

CERTIK VERIFICATION REPORT FOR SPENDCOIN



Request Date: 2019-03-29
Revision Date: 2019-03-31

Disclaimer

This Report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and SpendCoin(the “Company”), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the “Agreement”). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK’s prior written consent.

PASS

CERTIK believes this smart contract passes security qualifications to be listed on digital asset exchanges.

Mar 31, 2019



Summary

This audit report summarises the smart contract verification service requested by Spend-Coin. The goal of this security audit is to guarantee that the audited smart contracts are robust enough to avoid any potential security loopholes.

The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the time of the audit.

Type of Issues

CertiK smart label engine applied 100% covered formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116

Insecure Compiler Version	Com-	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Ran-	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120
“tx.origin” for authorization	for	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	to	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Variable	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Default	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables		Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure		The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features		Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables		Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

modifier validLock unused

Functions are protected by `onlyOwner`. So vulnerability is not a big issue. But modifier `validLock` is unnecessary at that point. Also the hardcoded `endtime` is not realistic.

For every issues found, CertiK categorizes them into 3 buckets based on its risk level:

- Critical: The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

- Medium: The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.
- Low: The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Source Code with CertiK Labels

File spendcoin.sol

```

1  pragma solidity ^0.4.18;
2
3
4  // -----
5
6  // Spendcoin Contract
7  //
8  // Symbol      : SPND
9  // Name        : Spendcoin
10 // Total supply: 2,000,000,000.000000000000000000
11 // Decimals    : 18
12 // Website     : https://spendcoin.org
13 // -----
14
15
16
17 // -----
18
19 // Safe maths
20
21 // -----
22
23 library SafeMath {
24     /*CTK SafeMath_add
25     @post __reverted == __has_overflow
26     @post __reverted == false -> c == a + b
27     @post msg == msg__post
28     @post (a + b < a) == __has_overflow
29     @post __addr_map == __addr_map__post
30     */
31     function add(uint a, uint b) internal pure returns (uint c) {
32
33         c = a + b;
34
35         require(c >= a);
36
37     }
38
39     /*CTK "SafeMath sub"
40     @post (a < b) == __reverted
41     @post !__reverted -> c == a - b
42     @post !__reverted -> !__has_overflow
43     */
44     function sub(uint a, uint b) internal pure returns (uint c) {
45
46         require(b <= a);
47
48         c = a - b;
49
50     }
51
52     /*CTK SafeMath_mul
53     @post __reverted == __has_overflow
54     @post __reverted == false -> c == a * b

```

```

55     @post a == 0 -> c == 0
56     @post msg == msg__post
57     @post (a > 0 && (a * b / a != b)) == __reverted
58     @post __addr_map == __addr_map__post
59     */
60     function mul(uint a, uint b) internal pure returns (uint c) {
61
62         c = a * b;
63
64         require(a == 0 || c / a == b);
65
66     }
67
68     /*@CTK "SafeMath div"
69     @post b != 0 -> !__reverted
70     @post !__reverted -> c == a / b
71     @post !__reverted -> !__has_overflow
72     */
73     function div(uint a, uint b) internal pure returns (uint c) {
74
75         require(b > 0);
76
77         c = a / b;
78
79     }
80
81 }
82
83
84
85 // -----
86
87 // ERC Token Standard #20 Interface
88
89 // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
90
91 // -----
92
93 contract ERC20Interface {
94
95     function totalSupply() public constant returns (uint);
96
97     function balanceOf(address tokenOwner) public constant returns (uint balance);
98
99     function allowance(address tokenOwner, address spender) public constant returns (
100         uint remaining);
101
102     function transfer(address to, uint tokens) public returns (bool success);
103
104     function approve(address spender, uint tokens) public returns (bool success);
105
106     function transferFrom(address from, address to, uint tokens) public returns (bool
107         success);
108
109     event Transfer(address indexed from, address indexed to, uint tokens);
110
111     event Approval(address indexed tokenOwner, address indexed spender, uint tokens);

```

```

111
112 }
113
114
115
116 // -----
117
118 // Owned contract
119
120 // -----
121
122 contract Owned {
123
124     address public owner;
125
126     /*@CTK Ownable
127     @post __post.owner == msg.sender
128     */
129     function Owned() public {
130
131         owner = msg.sender;
132
133     }
134
135
136     modifier onlyOwner {
137
138         require(msg.sender == owner);
139
140         _;
141
142     }
143
144 }
145
146 contract Tokenlock is Owned {
147
148     uint lockStartTime = 0; //time from when token will be locked
149     uint lockEndTime = 0;   //time from when token will be locked
150     uint8 isLocked = 0;     //flag indicates if token is locked
151
152     event Freezed(uint starttime, uint endtime);
153     event UnFreezed();
154
155     modifier validLock {
156         require(isLocked == 0 || (now < lockStartTime || now > lockEndTime));
157         _;
158     }
159
160     /*@CTK freezeTime
161     @tag assume_completion
162     @post owner == msg.sender
163     @post __post.isLocked == 1
164     @post __post.lockStartTime == _startTime
165     @post __post.lockEndTime == _endTime
166     */
167     function freezeTime(uint _startTime, uint _endTime) public onlyOwner {
168         isLocked = 1;

```



```

169         lockStartTime = _startTime;
170         lockEndTime = _endTime;
171
172         emit Freezed(lockStartTime, lockEndTime);
173     }
174
175     /*@CTK freeze
176     @tag assume_completion
177     @post owner == msg.sender
178     @post __post.isLocked == 1
179     @post __post.lockStartTime == 0
180     @post __post.lockEndTime == 90000000000
181     */
182     function freeze() public onlyOwner {
183         isLocked = 1;
184         lockStartTime = 0;
185         lockEndTime = 90000000000;
186
187         emit Freezed(lockStartTime, lockEndTime);
188     }
189
190     /*@CTK unfreeze
191     @tag assume_completion
192     @post owner == msg.sender
193     @post __post.isLocked == 0
194     @post __post.lockStartTime == 0
195     @post __post.lockEndTime == 0
196     */
197     function unfreeze() public onlyOwner {
198         isLocked = 0;
199         lockStartTime = 0;
200         lockEndTime = 0;
201
202         emit UnFreezed();
203     }
204 }
205
206
207 // -----
208
209 // ERC20 Token, with the addition of symbol, name and decimals and an
210
211 // initial fixed supply
212
213 // -----
214
215 contract Spendcoin is ERC20Interface, Tokenlock {
216
217     using SafeMath for uint;
218
219
220     string public symbol;
221
222     string public name;
223
224     uint8 public decimals;
225
226     uint public _totalSupply;

```

```

227
228
229 mapping(address => uint) balances;
230
231 mapping(address => mapping(address => uint)) allowed;
232
233
234
235 // -----
236
237 // Constructor
238
239 // -----
240 /*@CTK Spendcoin
241   @post __post.symbol == "SPND"
242   @post __post.name == "Spendcoin"
243   @post __post.decimals == 18
244   @post __post.balances[owner] == __post._totalSupply
245 */
246 function Spendcoin() public {
247
248     symbol = "SPND";
249
250     name = "Spendcoin";
251
252     decimals = 18;
253
254     _totalSupply = 2000000000 * 10**uint(decimals);
255
256     balances[owner] = _totalSupply;
257
258     emit Transfer(address(0), owner, _totalSupply);
259
260 }
261
262
263
264 // -----
265
266 // Total supply
267
268 // -----
269 /*@CTK totalSupply
270   @post __return == _totalSupply - balances[address(0)]
271 */
272 function totalSupply() public constant returns (uint) {
273
274     return _totalSupply - balances[address(0)];
275
276 }
277
278
279
280 // -----
281
282 // Get the token balance for account 'tokenOwner'
283
284 // -----

```

```

285  /*@CTK balanceOf
286      @post balance == balances[tokenOwner]
287  */
288  function balanceOf(address tokenOwner) public constant returns (uint balance) {
289
290      return balances[tokenOwner];
291
292  }
293
294
295
296  // -----
297
298  // Transfer the balance from token owner's account to 'to' account
299
300  // - Owner's account must have sufficient balance to transfer
301
302  // - 0 value transfers are allowed
303
304  // -----
305  /*@CTK transfer
306      @tag assume_completion
307      @pre msg.sender != to
308      @post __post.balances[msg.sender] == balances[msg.sender] - tokens
309      @post __post.balances[to] == balances[to] + tokens
310  */
311  function transfer(address to, uint tokens) public returns (bool success) {
312
313      balances[msg.sender] = balances[msg.sender].sub(tokens);
314
315      balances[to] = balances[to].add(tokens);
316
317      emit Transfer(msg.sender, to, tokens);
318
319      return true;
320
321  }
322
323
324
325  // -----
326
327  // Token owner can approve for 'spender' to transferFrom(...) 'tokens'
328
329  // from the token owner's account
330
331  //
332
333  // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
334
335  // recommends that there are no checks for the approval double-spend attack
336
337  // as this should be implemented in user interfaces
338
339  // -----
340  /*@CTK approve
341      @post __post.allowed[msg.sender][spender] == tokens
342  */

```

```

343 function approve(address spender, uint tokens) public returns (bool success) {
344
345     allowed[msg.sender][spender] = tokens;
346
347     emit Approval(msg.sender, spender, tokens);
348
349     return true;
350
351 }
352
353
354
355 // -----
356
357 // Transfer 'tokens' from the 'from' account to the 'to' account
358
359 //
360
361 // The calling account must already have sufficient tokens approve(...)-d
362
363 // for spending from the 'from' account and
364
365 // - From account must have sufficient balance to transfer
366
367 // - Spender must have sufficient allowance to transfer
368
369 // - 0 value transfers are allowed
370
371 // -----
372 /*@CTK transferFrom
373    @tag assume_completion
374    @pre from != to
375    @post __post.balances[from] == balances[from] - tokens
376    @post __post.balances[to] == balances[to] + tokens
377    @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
378 */
379 function transferFrom(address from, address to, uint tokens) public returns (bool
    success) {
380
381     balances[from] = balances[from].sub(tokens);
382
383     allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
384
385     balances[to] = balances[to].add(tokens);
386
387     emit Transfer(from, to, tokens);
388
389     return true;
390
391 }
392
393
394
395 // -----
396
397 // Returns the amount of tokens approved by the owner that can be
398
399 // transferred to the spender's account

```

```

400
401 // -----
402 /*@CTK allowance
403   @post remaining == allowed[tokenOwner][spender]
404 */
405 function allowance(address tokenOwner, address spender) public constant returns (
406   uint remaining) {
407     return allowed[tokenOwner][spender];
408 }
409
410
411 // -----
412 // Do accept ETH
413 // -----
414
415 function () public payable {
416 }
417
418 // -----
419 // Owner can withdraw ether if token received.
420 // -----
421 function withdraw() public onlyOwner returns (bool result) {
422   address tokenaddress = this;
423   // CTK: owner.send(this.balance)
424   return owner.send(tokenaddress.balance);
425 }
426
427 // -----
428 // Owner can transfer out any accidentally sent ERC20 tokens
429 // -----
430
431 function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner
432   returns (bool success) {
433     return ERC20Interface(tokenAddress).transfer(owner, tokens);
434 }
435
436 }

```

How to read

Detail for Request 1

transferFrom to same address


Verification date	 20, Oct 2018
Verification timespan	 395.38 ms

CERTIK label location	Line 30-34 in File howtoread.sol
-----------------------	----------------------------------

CERTIK label	30	/*@CTK FAIL "transferFrom to same address"
	31	@tag assume_completion
	32	@pre from == to
	33	@post __post.allowed[from][msg.sender] ==
	34	*/

Raw code location	Line 35-41 in File howtoread.sol
-------------------	----------------------------------

Raw code	35	function transferFrom(address from, address to
) {
	36	balances[from] = balances[from].sub(tokens
	37	allowed[from][msg.sender] = allowed[from][
	38	balances[to] = balances[to].add(tokens);
	39	emit Transfer(from, to, tokens);
	40	return true;
	41	}

Counterexample	 This code violates the specification
----------------	--

Initial environment	1	Counter Example:
	2	Before Execution:
	3	Input = {
	4	from = 0x0
	5	to = 0x0
	6	tokens = 0x6c
	7	}
	8	This = 0
	52	}
	53	balance: 0x0
	54	}
	55	}
	56	
Post environment	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c

Static Analysis Request

TIMESTAMP_DEPENDENCY

Line 156 in File spendcoin.sol

```
156     require(isLocked == 0 || (now < lockStartTime || now > lockEndTime));
```

! "now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 156 in File spendcoin.sol

```
156     require(isLocked == 0 || (now < lockStartTime || now > lockEndTime));
```

! "now" can be influenced by minors to some degree

Formal Verification Request 1

SafeMath.add

📅 31, Mar 2019

🕒 17.97 ms

Line 24-30 in File spendcoin.sol

```
24  /*@CTK SafeMath_add
25     @post __reverted == __has_overflow
26     @post __reverted == false -> c == a + b
27     @post msg == msg__post
28     @post (a + b < a) == __has_overflow
29     @post __addr_map == __addr_map__post
30  */
```

Line 31-37 in File spendcoin.sol

```
31  function add(uint a, uint b) internal pure returns (uint c) {
32
33      c = a + b;
34
35      require(c >= a);
36
37  }
```

✅ The code meets the specification

Formal Verification Request 2

SafeMath.sub

📅 31, Mar 2019

🕒 14.26 ms

Line 39-43 in File spendcoin.sol

```
39  /*@CTK "SafeMath sub"
40     @post (a < b) == __reverted
41     @post !__reverted -> c == a - b
42     @post !__reverted -> !__has_overflow
43  */
```

Line 44-50 in File spendcoin.sol

```
44  function sub(uint a, uint b) internal pure returns (uint c) {
45
46      require(b <= a);
47
48      c = a - b;
49
50  }
```

✅ The code meets the specification

Formal Verification Request 3

SafeMath_mul

📅 31, Mar 2019

🕒 126.06 ms

Line 52-59 in File spendcoin.sol

```
52  /*@CTK SafeMath_mul
53      @post __reverted == __has_overflow
54      @post __reverted == false -> c == a * b
55      @post a == 0 -> c == 0
56      @post msg == msg__post
57      @post (a > 0 && (a * b / a != b)) == __reverted
58      @post __addr_map == __addr_map__post
59  */
```

Line 60-66 in File spendcoin.sol

```
60  function mul(uint a, uint b) internal pure returns (uint c) {
61
62      c = a * b;
63
64      require(a == 0 || c / a == b);
65
66  }
```

✅ The code meets the specification

Formal Verification Request 4

SafeMath div

📅 31, Mar 2019

🕒 14.81 ms

Line 68-72 in File spendcoin.sol

```
68  /*@CTK "SafeMath div"
69      @post b != 0 -> !__reverted
70      @post !__reverted -> c == a / b
71      @post !__reverted -> !__has_overflow
72  */
```

Line 73-79 in File spendcoin.sol


```
73  function div(uint a, uint b) internal pure returns (uint c) {
74
75      require(b > 0);
76
77      c = a / b;
78
79  }
```

✅ The code meets the specification

Formal Verification Request 5

Ownable

 31, Mar 2019

 6.0 ms

Line 126-128 in File spendcoin.sol

```
126  /*@CTK Ownable
127      @post __post.owner == msg.sender
128  */
```

Line 129-133 in File spendcoin.sol


```
129  function Owned() public {
130
131      owner = msg.sender;
132
133  }
```

 The code meets the specification

Formal Verification Request 6

freezeTime

 31, Mar 2019

 26.1 ms

Line 160-166 in File spendcoin.sol

```
160  /*@CTK freezeTime
161      @tag assume_completion
162      @post owner == msg.sender
163      @post __post.isLocked == 1
164      @post __post.lockStartTime == _startTime
165      @post __post.lockEndTime == _endTime
166  */
```

Line 167-173 in File spendcoin.sol

```
167  function freezeTime(uint _startTime, uint _endTime) public onlyOwner {
168      isLocked = 1;
169      lockStartTime = _startTime;
170      lockEndTime = _endTime;
171
172      emit Freezed(lockStartTime, lockEndTime);
173  }
```

 The code meets the specification

Formal Verification Request 7

freeze

📅 31, Mar 2019

🕒 23.95 ms

Line 175-181 in File spendcoin.sol

```
175  /*@CTK freeze
176      @tag assume_completion
177      @post owner == msg.sender
178      @post __post.isLocked == 1
179      @post __post.lockStartTime == 0
180      @post __post.lockEndTime == 90000000000
181  */
```

Line 182-188 in File spendcoin.sol

```
182  function freeze() public onlyOwner {
183      isLocked = 1;
184      lockStartTime = 0;
185      lockEndTime = 90000000000;
186
187      emit Freezed(lockStartTime, lockEndTime);
188  }
```

✅ The code meets the specification

Formal Verification Request 8

unfreeze

📅 31, Mar 2019

🕒 26.4 ms

Line 190-196 in File spendcoin.sol

```
190  /*@CTK unfreeze
191      @tag assume_completion
192      @post owner == msg.sender
193      @post __post.isLocked == 0
194      @post __post.lockStartTime == 0
195      @post __post.lockEndTime == 0
196  */
```

Line 197-203 in File spendcoin.sol

```
197  function unfreeze() public onlyOwner {
198      isLocked = 0;
199      lockStartTime = 0;
200      lockEndTime = 0;
201
202      emit UnFreezed();
203  }
```

✅ The code meets the specification

Formal Verification Request 9

Spendcoin

📅 31, Mar 2019

🕒 34.86 ms

Line 240-245 in File spendcoin.sol

```
240  /*@CTK Spendcoin
241      @post __post.symbol == "SPND"
242      @post __post.name == "Spendcoin"
243      @post __post.decimals == 18
244      @post __post.balances[owner] == __post._totalSupply
245  */
```

Line 246-260 in File spendcoin.sol

```
246  function Spendcoin() public {
247
248      symbol = "SPND";
249
250      name = "Spendcoin";
251
252      decimals = 18;
253
254      _totalSupply = 2000000000 * 10**uint(decimals);
255
256      balances[owner] = _totalSupply;
257
258      emit Transfer(address(0), owner, _totalSupply);
259
260  }
```

✅ The code meets the specification

Formal Verification Request 10

totalSupply

📅 31, Mar 2019

🕒 8.73 ms

Line 269-271 in File spendcoin.sol

```
269  /*@CTK totalSupply
270      @post __return == _totalSupply - balances[address(0)]
271  */
```

Line 272-276 in File spendcoin.sol


```
272  function totalSupply() public constant returns (uint) {
273
274      return _totalSupply - balances[address(0)];
275
276  }
```

✅ The code meets the specification

Formal Verification Request 11

balanceOf

 31, Mar 2019

 7.22 ms

Line 285-287 in File spendcoin.sol

```
285  /*@CTK balanceOf
286      @post balance == balances[tokenOwner]
287  */
```

Line 288-292 in File spendcoin.sol


```
288  function balanceOf(address tokenOwner) public constant returns (uint balance) {
289
290      return balances[tokenOwner];
291
292  }
```

 The code meets the specification

Formal Verification Request 12

transfer

 31, Mar 2019

 110.56 ms

Line 305-310 in File spendcoin.sol

```
305  /*@CTK transfer
306      @tag assume_completion
307      @pre msg.sender != to
308      @post __post.balances[msg.sender] == balances[msg.sender] - tokens
309      @post __post.balances[to] == balances[to] + tokens
310  */
```

Line 311-321 in File spendcoin.sol

```
311  function transfer(address to, uint tokens) public returns (bool success) {
312
313      balances[msg.sender] = balances[msg.sender].sub(tokens);
314
315      balances[to] = balances[to].add(tokens);
316
317      emit Transfer(msg.sender, to, tokens);
318
319      return true;
320
321  }
```

 The code meets the specification

Formal Verification Request 13

approve

📅 31, Mar 2019

🕒 15.23 ms

Line 340-342 in File spendcoin.sol

```
340  /*@CTK approve
341      @post __post.allowed[msg.sender][spender] == tokens
342  */
```

Line 343-351 in File spendcoin.sol

```
343  function approve(address spender, uint tokens) public returns (bool success) {
344
345      allowed[msg.sender][spender] = tokens;
346
347      emit Approval(msg.sender, spender, tokens);
348
349      return true;
350
351  }
```

✅ The code meets the specification

Formal Verification Request 14

transferFrom

📅 31, Mar 2019

🕒 178.42 ms

Line 372-378 in File spendcoin.sol

```
372  /*@CTK transferFrom
373      @tag assume_completion
374      @pre from != to
375      @post __post.balances[from] == balances[from] - tokens
376      @post __post.balances[to] == balances[to] + tokens
377      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
378  */
```

Line 379-391 in File spendcoin.sol

```
379  function transferFrom(address from, address to, uint tokens) public returns (bool
380      success) {
381
382      balances[from] = balances[from].sub(tokens);
383
384      allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
385
386      balances[to] = balances[to].add(tokens);
387
388      emit Transfer(from, to, tokens);
```


```
389     return true;
390
391 }
```

✓ The code meets the specification

Formal Verification Request 15

allowance

 31, Mar 2019

 7.11 ms

Line 402-404 in File spendcoin.sol

```
402  /*@CTK allowance
403     @post remaining == allowed[tokenOwner][spender]
404  */
```

Line 405-409 in File spendcoin.sol

```
405  function allowance(address tokenOwner, address spender) public constant returns (
406      uint remaining) {
407      return allowed[tokenOwner][spender];
408
409  }
```

✓ The code meets the specification