CertiK Audit Report For ThunderCore

Request Date: 2019-05-01
Revision Date: 2019-05-05
Platform Name:

CERTIK

Contents

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 ThunderCore(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.

Exective Summary

This report has been prepared as product of the Smart Contract Audit request by ThunderCore. This audit was conducted to discover issues and vulnerabilities in the source code of ThunderCore's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

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

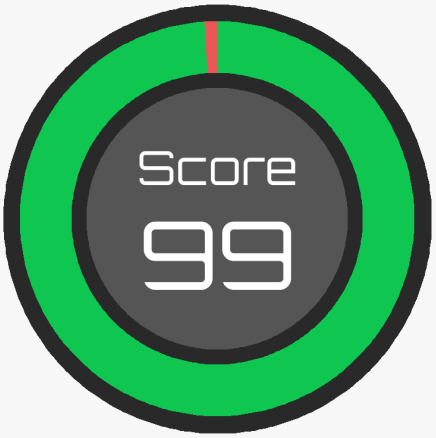
- Critical: The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.
- Medium: The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.
- Low: The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

Testing Summary

PASS

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

May 05, 2019



Type of Issues

CertiK smart label engine applied 100% coveraged 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	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	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	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	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

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Formal Verification Results

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
53             balance: 0x0
54         }
55     }
56
```

Post environment

```
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

endow

📅 05, May 2019
⏱ 48.75 ms

Line 32-35 in File Hodl.sol

```
32  /*@CTK endow
33    @tag assume_completion
34    @post __post.endowment == endowment + msg.value
35   */
```

Line 36-40 in File Hodl.sol

```
36  function endow() public payable {
37    // allow anyone to transfer funds to this contract
38    endowment = endowment.add(msg.value);
39    emit Endowed(msg.sender, msg.value);
40  }
```

✅ The code meets the specification

Formal Verification Request 2

withdrawEndowment

📅 05, May 2019
⏱ 179.04 ms

Line 42-46 in File Hodl.sol

```
42  /*@CTK withdrawEndowment
43    @tag assume_completion
44    @post _owner == msg.sender
45    @post __post.endowment == endowment - amount
46   */
```

Line 47-52 in File Hodl.sol

```
47  function withdrawEndowment(uint256 amount) public onlyOwner {
48    require(amount <= endowment, 'Withdrawing amount is larger than endowment.');
49    Ownable.owner().transfer(amount);
50    endowment = endowment.sub(amount);
51    emit WithdrawnEndowment(msg.sender, amount);
52  }
```

✅ The code meets the specification

Formal Verification Request 3

getTime

📅 05, May 2019
⏱ 5.48 ms

Line 80-82 in File Hodl.sol

```
/*@CTK getTime
  @post __return == now
 */
```

Line 83-85 in File Hodl.sol

```
function getTime() public view returns(uint256) {
  return now; // solium-disable-line security/no-block-members
}
```

The code meets the specification

Formal Verification Request 4

interestRate

05, May 2019
56.6 ms

Line 106-113 in File Hodl.sol

```
/*@CTK interestRate
  @post duration == Duration.OneDay -> (__return == 1 && __return1 == 10000)
  @post duration == Duration.OneWeek -> (__return == 1 && __return1 == 1000)
  @post duration == Duration.OneMonth -> (__return == 7 && __return1 == 1000)
  @post duration == Duration.OneQuarter -> (__return == 25 && __return1 == 1000)
  @post duration == Duration.HalfYear -> (__return == 1 && __return1 == 10)
  @post duration == Duration.OneYear -> (__return == 3 && __return1 == 10)
 */
```

Line 114-131 in File Hodl.sol

```
function interestRate(Duration duration) private pure returns(uint256, uint256) {
  if (duration == Duration.OneDay) {
    return (1, 10000);
  } else if (duration == Duration.OneWeek) {
    return (1, 1000);
  } else if (duration == Duration.OneMonth) {
    return (7, 1000);
  } else if (duration == Duration.OneQuarter) {
    return (25, 1000);
  } else if (duration == Duration.HalfYear) {
    return (1, 10);
  } else if (duration == Duration.OneYear) {
    return (3, 10);
  }

  revert('Invalid duration');
}
```

The code meets the specification

Formal Verification Request 5

toDays

05, May 2019
50.98 ms

Line 133-140 in File Hodl.sol

```
133  /*@CTK toDays
134    @post duration == Duration.OneDay -> __return == 1
135    @post duration == Duration.OneWeek -> __return == 7
136    @post duration == Duration.OneMonth -> __return == 30
137    @post duration == Duration.OneQuarter -> __return == 90
138    @post duration == Duration.HalfYear -> __return == 180
139    @post duration == Duration.OneYear -> __return == 365
140   */
```

Line 141-158 in File Hodl.sol

```
141  function toDays(Duration duration) private pure returns(uint256) {
142    if (duration == Duration.OneDay) {
143      return 1;
144    } else if (duration == Duration.OneWeek) {
145      return 7;
146    } else if (duration == Duration.OneMonth) {
147      return 30;
148    } else if (duration == Duration.OneQuarter) {
149      return 90;
150    } else if (duration == Duration.HalfYear) {
151      return 180;
152    } else if (duration == Duration.OneYear) {
153      return 365;
154    }
155
156    revert('Invalid duration');
157  }
```

The code meets the specification

Formal Verification Request 6

getDeposits__Generated

05, May 2019
123.85 ms

(Loop) Line 206-213 in File Hodl.sol

```
206    /*@CTK getDeposits
207      @inv i <= len
208      @inv i >= 1 -> packed[i - 1][0] == depositRecords[target][i - 1].startTime
209      @inv i >= 1 -> packed[i - 1][1] == depositRecords[target][i - 1].principal
210      @inv i >= 1 -> packed[i - 1][3] == depositRecords[target][i - 1].collected
211      @post i == len
212      @post !__should_return
213     */
```

(Loop) Line 206-220 in File Hodl.sol

```
206    /*@CTK getDeposits
207      @inv i <= len
208      @inv i >= 1 -> packed[i - 1][0] == depositRecords[target][i - 1].startTime
209      @inv i >= 1 -> packed[i - 1][1] == depositRecords[target][i - 1].principal
210      @inv i >= 1 -> packed[i - 1][3] == depositRecords[target][i - 1].collected
```

```
211       @post i == len
212       @post !__should_return
213      */
214     for (uint256 i = 0; i < len; i++) {
215       DepositRecord memory certificate = depositRecords[target][i];
216       packed[i][0] = certificate.startTime;
217       packed[i][1] = certificate.principal;
218       packed[i][2] = uint256(certificate.duration);
219       packed[i][3] = certificate.collected;
220     }
```

The code meets the specification

Formal Verification Request 7

getTime

05, May 2019
6.1 ms

Line 9-11 in File HodlWithFakeTime.sol

```
9   /*@CTK getTime
10    @post __return == fakeTime
11   */
```

Line 12-14 in File HodlWithFakeTime.sol

```
12  function getTime() public view returns(uint256) {
13    return fakeTime;
14  }
```

The code meets the specification

Formal Verification Request 8

setTime

05, May 2019
6.03 ms

Line 16-18 in File HodlWithFakeTime.sol

```
16  /*@CTK setTime
17    @post __post.fakeTime == f
18   */
```

Line 19-21 in File HodlWithFakeTime.sol

```
19  function setTime(uint256 f) public {
20    fakeTime = f;
21  }
```

The code meets the specification

Formal Verification Request 9

Ownable

📅 05, May 2019
⏱ 5.84 ms

Line 17-19 in File Ownable.sol

```
17    /*@CTK Ownable
18      @post __post._owner == msg.sender
19     */
```

Line 20-23 in File Ownable.sol

```
20    constructor () internal {
21        _owner = msg.sender;
22        emit OwnershipTransferred(address(0), _owner);
23    }
```

✅ The code meets the specification

Formal Verification Request 10

owner

📅 05, May 2019
⏱ 5.69 ms

Line 28-30 in File Ownable.sol

```
28  /*@CTK owner
29    @post __return == _owner
30   */
```

Line 31-33 in File Ownable.sol

```
31    function owner() public view returns (address) {
32        return _owner;
33    }
```

✅ The code meets the specification

Formal Verification Request 11

isOwner

📅 05, May 2019
⏱ 6.13 ms

Line 46-48 in File Ownable.sol

```
46  /*@CTK isOwner
47    @post __return == (msg.sender == _owner)
48   */
```

Line 49-51 in File Ownable.sol

```
49      function isOwner() public view returns (bool) {
50          return msg.sender == _owner;
51      }
```

The code meets the specification

Formal Verification Request 12

renounceOwnership

05, May 2019
26.83 ms

Line 59-63 in File Ownable.sol

```
59    /*@CTK renounceOwnership
60      @tag assume_completion
61      @post _owner == msg.sender
62      @post __post._owner == address(0)
63     */
```

Line 64-67 in File Ownable.sol

```
64      function renounceOwnership() public onlyOwner {
65          emit OwnershipTransferred(_owner, address(0));
66          _owner = address(0);
67      }
```

The code meets the specification

Formal Verification Request 13

transferOwnership

05, May 2019
61.88 ms

Line 73-76 in File Ownable.sol

```
73    /*@CTK transferOwnership
74      @tag assume_completion
75      @post _owner == msg.sender
76     */
```

Line 77-79 in File Ownable.sol

```
77      function transferOwnership(address newOwner) public onlyOwner {
78          _transferOwnership(newOwner);
79      }
```

The code meets the specification

Formal Verification Request 14

_transferOwnership

📅 05, May 2019
⏱ 1.19 ms

Line 85-89 in File Ownable.sol

```
85  /*@CTK _transferOwnership
86    @tag assume_completion
87    @post newOwner != address(0)
88    @post __post._owner == newOwner
89   */
```

Line 90-94 in File Ownable.sol

```
90    function _transferOwnership(address newOwner) internal {
91        require(newOwner != address(0));
92        emit OwnershipTransferred(_owner, newOwner);
93        _owner = newOwner;
94    }
```

✅ The code meets the specification

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File Hodl.sol

```
1 pragma solidity >=0.4.25 < 0.5.0;
```

ℹ Only these compiler versions are safe to compile your code: 0.4.25

TIMESTAMP_DEPENDENCY

Line 84 in File Hodl.sol

```
84     return now; // solium-disable-line security/no-block-members
```

❗ ”now” can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File HodlWithFakeTime.sol

```
1 pragma solidity >=0.4.25 < 0.5.0;
```

ℹ Only these compiler versions are safe to compile your code: 0.4.25

INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

```
1 pragma solidity ^0.5.0;
```

ℹ Only these compiler versions are safe to compile your code: 0.5.0, 0.5.1, 0.5.2, 0.5.3, 0.5.4, 0.5.6

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **Hodl.sol** `444db6df9a27e28dab7bd84db8eb59b7695f37c3f848d6066e96ec12043ec39f`
- **HodlWithFakeTime.sol** `c284afd45ce8dab3aabae52b8a1d3c6736f2120b60780448d7c587f54ca2b2e9`
- **Migrations.sol** `1c4e30fd3aa765cb0ee259a29dead71c1c99888dcc7157c25df3405802cf5b09`

Summary

CertiK team is invited by ThunderCore to audit the design and implementations of its to be released financial product smart contract called hodl, and the source code has been analyzed under different perspectives and with different tools such as CertiK formal verification checkings as well as manual reviews by smart contract experts. We have been actively interacting with ThunderCore engineers when there was any potential loopholes or recommended design changes during the audit process, and ThunderCore team has been actively giving us updates for the source code and feedback about the business logic.

In general, the audit process went very effectively, as a result of the high code quality, and good practices works done by the ThunderCore team. The business logic and intentions are well-defined, straight-forward, and following industrial standards. Corresponding unit tests were added to cover the possible scenarios, and potential edge cases, well in the meantime we recommend to have more detailed documents describing the product and rules. As summary, `Hodl.sol` smart contract is designing with the business model of a saving-account-alike, where people can deposit principle and lock with a period of time, and finally return in interest.

CertiK team did not find any potential security risks from the smart contract, but again we urge users who plan to purchase such financial product to have a fully understanding before taking further actions. The contract leans on appropriate standards with minimal storage, and cost consumption on each function invocation, in order to fulfill the business requirements. The proper intervention mechanism helps to minimize and prevent human errors. We conclude that `Hodl.sol` smart contract shall launch in a well-tested and secure state, is not vulnerable to any known antipatterns or bugs, and the risk is likely very low.

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

Hodl.sol

- **deposit(uint256 durationInt)** – durationInt represents the index of the enum defined in the contract, use Duration is better practice there as Solidity will convert into uint8 (also aligns with the coding style from the rest of the code).

- **collect()** – The `depositRecords` array is updated/truncated via a smart but a bit tricky solution, we suggest to have more comments describing how it works, and make the variable namings more human readable. Also, we suggest to have an illustration for how interests were calculated for end users, i.e. if a user choose to buy a 1-day period deposit but withdraw on day 4 will only get one day interest (instead of 3).

Source Code with CertiK Labels

File Hodl.sol

```
1  pragma solidity >=0.4.25 < 0.5.0;
2
3  import 'openzeppelin-solidity/contracts/math/SafeMath.sol';
4  import 'openzeppelin-solidity/contracts/ownership/Ownable.sol';
5
6
7  contract Hodl is Ownable {
8    enum Duration {
9      OneDay, OneWeek, OneMonth, OneQuarter, HalfYear, OneYear
10   }
11
12   struct DepositRecord {
13     uint256 startTime;
14     uint256 principal;
15     Duration duration;
16     uint256 collected;
17   }
18
19   mapping(address => DepositRecord[]) public depositRecords;
20   uint256 public endowment;
21   using SafeMath for uint256;
22
23   event Collected(address indexed account, uint256 amount, uint256 time);
24   event Deposited(address indexed account, uint256 amount, uint256 time, Duration
         duration);
25   event Endowed(address indexed account, uint256 amount);
26   event Closed(address indexed account, uint256 amount, uint256 time, Duration
         duration);
27   event WithdrawnEndowment(address indexed account, uint256 amount);
28
29   uint8 constant MAX_ACTIVE_DEPOSITS = 16;
30   uint256 constant MAX_DEPOSIT_AMOUNT = 1000000 ether;
31
32   /*@CTK endow
33     @tag assume_completion
34     @post __post.endowment == endowment + msg.value
35    */
36   function endow() public payable {
37     // allow anyone to transfer funds to this contract
38     endowment = endowment.add(msg.value);
39     emit Endowed(msg.sender, msg.value);
40   }
41
42   /*@CTK withdrawEndowment
43     @tag assume_completion
44     @post _owner == msg.sender
45     @post __post.endowment == endowment - amount
46    */
47   function withdrawEndowment(uint256 amount) public onlyOwner {
48     require(amount <= endowment, 'Withdrawing amount is larger than endowment.');
49     Ownable.owner().transfer(amount);
50     endowment = endowment.sub(amount);
51     emit WithdrawnEndowment(msg.sender, amount);
52   }
```

```
53
54  /*CTK deposit
55    @tag assume_completion
56    @post depositRecords[msg.sender].length < MAX_ACTIVE_DEPOSITS
57   */
58  function deposit(uint256 durationInt) public payable {
59    require(depositRecords[msg.sender].length < MAX_ACTIVE_DEPOSITS, 'Exceeded max
          active deposits');
60    require(getTotalDeposits(msg.sender).add(msg.value) <= MAX_DEPOSIT_AMOUNT, '
          Exceeded max deposit amount');
61    require(durationInt <= uint256(Duration.OneYear), 'Invalid duration');
62    require(msg.value > 0, 'Need value to deposit');
63
64    Duration duration = Duration(durationInt);
65    DepositRecord memory dr;
66    dr.startTime = getTime();
67    dr.duration = duration;
68    dr.principal = msg.value;
69    dr.collected = 0;
70
71    uint256 totalInterest = interest(dr.principal, dr.duration);
72
73    require(totalInterest <= endowment, 'Insufficient endowment');
74
75    endowment = endowment.sub(totalInterest);
76    depositRecords[msg.sender].push(dr);
77    emit Deposited(msg.sender, dr.principal, dr.startTime, dr.duration);
78  }
79
80  /*@CTK getTime
81    @post __return == now
82   */
83  function getTime() public view returns(uint256) {
84    return now; // solium-disable-line security/no-block-members
85  }
86
87  /*CTK dayInterest
88    @pre duration == Duration.OneDay
89    @post __return
90   */
91  function dayInterest(uint256 principal, Duration duration) public pure returns(
        uint256) {
92    uint256 dates = toDays(duration);
93    return interest(principal, duration).div(dates);
94  }
95
96  function interest(uint256 principal, Duration duration) public pure returns(uint256)
        {
97    (uint256 mul, uint256 div) = interestRate(duration);
98    (mul, div) = interestRate(duration);
99    return principal.mul(mul).div(div);
100 }
101
102 /*@CTK interestRate
103   @post duration == Duration.OneDay -> (__return == 1 && __return1 == 10000)
104   @post duration == Duration.OneWeek -> (__return == 1 && __return1 == 1000)
105   @post duration == Duration.OneMonth -> (__return == 7 && __return1 == 1000)
106   @post duration == Duration.OneQuarter -> (__return == 25 && __return1 == 1000)
```

```
107     @post duration == Duration.HalfYear -> (__return == 1 && __return1 == 10)
108     @post duration == Duration.OneYear -> (__return == 3 && __return1 == 10)
109    */
110   function interestRate(Duration duration) private pure returns(uint256, uint256) {
111     if (duration == Duration.OneDay) {
112       return (1, 10000);
113     } else if (duration == Duration.OneWeek) {
114       return (1, 1000);
115     } else if (duration == Duration.OneMonth) {
116       return (7, 1000);
117     } else if (duration == Duration.OneQuarter) {
118       return (25, 1000);
119     } else if (duration == Duration.HalfYear) {
120       return (1, 10);
121     } else if (duration == Duration.OneYear) {
122       return (3, 10);
123     }
124
125     revert('Invalid duration');
126   }
127
128   /*@CTK toDays
129     @post duration == Duration.OneDay -> __return == 1
130     @post duration == Duration.OneWeek -> __return == 7
131     @post duration == Duration.OneMonth -> __return == 30
132     @post duration == Duration.OneQuarter -> __return == 90
133     @post duration == Duration.HalfYear -> __return == 180
134     @post duration == Duration.OneYear -> __return == 365
135    */
136   function toDays(Duration duration) private pure returns(uint256) {
137     if (duration == Duration.OneDay) {
138       return 1;
139     } else if (duration == Duration.OneWeek) {
140       return 7;
141     } else if (duration == Duration.OneMonth) {
142       return 30;
143     } else if (duration == Duration.OneQuarter) {
144       return 90;
145     } else if (duration == Duration.HalfYear) {
146       return 180;
147     } else if (duration == Duration.OneYear) {
148       return 365;
149     }
150
151     revert('Invalid duration');
152   }
153
154   //
155   function collect() external {
156     uint256 s0; // principal plus interest for one deposit record
157     uint256 i0; // accrued interest for one deposit record
158     uint256 c0; // collectible funds for one deposit record
159     uint256 c; // total collectible for sender
160
161     // calculate how much to collect
162     address sender = msg.sender;
163     DepositRecord[] storage drlist = depositRecords[sender];
164     uint256 j = 0;
```

```
165     for (uint256 i = 0; i < drlist.length; i++) {
166       DepositRecord storage dr = drlist[i];
167       uint256 daysPassed = getTime().sub(dr.startTime).div(1 days);
168       uint256 durationDays = toDays(dr.duration);
169       if (daysPassed >= durationDays) {
170         s0 = interest(dr.principal, dr.duration).add(dr.principal);
171         c0 = s0.sub(dr.collected);
172         // Not updating dr.collected because drlist[i] would be either overwritten or "
              removed" by reducing drlist.length
173         emit Closed(sender, dr.principal, getTime(), dr.duration);
174       } else {
175         i0 = dayInterest(dr.principal, dr.duration).mul(daysPassed);
176         c0 = i0.sub(dr.collected);
177         if (i != j) {
178           drlist[j] = dr;
179         }
180         drlist[j].collected = i0;
181         j++;
182       }
183       c = c.add(c0);
184     }
185
186     // Reducing the length performs an implicit delete on each of the removed elements
          .
187     drlist.length = j;
188
189     // transfer
190     sender.transfer(c);
191
192     emit Collected(sender, c, getTime());
193   }
194
195   function getDeposits(address target) external view returns (
196     uint256 blockTime, uint256[4][]) {
197     uint256 len = depositRecords[target].length;
198     uint256[4][] memory packed = new uint256[4][](len);
199
200     /*@CTK getDeposits
201       @inv i <= len
202       @inv i >= 1 -> packed[i - 1][0] == depositRecords[target][i - 1].startTime
203       @inv i >= 1 -> packed[i - 1][1] == depositRecords[target][i - 1].principal
204       @inv i >= 1 -> packed[i - 1][3] == depositRecords[target][i - 1].collected
205       @post i == len
206       @post !__should_return
207      */
208     for (uint256 i = 0; i < len; i++) {
209       DepositRecord memory certificate = depositRecords[target][i];
210       packed[i][0] = certificate.startTime;
211       packed[i][1] = certificate.principal;
212       packed[i][2] = uint256(certificate.duration);
213       packed[i][3] = certificate.collected;
214     }
215
216     return (getTime(), packed);
217   }
218
219   function getTotalDeposits(address account) internal view returns (uint256) {
220     uint256 amount = 0;
```

```
221     /*CTK getTotalDeposits
222       @inv i < depositRecords[account].length
223       @inv depositRecords[account][i].principal >= 0
224       @inv amount >= amount__pre + depositRecords[account][i].principal
225       @post i == depositRecords[account].length
226       @post !__should_return
227      */
228     for(uint256 i = 0; i < depositRecords[account].length; i++) {
229       amount = amount.add(depositRecords[account][i].principal);
230     }
231     return amount;
232   }
233 }
```

File HodlWithFakeTime.sol

```
1  pragma solidity >=0.4.25 < 0.5.0;
2
3  import './Hodl.sol';
4
5  // HodlWithFakeTime provides `setTime()` to test time passage
6  // for the `Hodl` contract and should not be deployed on production networks.
7  contract HodlWithFakeTime is Hodl {
8    uint256 fakeTime;
9    /*@CTK getTime
10     @post __return == fakeTime
11    */
12   function getTime() public view returns(uint256) {
13     return fakeTime;
14   }
15
16   /*@CTK setTime
17     @post __post.fakeTime == f
18    */
19   function setTime(uint256 f) public {
20     fakeTime = f;
21   }
22 }
```

File openzeppelin-solidity/contracts/ownership/Ownable.sol

```
1  pragma solidity ^0.5.0;
2
3  /**
4   * @title Ownable
5   * @dev The Ownable contract has an owner address, and provides basic authorization
        control
6   * functions, this simplifies the implementation of "user permissions".
7   */
8  contract Ownable {
9     address private _owner;
10
11    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
          );
12
13    /**
14     * @dev The Ownable constructor sets the original `owner` of the contract to the
          sender
15     * account.
16     */
```

```
17      /*@CTK Ownable
18        @post __post._owner == msg.sender
19       */
20      constructor () internal {
21          _owner = msg.sender;
22          emit OwnershipTransferred(address(0), _owner);
23      }
24
25      /**
26       * @return the address of the owner.
27       */
28    /*@CTK owner
29      @post __return == _owner
30     */
31      function owner() public view returns (address) {
32          return _owner;
33      }
34
35      /**
36       * @dev Throws if called by any account other than the owner.
37       */
38      modifier onlyOwner() {
39          require(isOwner());
40          _;
41      }
42
43      /**
44       * @return true if `msg.sender` is the owner of the contract.
45       */
46    /*@CTK isOwner
47      @post __return == (msg.sender == _owner)
48     */
49      function isOwner() public view returns (bool) {
50          return msg.sender == _owner;
51      }
52
53      /**
54       * @dev Allows the current owner to relinquish control of the contract.
55       * @notice Renouncing to ownership will leave the contract without an owner.
56       * It will not be possible to call the functions with the `onlyOwner`
57       * modifier anymore.
58       */
59    /*@CTK renounceOwnership
60      @tag assume_completion
61      @post _owner == msg.sender
62      @post __post._owner == address(0)
63     */
64      function renounceOwnership() public onlyOwner {
65          emit OwnershipTransferred(_owner, address(0));
66          _owner = address(0);
67      }
68
69      /**
70       * @dev Allows the current owner to transfer control of the contract to a newOwner
           .
71       * @param newOwner The address to transfer ownership to.
72       */
73    /*@CTK transferOwnership
```

```
74      @tag assume_completion
75      @post _owner == msg.sender
76     */
77      function transferOwnership(address newOwner) public onlyOwner {
78          _transferOwnership(newOwner);
79      }
80
81      /**
82       * @dev Transfers control of the contract to a newOwner.
83       * @param newOwner The address to transfer ownership to.
84       */
85    /*@CTK _transferOwnership
86      @tag assume_completion
87      @post newOwner != address(0)
88      @post __post._owner == newOwner
89     */
90      function _transferOwnership(address newOwner) internal {
91          require(newOwner != address(0));
92          emit OwnershipTransferred(_owner, newOwner);
93          _owner = newOwner;
94      }
95  }
```