91642F10282b576d91922Ae6448C9d52f4E	v0.5.16+commit.9c3226ce

2 Security Audit Summary

2.1 Vulnerability Statistics

Vulnerabilities	High risk	Medium risk	Low risk
0	0	0	1

[Note] A brief description of the hazard classification method is as follows

High: It directly causes the system to be controlled or the data to be destroyed. Once it occurs, it is a serious security event.

Medium: It may lead to the leakage of important information or may cause the system to be controlled.

Low: Non-critical information leaks or minor security issues generally do not lead to serious security incidents.

2.2 Audit items

For the ERC20 token contract, we focus on the audit of the following inspection items:

Attack surface	Check list	status	description
Reentrancy Attack	Cross-contract interaction	Pass	Unprotected sensitive functions call external contracts
	ETH Transfer	Pass	Unrestricted Gas transfer ETH has a hidden danger of reentry
Unauthorized access	Constructor does not match	Pass	Whether the contract name and constructor in the lower version do not match
	Privileged function exposure	Pass	Exposure of privileged functions caused by incorrect authentication methods
	tx.origin variable abuse	Pass	Whether the contract uses tx.origin for identity authentication
	Access control flaws	Pass	Unreasonable settings for the visibility of functions and state variables
Numerical overflow	Overflow & underflow	Pass	Does the contract have common overflow or underflow vulnerabilities
Race condition	Transaction order dependence	Low risk	Does the final state of the contract depend on the order of transactions
Denial of service	Unexpected transaction rollback	Pass	Is the contract vulnerable to revert cause denial of service
	Gas Price exceeded	Pass	Excessive Gas Price caused by excessive loop
call injection	call function abuse	Pass	The contract receives external input as a parameter of the call function
Fake recharge	Recharge result check	Pass	Whether the contract uses an incorrect method to check the recharge result
Miner privileges	Timestamp dependence	Pass	Does the contract rely on the timestamp to complete the main function
	fake-random number dependence	Pass	Does the contract rely on pseudo-random numbers to complete its main functions
Other checks	External input check	Pass	Whether the contract verifies the legality of external input
	Use untrusted libraries	Pass	Whether the contract uses untrusted (unsafe) libraries

	Leakage of sensitive information	Pass	Does the contract have hidden dangers of leaking sensitive information
	Blackhole	Pass	Whether the contract locks ETH or tokens indefinitely
	Contract backdoor	Pass	Does the contract have a backdoor that can be controlled by the project party

3 Checklist

For smart contract source code audit, we focus on the detection of the following security points:

3.1 Reentrancy Attack

One of the main dangers of invoking external contracts is that they can take over the control process. In a reentrancy attack (also called a recursive call attack), a malicious contract will call back the calling contract before the first call to the function is completed. This may cause different calls of the function to interact in an undesirable way

Test result: **Pass**

3.2 Permission Control

Check whether the functions in the contract use public, private and other keywords correctly for visibility modification. Check whether the contract i

s correctly defined and use the modifier to restrict access to key functions to avoid unauthorized access.

Test result: **Pass**

3.3 Overflow & Underflow

The numerical processing in the smart contract needs to strictly check the arithmetic overflow/underflow problem, the conventional addition and subtraction arithmetic processing is easy to cause integer overflow or underflow, especially when processing the account amount of similar tokens, it is necessary to strictly judge the size of the account amount. Normally numerical calculation is recommended to use Zeppelin open source module SafeMath for processing.

Test result: **pass**

3.4 Call Injection

When using a low-level interface call() for contract interaction in a smart contract, the interface is improperly used. An attacker can control the parameters, method selectors, or execution byte contents of the call interface to perform dangerous operations such as unauthorized transfer, mint, and token burn.

Test result: **pass**

3.5 Race Conditions

Every transaction and smart contract calls requires miner to mine to con firm. Sharing a state between different functions may cause different proces sing results due to changes in the execution order.

Test result: **Low risk**

Vulnerability details: The location of the vulnerability is in the ERC20#ap prove() function. The approval function can be called multiple times. Ethere um In order to encourage miners to mine, miners can decide which transac tions to pack. In order to maximize benefits, miners usually choose to pack larger Gas Price transactions instead of relying on the order of transactions. Therefore, when the sender calls the approval function for the second tim e, if the consumer (attacker) knows the intention, and before the miner pac ks the second approval transaction, a higher Gas Price is used to spend the first authorization token . When the miner packs the second transaction, a token authorization beyond the original intention will be formed.

Fix suggestion: add require((allowance[_msgSender()][spender] == 0) || (a mount == 0)); code to limit the second call, forcibly use increaseAllowance & decreaseAllowance function instead.

```
310     /**
311      * @dev See {IERC20-approve}.
312      *
313      * Requirements:
314      *
315      * - `spender` cannot be the zero address.
316      *
317      * **ChainsGuard** 授权指定账户可操作的额度
318      */
319     function approve(address spender, uint256 amount) public returns (bool) {
320         // **ChainsGuard** 警告: 未缓解 approve 条件竞争隐患
321         // require((allowance[_msgSender()][spender] == 0) || (amount == 0));
322         _approve(_msgSender(), spender, amount);
323         return true;
324     }
325
```

3.6 Design Flaw

Unreasonable code logic processing or wrong coding will lead to abnormal contract execution, functional or security does not meet expectations, etc. Before the contract is officially deployed, the entire code needs to be checked for integrity and multi-faceted to ensure that the contract code logic is correct.

Test result: **pass**

3.7 DDoS Attack

"Unrecoverable malicious operations or controllable unlimited resource consumption" can all be called denial of service in smart contracts. An attacker may call a contract function by constructing malicious parameters, which may lead to logical processing problems that cannot be recovered by the

deployed contract, and the service is denied The smart contract usually m akes the entire code logic unable to continue execution and "stop service".

Test result: **pass**

3.8 Pseudo-random Number

The values such as block.timestamp, block.number, etc. in the smart cont ract are predictable. When the contract involves random number generatio n, do not use easily accessible variables or parameters as seeds to generate random numbers. A better method is to use The external service Oraclize performs random number generation and processing.

Test result: **pass**

3.9 Front-running

Every transaction and smart contract calls requires miners to confirm the mining. During this time period, it is extremely easy for an attacker to mal iciously read the transaction or call details, and then use the high Gas min er incentive mechanism to give priority to their transactions or calls. form F ront-Running.

Test result: **pass**

3.10 Short Address Attack

When the virtual machine parses the smart contract bytes, it will automa tically filling the number of parameter bytes. The attacker can construct mal

formed data to make smart contract calls. Automatic byte filling will cause unintended parameter values to change.

Test result: **pass**

3.11 Data Privacy

The smart contract code needs to encrypt and protect the stored user's private information. Anyone can obtain the relevant information stored in th e contract by querying the data on the chain. If the private information is not encrypted or the encryption is not strict, it is easy to cause privacy lea kage.

Test result: **pass**

3.12 Malicious backdoor

In the blockchain ecosystem, some project parties themselves do not ha ve the ability to develop smart contracts. They usually choose some smart contract automation generation tools for one-click generation and automati c deployment on the blockchain. This opens up opportunities for hackers t o insert backdoor codes into smart contracts. Once the backdoor code is in sert into the smart contract, the hacker can use its backdoor to manipulate the contract arbitrarily, which will cause fatal harm to the project party.

Test result: **pass**

3.13 Code Optimization

Each step of smart contract code execution needs to spend corresponding gas. Non-standard code writing will cause unnecessary gas waste. Code logic optimization can help the smart contract logic to be clearer, the execution efficiency is higher, and the Gas costs less.

Test result: **pass**

4 Summary of Security Audit

This contract, code style is good, no exploitable security issues found. The overall estimated safety status is **warning status.**

The intelligent contract audit results only provide the actual basis for the authorizer to formulate corresponding security measures and solutions.

5 Smart Contract Code

```
1   /**
2    *Submitted for verification at Etherscan.io on 2020-04-30
3   */
4
5   pragma solidity ^0.5.0;
6
7   // **ChainsGuard** 通过：合约调用者模块
8   contract Context {
9       // Empty internal constructor, to prevent people from mistakenly deploying
10      // an instance of this contract, which should be used via inheritance.
11      constructor () internal { }
12      // solhint-disable-previous-line no-empty-blocks
13
14      // **ChainsGuard** 通过：当前调用者
15      function _msgSender() internal view returns (address payable) {
16          return msg.sender;
17      }
18
19      // **ChainsGuard** 通过：当前调用数据
20      function _msgData() internal view returns (bytes memory) {
```

```
21              this; // silence state mutability warning without generating bytecode - see
    https://github.com/ethereum/solidity/issues/2691
22              return msg.data;
23          }
24  }
25
26  // **ChainsGuard** 通过：标准 ERC20 接口
27  interface IERC20 {
28          /**
29           * @dev Returns the amount of tokens in existence.
30           */
31          function totalSupply() external view returns (uint256);
32
33          /**
34           * @dev Returns the amount of tokens owned by `account`.
35           */
36          function balanceOf(address account) external view returns (uint256);
37
38          /**
39           * @dev Moves `amount` tokens from the caller's account to `recipient`.
40           *
```

```
41          * Returns a boolean value indicating whether the operation succeeded.
42          *
43          * Emits a {Transfer} event.
44          */
45         function transfer(address recipient, uint256 amount) external returns (bool);
46
47         /**
48          * @dev Returns the remaining number of tokens that `spender` will be
49          * allowed to spend on behalf of `owner` through {transferFrom}. This is
50          * zero by default.
51          *
52          * This value changes when {approve} or {transferFrom} are called.
53          */
54         function allowance(address owner, address spender) external view returns (uint256);
55
56         /**
57          * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
58          *
59          * Returns a boolean value indicating whether the operation succeeded.
60          *
61          * IMPORTANT: Beware that changing an allowance with this method brings the risk
```

```
62          * that someone may use both the old and the new allowance by unfortunate
63          * transaction ordering. One possible solution to mitigate this race
64          * condition is to first reduce the spender's allowance to 0 and set the
65          * desired value afterwards:
66          * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
67          *
68          * Emits an {Approval} event.
69          */
70         function approve(address spender, uint256 amount) external returns (bool);
71
72         /**
73          * @dev Moves `amount` tokens from `sender` to `recipient` using the
74          * allowance mechanism. `amount` is then deducted from the caller's
75          * allowance.
76          *
77          * Returns a boolean value indicating whether the operation succeeded.
78          *
79          * Emits a {Transfer} event.
80          */
81         function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
82
```

```
83      /**
84       * @dev Emitted when `value` tokens are moved from one account (`from`) to
85       * another (`to`).
86       *
87       * Note that `value` may be zero.
88       */
89      event Transfer(address indexed from, address indexed to, uint256 value);
90
91      /**
92       * @dev Emitted when the allowance of a `spender` for an `owner` is set by
93       * a call to {approve}. `value` is the new allowance.
94       */
95      event Approval(address indexed owner, address indexed spender, uint256 value);
96  }
97
98  // **ChainsGuard** 通过：安全的数学计算模块
99  library SafeMath {
100     /**
101      * @dev Returns the addition of two unsigned integers, reverting on
102      * overflow.
103      *
```

```
104         * Counterpart to Solidity's `+` operator.
105         *
106         * Requirements:
107         * - Addition cannot overflow.
108         *
109         * **ChainsGuard** 通过：加法运算
110         */
111     function add(uint256 a, uint256 b) internal pure returns (uint256) {
112             uint256 c = a + b;
113             require(c >= a, "SafeMath: addition overflow");// **ChainsGuard** 溢出检查
114
115             return c;
116     }
117
118     /**
119         * @dev Returns the subtraction of two unsigned integers, reverting on
120         * overflow (when the result is negative).
121         *
122         * Counterpart to Solidity's `-` operator.
123         *
124         * Requirements:
```

```
125         * - Subtraction cannot overflow.
126         *
127         * **ChainsGuard** 通过：减法运算
128         */
129        function sub(uint256 a, uint256 b) internal pure returns (uint256) {
130            return sub(a, b, "SafeMath: subtraction overflow");
131        }
132
133        /**
134         * @dev Returns the subtraction of two unsigned integers, reverting with custom message on
135         * overflow (when the result is negative).
136         *
137         * Counterpart to Solidity's `-` operator.
138         *
139         * Requirements:
140         * - Subtraction cannot overflow.
141         *
142         * _Available since v2.4.0._
143         *
144         * **ChainsGuard** 通过：减法运算
145         */
```

```
146     function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
147         require(b <= a, errorMessage);// **ChainsGuard** 防止溢出的检查
148         uint256 c = a - b;
149
150         return c;
151     }
152
153     /**
154      * @dev Returns the multiplication of two unsigned integers, reverting on
155      * overflow.
156      *
157      * Counterpart to Solidity's `*` operator.
158      *
159      * Requirements:
160      * - Multiplication cannot overflow.
161      *
162      * **ChainsGuard** 通过：乘法运算
163      */
164     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
165         // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
166         // benefit is lost if 'b' is also tested.
```

```
167         // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
168         if (a == 0) {
169             return 0;
170         }
171
172         uint256 c = a * b;
173         require(c / a == b, "SafeMath: multiplication overflow");// **ChainsGuard** 溢出检查
174
175         return c;
176     }
177
178     /**
179      * @dev Returns the integer division of two unsigned integers. Reverts on
180      * division by zero. The result is rounded towards zero.
181      *
182      * Counterpart to Solidity's `/` operator. Note: this function uses a
183      * `revert` opcode (which leaves remaining gas untouched) while Solidity
184      * uses an invalid opcode to revert (consuming all remaining gas).
185      *
186      * Requirements:
187      * - The divisor cannot be zero.
```

```
188      *
189      * **ChainsGuard** 通过：除法运算
190      */
191     function div(uint256 a, uint256 b) internal pure returns (uint256) {
192         return div(a, b, "SafeMath: division by zero");
193     }
194
195     /**
196      * @dev Returns the integer division of two unsigned integers. Reverts with custom message on
197      * division by zero. The result is rounded towards zero.
198      *
199      * Counterpart to Solidity's `/` operator. Note: this function uses a
200      * `revert` opcode (which leaves remaining gas untouched) while Solidity
201      * uses an invalid opcode to revert (consuming all remaining gas).
202      *
203      * Requirements:
204      * - The divisor cannot be zero.
205      *
206      * _Available since v2.4.0._
207      *
208      * **ChainsGuard** 通过：除法运算
```

```
209      */
210     function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
211         // Solidity only automatically asserts when dividing by 0
212         require(b > 0, errorMessage);// **ChainsGuard** 防止除零的检查
213         uint256 c = a / b;
214         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
215
216         return c;
217     }
218
219     /**
220      * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
221      * Reverts when dividing by zero.
222      *
223      * Counterpart to Solidity's `%` operator. This function uses a `revert`
224      * opcode (which leaves remaining gas untouched) while Solidity uses an
225      * invalid opcode to revert (consuming all remaining gas).
226      *
227      * Requirements:
228      * - The divisor cannot be zero.
229      *
```

```
230      * **ChainsGuard** 通过：取模运算
231      */
232     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
233         return mod(a, b, "SafeMath: modulo by zero");
234     }
235
236     /**
237      * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
238      * Reverts with custom message when dividing by zero.
239      *
240      * Counterpart to Solidity's `%` operator. This function uses a `revert`
241      * opcode (which leaves remaining gas untouched) while Solidity uses an
242      * invalid opcode to revert (consuming all remaining gas).
243      *
244      * Requirements:
245      * - The divisor cannot be zero.
246      *
247      * _Available since v2.4.0._
248      *
249      * **ChainsGuard** 通过：取模运算
250      */
```

```
251     function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
252         require(b != 0, errorMessage);// **ChainsGuard** 防止除零的检查
253         return a % b;
254     }
255 }
256
257 // **ChainsGuard** 警告：未缓解 approve 条件竞争隐患
258 contract ERC20 is Context, IERC20 {
259     // **ChainsGuard** 使用 SafeMath 模块辅助 uint256 类型进行数学计算
260     using SafeMath for uint256;
261     // **ChainsGuard** 账户余额账本
262     mapping (address => uint256) private _balances;
263     // **ChainsGuard** 账户授权账本
264     mapping (address => mapping (address => uint256)) private _allowances;
265     // **ChainsGuard** 代币总量
266     uint256 private _totalSupply;
267
268     /**
269      * @dev See {IERC20-totalSupply}.
270      *
271      * **ChainsGuard** 通过：返回代币总量
```

```
272      */
273     function totalSupply() public view returns (uint256) {
274         return _totalSupply;
275     }
276
277     /**
278      * @dev See {IERC20-balanceOf}.
279      *
280      * **ChainsGuard** 通过：返回指定账户余额
281      */
282     function balanceOf(address account) public view returns (uint256) {
283         return _balances[account];
284     }
285
286     /**
287      * @dev See {IERC20-transfer}.
288      *
289      * Requirements:
290      *
291      * - `recipient` cannot be the zero address.
292      * - the caller must have a balance of at least `amount`.
```

```
293      *
294      * **ChainsGuard** 通过：转账函数
295      */
296     function transfer(address recipient, uint256 amount) public returns (bool) {
297         _transfer(_msgSender(), recipient, amount);
298         return true;
299     }
300
301     /**
302      * @dev See {IERC20-allowance}.
303      *
304      * **ChainsGuard** 通过：获取指定账户间的授权额度
305      */
306     function allowance(address owner, address spender) public view returns (uint256) {
307         return _allowances[owner][spender];
308     }
309
310     /**
311      * @dev See {IERC20-approve}.
312      *
313      * Requirements:
```

```
314         *
315         * - `spender` cannot be the zero address.
316         *
317         * **ChainsGuard** 授权指定账户可操作的额度
318         */
319     function approve(address spender, uint256 amount) public returns (bool) {
320             // **ChainsGuard** 警告：未缓解 approve 条件竞争隐患
321             // require((allowance[_msgSender()][spender] == 0) || (amount == 0));
322             _approve(_msgSender(), spender, amount);
323             return true;
324     }
325
326     /**
327      * @dev See {IERC20-transferFrom}.
328      *
329      * Emits an {Approval} event indicating the updated allowance. This is not
330      * required by the EIP. See the note at the beginning of {ERC20};
331      *
332      * Requirements:
333      * - `sender` and `recipient` cannot be the zero address.
334      * - `sender` must have a balance of at least `amount`.
```

```
335         * - the caller must have allowance for `sender`'s tokens of at least
336         * `amount`.
337         *
338         * **ChainsGuard** 通过：代理转账函数
339         */
340     function transferFrom(address sender, address recipient, uint256 amount) public returns (bool) {
341             // **ChainsGuard** 代理转账操作
342             _transfer(sender, recipient, amount);
343             // **ChainsGuard** 调整授权额度
344             _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer
    amount exceeds allowance"));
345             return true;
346     }
347
348     /**
349      * @dev Atomically increases the allowance granted to `spender` by the caller.
350      *
351      * This is an alternative to {approve} that can be used as a mitigation for
352      * problems described in {IERC20-approve}.
353      *
354      * Emits an {Approval} event indicating the updated allowance.
```

```
355         *
356         * Requirements:
357         *
358         * - `spender` cannot be the zero address.
359         *
360         * **ChainsGuard** 通过：增加授权额度
361         */
362     function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
363             _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
364             return true;
365     }
366
367     /**
368      * @dev Atomically decreases the allowance granted to `spender` by the caller.
369      *
370      * This is an alternative to {approve} that can be used as a mitigation for
371      * problems described in {IERC20-approve}.
372      *
373      * Emits an {Approval} event indicating the updated allowance.
374      *
375      * Requirements:
```

```
376         *
377         * - `spender` cannot be the zero address.
378         * - `spender` must have allowance for the caller of at least
379         * `subtractedValue`.
380         *
381         * **ChainsGuard** 通过：减少授权额度
382         */
383     function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
384             _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue,
    "ERC20: decreased allowance below zero"));
385             return true;
386     }
387
388     /**
389      * @dev Moves tokens `amount` from `sender` to `recipient`.
390      *
391      * This is internal function is equivalent to {transfer}, and can be used to
392      * e.g. implement automatic token fees, slashing mechanisms, etc.
393      *
394      * Emits a {Transfer} event.
395      *
```

```
396      * Requirements:
397      *
398      * - `sender` cannot be the zero address.
399      * - `recipient` cannot be the zero address.
400      * - `sender` must have a balance of at least `amount`.
401      *
402      * **ChainsGuard** 通过：转账操作
403      */
404     function _transfer(address sender, address recipient, uint256 amount) internal {
405         require(sender != address(0), "ERC20: transfer from the zero address");
406         require(recipient != address(0), "ERC20: transfer to the zero address");
407
408         _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");
409         _balances[recipient] = _balances[recipient].add(amount);
410         emit Transfer(sender, recipient, amount);
411     }
412
413     /** @dev Creates `amount` tokens and assigns them to `account`, increasing
414      * the total supply.
415      *
416      * Emits a {Transfer} event with `from` set to the zero address.
```

```
417         *
418         * Requirements
419         *
420         * - `to` cannot be the zero address.
421         *
422         * **ChainsGuard** 通过：铸造 token
423         */
424        function _mint(address account, uint256 amount) internal {
425            require(account != address(0), "ERC20: mint to the zero address");
426
427            // **ChainsGuard** 增加总量
428            _totalSupply = _totalSupply.add(amount);
429            // **ChainsGuard** 增加账户余额
430            _balances[account] = _balances[account].add(amount);
431            emit Transfer(address(0), account, amount);
432        }
433
434        /**
435         * @dev Destroys `amount` tokens from `account`, reducing the
436         * total supply.
437         *
```

```
438     * Emits a {Transfer} event with `to` set to the zero address.
439     *
440     * Requirements
441     *
442     * - `account` cannot be the zero address.
443     * - `account` must have at least `amount` tokens.
444     *
445     * **ChainsGuard** 通过：销毁 token
446     */
447    function _burn(address account, uint256 amount) internal {
448        require(account != address(0), "ERC20: burn from the zero address");
449
450        // **ChainsGuard** 减少账户余额
451        _balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds balance");
452        // **ChainsGuard** 减少总量
453        _totalSupply = _totalSupply.sub(amount);
454        emit Transfer(account, address(0), amount);
455    }
456
457    /**
458     * @dev Sets `amount` as the allowance of `spender` over the `owner`s tokens.
```

```
459         *
460         * This is internal function is equivalent to `approve`, and can be used to
461         * e.g. set automatic allowances for certain subsystems, etc.
462         *
463         * Emits an {Approval} event.
464         *
465         * Requirements:
466         *
467         * - `owner` cannot be the zero address.
468         * - `spender` cannot be the zero address.
469         *
470         * **ChainsGuard** 通过：token 授权
471         */
472        function _approve(address owner, address spender, uint256 amount) internal {
473            require(owner != address(0), "ERC20: approve from the zero address");
474            require(spender != address(0), "ERC20: approve to the zero address");
475
476            _allowances[owner][spender] = amount;
477            emit Approval(owner, spender, amount);
478        }
479
```

```
480     /**
481      * @dev Destroys `amount` tokens from `account`.`amount` is then deducted
482      * from the caller's allowance.
483      *
484      * See {_burn} and {_approve}.
485      *
486      * **ChainsGuard** 通过：代理销毁 token
487      */
488     function _burnFrom(address account, uint256 amount) internal {
489         // **ChainsGuard** 销毁 token
490         _burn(account, amount);
491         // **ChainsGuard** 调整授权额度
492         _approve(account, _msgSender(), _allowances[account][_msgSender()].sub(amount, "ERC20: burn
    amount exceeds allowance"));
493     }
494 }
495
496 // **ChainsGuard** 通过
497 contract ERC20Detailed is IERC20 {
498     string private _name;
499     string private _symbol;
```

```
500     uint8 private _decimals;
501
502     /**
503      * @dev Sets the values for `name`, `symbol`, and `decimals`. All three of
504      * these values are immutable: they can only be set once during
505      * construction.
506      */
507     constructor (string memory name, string memory symbol, uint8 decimals) public {
508         _name = name;
509         _symbol = symbol;
510         _decimals = decimals;
511     }
512
513     /**
514      * @dev Returns the name of the token.
515      *
516      * **ChainsGuard** 通过：返回 token 名称
517      */
518     function name() public view returns (string memory) {
519         return _name;
520     }
```

```
521
522     /**
523      * @dev Returns the symbol of the token, usually a shorter version of the
524      * name.
525      *
526      * **ChainsGuard** 通过：返回 token 符号
527      */
528     function symbol() public view returns (string memory) {
529         return _symbol;
530     }
531
532     /**
533      * @dev Returns the number of decimals used to get its user representation.
534      * For example, if `decimals` equals `2`, a balance of `505` tokens should
535      * be displayed to a user as `5,05` (`505 / 10 ** 2`).
536      *
537      * Tokens usually opt for a value of 18, imitating the relationship between
538      * Ether and Wei.
539      *
540      * NOTE: This information is only used for _display_ purposes: it in
541      * no way affects any of the arithmetic of the contract, including
```

```
542         * {IERC20-balanceOf} and {IERC20-transfer}.
543         *
544         * **ChainsGuard** 通过：返回 token 小数点
545         */
546     function decimals() public view returns (uint8) {
547         return _decimals;
548     }
549 }
550
551 // **ChainsGuard** 提示：合约部署时初始化部署者为合约所有者
552 contract Ownable is Context {
553     // **ChainsGuard** 合约所有者账户
554     address private _owner;
555
556     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
557
558     /**
559      * @dev Initializes the contract setting the deployer as the initial owner.
560      */
561     constructor () internal {
562         // **ChainsGuard** 提示：合约部署时初始化部署者为合约所有者
```

```
563         address msgSender = _msgSender();
564         _owner = msgSender;
565         emit OwnershipTransferred(address(0), msgSender);
566     }
567
568     /**
569      * @dev Returns the address of the current owner.
570      *
571      * **ChainsGuard** 通过：查询合约所有者账户
572      */
573     function owner() public view returns (address) {
574         return _owner;
575     }
576
577     /**
578      * @dev Throws if called by any account other than the owner.
579      *
580      * **ChainsGuard** 通过：确保只有合约拥有者能够正确调用
581      */
582     modifier onlyOwner() {
583         require(isOwner(), "Ownable: caller is not the owner");
```

```
584             _;
585         }
586
587         /**
588          * @dev Returns true if the caller is the current owner.
589          *
590          * **ChainsGuard** 通过：判断当前调用者是否为合约所有者
591          */
592         function isOwner() public view returns (bool) {
593             return _msgSender() == _owner;
594         }
595
596         /**
597          * @dev Leaves the contract without owner. It will not be possible to call
598          * `onlyOwner` functions anymore. Can only be called by the current owner.
599          *
600          * NOTE: Renouncing ownership will leave the contract without an owner,
601          * thereby removing any functionality that is only available to the owner.
602          *
603          * **ChainsGuard** 通过：放弃合约所有者权限
604          */
```

```
605     function renounceOwnership() public onlyOwner {
606         emit OwnershipTransferred(_owner, address(0));
607         _owner = address(0);
608     }
609
610     /**
611      * @dev Transfers ownership of the contract to a new account (`newOwner`).
612      * Can only be called by the current owner.
613      *
614      * **ChainsGuard** 通过：转移合约所有者权限
615      */
616     function transferOwnership(address newOwner) public onlyOwner {
617         _transferOwnership(newOwner);
618     }
619
620     /**
621      * @dev Transfers ownership of the contract to a new account (`newOwner`).
622      *
623      * **ChainsGuard** 通过：转移合约所有者权限
624      */
625     function _transferOwnership(address newOwner) internal {
```

```
626             require(newOwner != address(0), "Ownable: new owner is the zero address");
627             emit OwnershipTransferred(_owner, newOwner);
628             _owner = newOwner;
629         }
630 }
631
632 // **ChainsGuard** 通过：抽象角色定义模块
633 library Roles {
634
635     struct Role {
636         // **ChainsGuard** 角色抽象定义
637         mapping (address => bool) bearer;
638     }
639
640     /**
641      * @dev Give an account access to this role.
642      *
643      * **ChainsGuard** 通过：添加账户为指定角色
644      */
645     function add(Role storage role, address account) internal {
646         require(!has(role, account), "Roles: account already has role");
```

```
647             role.bearer[account] = true;
648         }
649
650         /**
651          * @dev Remove an account's access to this role.
652          * **ChainsGuard** 通过：移除账户指定角色
653          */
654         function remove(Role storage role, address account) internal {
655             require(has(role, account), "Roles: account does not have role");
656             role.bearer[account] = false;
657         }
658
659         /**
660          * @dev Check if an account has this role.
661          * @return bool
662          *
663          * **ChainsGuard** 通过：判断账户是否具有指定角色
664          */
665         function has(Role storage role, address account) internal view returns (bool) {
666             require(account != address(0), "Roles: account is the zero address");
667             return role.bearer[account];
```

```
668     }
669 }
670
671 // **ChainsGuard** 通过：暂停者角色模块
672 contract PauserRole is Context {
673     using Roles for Roles.Role;
674
675     event PauserAdded(address indexed account);
676     event PauserRemoved(address indexed account);
677     // **ChainsGuard** 定义暂停者角色
678     Roles.Role private _pausers;
679
680     constructor () internal {
681         // **ChainsGuard** 添加合约部署者为暂停者角色
682         _addPauser(_msgSender());
683     }
684
685     modifier onlyPauser() {
686         // **ChainsGuard** 通过：确保只有暂停者角色能够正确调用
687         require(isPauser(_msgSender()), "PauserRole: caller does not have the Pauser role");
688         _;
```

```
689     }
690
691     // **ChainsGuard** 通过：获取指定账户是否具有暂停者角色
692     function isPauser(address account) public view returns (bool) {
693         return _pausers.has(account);
694     }
695
696     // **ChainsGuard** 通过：添加指定账户为暂停者角色（只有暂停者能够调用）
697     function addPauser(address account) public onlyPauser {
698         _addPauser(account);
699     }
700
701     // **ChainsGuard** 通过：放弃自己的暂停者角色
702     function renouncePauser() public {
703         _removePauser(_msgSender());
704     }
705
706     // **ChainsGuard** 通过：添加暂停者角色
707     function _addPauser(address account) internal {
708         _pausers.add(account);
709         emit PauserAdded(account);
```

```
710     }
711
712     // **ChainsGuard** 通过：移除暂停者角色
713     function _removePauser(address account) internal {
714         _pausers.remove(account);
715         emit PauserRemoved(account);
716     }
717 }
718
719 // **ChainsGuard** 通过：合约暂停模块
720 contract Pausable is Context, PauserRole {
721     /**
722      * @dev Emitted when the pause is triggered by a pauser (`account`).
723      */
724     event Paused(address account);
725
726     /**
727      * @dev Emitted when the pause is lifted by a pauser (`account`).
728      */
729     event Unpaused(address account);
730     // **ChainsGuard** 暂停状态
```

```
731     bool private _paused;
732
733     /**
734      * @dev Initializes the contract in unpaused state. Assigns the Pauser role
735      * to the deployer.
736      */
737     constructor () internal {
738         // **ChainsGuard** 部署时状态为不暂停
739         _paused = false;
740     }
741
742     /**
743      * @dev Returns true if the contract is paused, and false otherwise.
744      *
745      * **ChainsGuard** 通过：获取暂停状态
746      */
747     function paused() public view returns (bool) {
748         return _paused;
749     }
750
751     /**
```

```
752      * @dev Modifier to make a function callable only when the contract is not paused.
753      *
754      * **ChainsGuard** 通过：确保未暂停时正确调用
755      */
756     modifier whenNotPaused() {
757         require(!_paused, "Pausable: paused");
758         _;
759     }
760
761     /**
762      * @dev Modifier to make a function callable only when the contract is paused.
763      *
764      * **ChainsGuard** 通过：确保暂停时正确调用
765      */
766     modifier whenPaused() {
767         require(_paused, "Pausable: not paused");
768         _;
769     }
770
771     /**
772      * @dev Called by a pauser to pause, triggers stopped state.
```

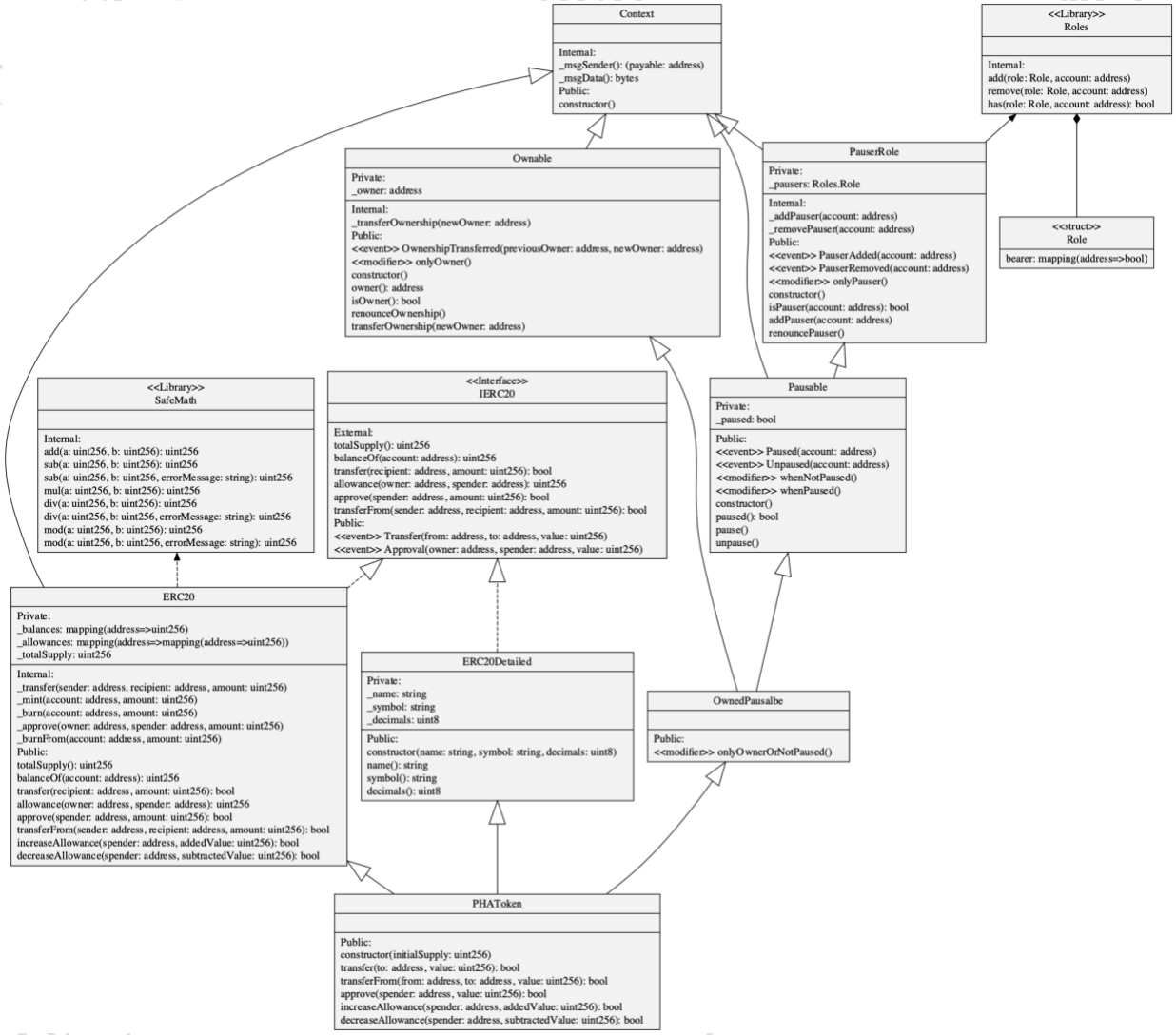
```
773         *
774         * **ChainsGuard** 通过：设置为暂停状态（只有暂停者能够调用）
775         */
776     function pause() public onlyPauser whenNotPaused {
777             _paused = true;
778             emit Paused(_msgSender());
779     }
780
781     /**
782         * @dev Called by a pauser to unpause, returns to normal state.
783         *
784         * **ChainsGuard** 通过：解锁暂停状态（只有暂停者能够调用）
785         */
786     function unpause() public onlyPauser whenPaused {
787             _paused = false;
788             emit Unpaused(_msgSender());
789     }
790 }
791
792 // **ChainsGuard** 通过：合约权限状态模块
793 contract OwnedPausalbe is Pausable, Ownable {
```

```
794     // **ChainsGuard** 通过：确保只有合约拥有者或合约未暂停状态能够正确调用
795     modifier onlyOwnerOrNotPaused() {
796         if (!isOwner()) {
797             require(!paused(), "Pausable: paused");
798         }
799         _;
800     }
801 }
802
803 contract PHAToken is ERC20, ERC20Detailed, OwnedPausalbe {
804     constructor(uint256 initialSupply) ERC20Detailed("Phala", "PHA", 18) public {
805         // **ChainsGuard** 初始化代币总量到部署者账户
806         _mint(msg.sender, initialSupply);
807         // **ChainsGuard** 暂停合约
808         pause();
809     }
810
811     // **ChainsGuard** 通过：转账（只有合约拥有者或合约未暂停状态能够正确调用）
812     function transfer(address to, uint256 value) public onlyOwnerOrNotPaused returns (bool) {
813         return super.transfer(to, value);
814     }
```

```
815
816     // **ChainsGuard** 通过：代理转账
817     function transferFrom(address from, address to, uint256 value) public returns (bool) {
818         // **ChainsGuard** 如果被代理账户不是合约拥有者则需要合约处于未暂停状态才能完成代理转账操作
819         if (from != owner()) {
820             require(!paused(), "Pausable: paused");
821         }
822         return super.transferFrom(from, to, value);
823     }
824
825     // **ChainsGuard** 通过：直接授权指定额度（只有合约拥有者或合约未暂停状态能够正确调用）
826     function approve(address spender, uint256 value) public onlyOwnerOrNotPaused returns (bool) {
827         return super.approve(spender, value);
828     }
829
830     // **ChainsGuard** 通过：增加授权额度（只有合约拥有者或合约未暂停状态能够正确调用）
831     function increaseAllowance(address spender, uint256 addedValue) public onlyOwnerOrNotPaused returns
    (bool) {
832         return super.increaseAllowance(spender, addedValue);
833     }
834
```

```
835     // **ChainsGuard** 通过：减少授权额度（只有合约拥有者或合约未暂停状态能够正确调用）
836     function decreaseAllowance(address spender, uint256 subtractedValue) public onlyOwnerOrNotPaused
    returns (bool) {
837         return super.decreaseAllowance(spender, subtractedValue);
838     }
839 }
```

6 Smart Contract UML



Appendix A. Explanation of Security Risk Status Levels

Security risk status statement	
1	**good status** The contract is in good running condition, and there are no or only sporadic low-risk security problems. At this time, as long as the existing security policy is maintained, the safety level requirements of the system can be met.
2	**warning status** There are some loopholes or security risks in the smart contract, which have not been used on a large scale. At this time, targeted reinforcement or improvement should be carried out according to the problems found in the evaluation, and then redeployment.
3	**serious status** Smart contract has been widely used. Serious loopholes or security problems that may seriously threaten the normal operation of the contract are found in the intelligent contract. At this time, measures should be taken immediately to redeploy the strengthened intelligent contract.
4	**emergency status** The tokens related to the intelligent contract have been opened for trading. Serious loopholes or security problems that may seriously threaten the normal operation of the contract have been found in the intelligent contract, which may cause serious damage to economic interests. At this point, should immediately stop the contract related token trading, immediately take measures to redeploy the strengthened intelligent contract.