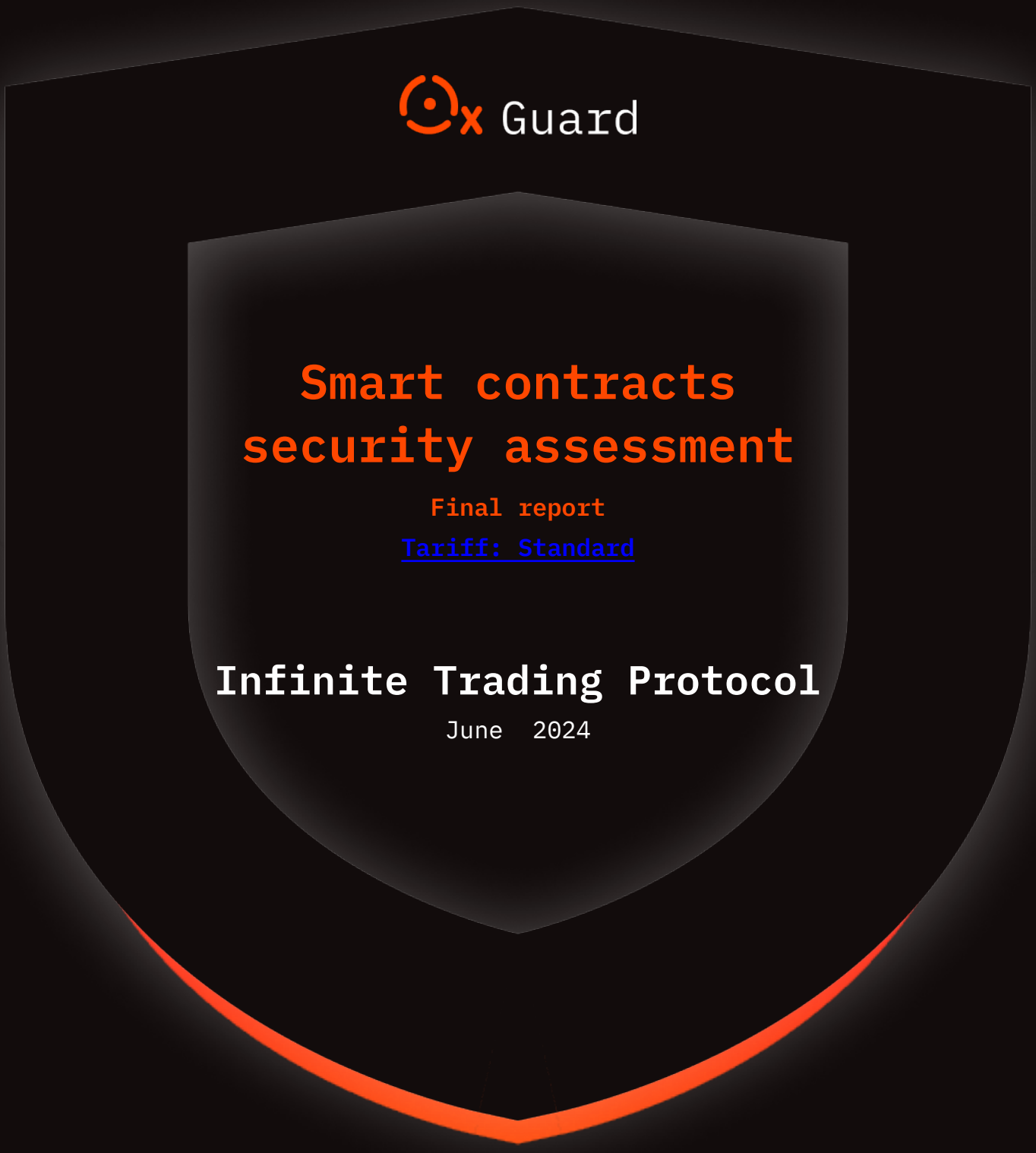

Guard

Smart contracts security assessment

Final report

Tariff: Standard

Infinite Trading Protocol

June 2024

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **Infinite Trading Protocol**.

ItpStakingV1 is a staking contract with multiple locking periods and yield rates available for staking of a single ERC-20 token. The rewards are accrued in the same ERC-20 token, the reward pool is pre-allocated and can be replenished by the contract's owner. Withdrawing prior unlocking timestamp results in loss of rewards and additional penalty applied to the body of the staking.

The code is available at the GitHub repository @InfiniteTradingProtocol/infinite-trading-contracts and was audited after the commit [00ec45d9da6db07dade184c6ed1be7bbfb06a057].

Report Update

The contract's code was updated according to this report and rechecked after the commit [946b48248a107dc52e89f94ef8079ae30b55ad0a].

Name	Infinite Trading Protocol
Audit date	2024-06-04 - 2024-06-06
Language	Solidity
Platform	Optimism Network

Contracts checked

Name	Address
ItpStakingV1	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed

Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

No issues were found

Medium severity issues

No issues were found

Low severity issues

1. Decreased precision of calculations (ItpStakingV1)

Status: Fixed

The calculations of penalty for early withdrawals include unnecessary loss of precision in `unlockTimeDelta = (numerator * 10000) / denominator` and `penalty = (penaltyRateBps * unlockTimeDelta) / 10000`.

Recommendation: Use `penalty = (penaltyRateBps * numerator) / denominator`.

2. Non-indexed events (ItpStakingV1)

Status: Fixed

The user-interacting functions emit individual events, many of which containing user's address. Consider marking it with the `indexed` keyword, if these events are meant to be tracked.

Conclusion

Infinite Trading Protocol ItpStakingV1 contract was audited. 2 low severity issues were found.

2 low severity issues have been fixed in the update.

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 Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Slither output

```
INFO:Detectors:
ItpStakingV1._calculatePenalty(uint256,uint256,uint256,uint256) (contracts/
ItpStakingV1.sol#480-504) performs a multiplication on the result of a division:
        - unlockTimeDelta = (numerator * 10000) / denominator (contracts/
ItpStakingV1.sol#500)
        - penalty = (penaltyRateBps * unlockTimeDelta) / 10000 (contracts/
ItpStakingV1.sol#501)
ItpStakingV1._calculatePenalty(uint256,uint256,uint256,uint256) (contracts/
ItpStakingV1.sol#480-504) performs a multiplication on the result of a division:
        - penalty = (penaltyRateBps * unlockTimeDelta) / 10000 (contracts/
ItpStakingV1.sol#501)
        - (amount * penalty) / 10000 (contracts/ItpStakingV1.sol#503)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply
INFO:Detectors:
Reentrancy in ItpStakingV1.burnPenalty(uint256) (contracts/ItpStakingV1.sol#415-425):
        External calls:
        - ERC20Burnable(address(stakedToken)).burn(amount) (contracts/
ItpStakingV1.sol#419)
        State variables written after the call(s):
        - totalPenalty -= amount (contracts/ItpStakingV1.sol#422)
        ItpStakingV1.totalPenalty (contracts/ItpStakingV1.sol#100) can be used in cross
function reentrancies:
        - ItpStakingV1._validatePenalty(uint256) (contracts/ItpStakingV1.sol#634-638)
        - ItpStakingV1.convertPenaltyIntoRewards(uint256) (contracts/
ItpStakingV1.sol#431-444)
        - ItpStakingV1.getVaultInfo() (contracts/ItpStakingV1.sol#183-203)
        - ItpStakingV1.totalPenalty (contracts/ItpStakingV1.sol#100)
Reentrancy in ItpStakingV1.deposit(uint256,uint256) (contracts/
ItpStakingV1.sol#222-259):
        External calls:
        - stakedToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/
ItpStakingV1.sol#233)
        State variables written after the call(s):
        - rewardsLeft -= rewardAmountToPay (contracts/ItpStakingV1.sol#248)
        ItpStakingV1.rewardsLeft (contracts/ItpStakingV1.sol#97) can be used in cross
function reentrancies:
```

```
        - ItpStakingV1._validateRewards(uint256) (contracts/ItpStakingV1.sol#628-632)
        - ItpStakingV1.convertPenaltyIntoRewards(uint256) (contracts/
ItpStakingV1.sol#431-444)
        - ItpStakingV1.extendLock(uint256,uint256) (contracts/ItpStakingV1.sol#322-358)
        - ItpStakingV1.getVaultInfo() (contracts/ItpStakingV1.sol#183-203)
        - ItpStakingV1.rewardsLeft (contracts/ItpStakingV1.sol#97)
Reentrancy in ItpStakingV1.withdrawPenalty(uint256) (contracts/
ItpStakingV1.sol#382-390):
        External calls:
        - stakedToken.safeTransfer(msg.sender,amount) (contracts/ItpStakingV1.sol#386)
        State variables written after the call(s):
        - totalPenalty -= amount (contracts/ItpStakingV1.sol#387)
        ItpStakingV1.totalPenalty (contracts/ItpStakingV1.sol#100) can be used in cross
function reentrancies:
        - ItpStakingV1._validatePenalty(uint256) (contracts/ItpStakingV1.sol#634-638)
        - ItpStakingV1.convertPenaltyIntoRewards(uint256) (contracts/
ItpStakingV1.sol#431-444)
        - ItpStakingV1.getVaultInfo() (contracts/ItpStakingV1.sol#183-203)
        - ItpStakingV1.totalPenalty (contracts/ItpStakingV1.sol#100)
Reentrancy in ItpStakingV1.withdrawRewards(uint256) (contracts/
ItpStakingV1.sol#371-380):
        External calls:
        - stakedToken.safeTransfer(msg.sender,amount) (contracts/ItpStakingV1.sol#375)
        State variables written after the call(s):
        - rewardsLeft -= amount (contracts/ItpStakingV1.sol#376)
        ItpStakingV1.rewardsLeft (contracts/ItpStakingV1.sol#97) can be used in cross
function reentrancies:
        - ItpStakingV1._validateRewards(uint256) (contracts/ItpStakingV1.sol#628-632)
        - ItpStakingV1.convertPenaltyIntoRewards(uint256) (contracts/
ItpStakingV1.sol#431-444)
        - ItpStakingV1.extendLock(uint256,uint256) (contracts/ItpStakingV1.sol#322-358)
        - ItpStakingV1.getVaultInfo() (contracts/ItpStakingV1.sol#183-203)
        - ItpStakingV1.rewardsLeft (contracts/ItpStakingV1.sol#97)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
INFO:Detectors:
ItpStakingV1.deposit(uint256,uint256) (contracts/ItpStakingV1.sol#222-259) ignores
return value by _stakes[msg.sender].add(tokenId) (contracts/ItpStakingV1.sol#239)
ItpStakingV1.withdraw(uint256[]) (contracts/ItpStakingV1.sol#265-285) ignores return
value by _stakes[msg.sender].remove(tokenId) (contracts/ItpStakingV1.sol#278)
ItpStakingV1.earlyWithdraw(uint256) (contracts/ItpStakingV1.sol#291-315) ignores return
```

```
value by _stakes[msg.sender].remove(tokenId) (contracts/ItpStakingV1.sol#310)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in ItpStakingV1.burnPenalty(uint256) (contracts/ItpStakingV1.sol#415-425):
        External calls:
        - ERC20Burnable(address(stakedToken)).burn(amount) (contracts/
ItpStakingV1.sol#419)
        State variables written after the call(s):
        - totalPenaltyBurned += amount (contracts/ItpStakingV1.sol#421)
Reentrancy in ItpStakingV1.deposit(uint256,uint256) (contracts/
ItpStakingV1.sol#222-259):
        External calls:
        - stakedToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/
ItpStakingV1.sol#233)
        State variables written after the call(s):
        - _stakeInfo[tokenId] =
StakeInfo(tokenId,amount,rewardAmountToPay,block.timestamp,unlockTime) (contracts/
ItpStakingV1.sol#240-246)
        - tokenId = ++ _tokenId (contracts/ItpStakingV1.sol#235)
        - totalStaked += amount + rewardAmountToPay (contracts/ItpStakingV1.sol#249)
Reentrancy in ItpStakingV1.depositRewards(uint256) (contracts/
ItpStakingV1.sol#361-369):
        External calls:
        - stakedToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/
ItpStakingV1.sol#364)
        State variables written after the call(s):
        - rewardsLeft += amount (contracts/ItpStakingV1.sol#365)
        - totalRewards += amount (contracts/ItpStakingV1.sol#366)
Reentrancy in ItpStakingV1.withdrawRewards(uint256) (contracts/
ItpStakingV1.sol#371-380):
        External calls:
        - stakedToken.safeTransfer(msg.sender,amount) (contracts/ItpStakingV1.sol#375)
        State variables written after the call(s):
        - totalRewards -= amount (contracts/ItpStakingV1.sol#377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
ItpStakingV1.extendLock(uint256,uint256) (contracts/ItpStakingV1.sol#322-358) uses
timestamp for comparisons
        Dangerous comparisons:
        - _stakeInfo[tokenId].unlockTime > block.timestamp (contracts/
ItpStakingV1.sol#332)
```

```
ItpStakingV1._calculatePenalty(uint256,uint256,uint256,uint256) (contracts/
ItpStakingV1.sol#480-504) uses timestamp for comparisons
        Dangerous comparisons:
        - currentTime >= unlockTime (contracts/ItpStakingV1.sol#486)
        - lockTime >= unlockTime (contracts/ItpStakingV1.sol#490)
ItpStakingV1._validateEarlyWithdraw(uint256) (contracts/ItpStakingV1.sol#589-593) uses
timestamp for comparisons
        Dangerous comparisons:
        - _stakeInfo[tokenId].unlockTime <= block.timestamp (contracts/
ItpStakingV1.sol#590)
ItpStakingV1._validateLockExtension(uint256,uint256) (contracts/
ItpStakingV1.sol#601-611) uses timestamp for comparisons
        Dangerous comparisons:
        - newLockDuration > maxLockDuration (contracts/ItpStakingV1.sol#608)
ItpStakingV1._validateRewards(uint256) (contracts/ItpStakingV1.sol#628-632) uses
timestamp for comparisons
        Dangerous comparisons:
        - amount > rewardsLeft (contracts/ItpStakingV1.sol#629)
ItpStakingV1._validatePenalty(uint256) (contracts/ItpStakingV1.sol#634-638) uses
timestamp for comparisons
        Dangerous comparisons:
        - amount > totalPenalty (contracts/ItpStakingV1.sol#635)
ItpStakingV1._validateUnlockTime(uint256) (contracts/ItpStakingV1.sol#640-648) uses
timestamp for comparisons
        Dangerous comparisons:
        - _stakeInfo[tokenId].unlockTime > block.timestamp (contracts/
ItpStakingV1.sol#641)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp
INFO:Detectors:
ItpStakingV1._validateUserTokenId(address,uint256) (contracts/ItpStakingV1.sol#622-626)
compares to a boolean constant:
        -_stakes[user].contains(tokenId) == false (contracts/ItpStakingV1.sol#623)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality
INFO:Detectors:
ItpStakingV1.withdraw(uint256[]) (contracts/ItpStakingV1.sol#265-285) has costly
operations inside a loop:
        - delete _stakeInfo[tokenId] (contracts/ItpStakingV1.sol#277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop
```

```
INFO:Detectors:
Pragma version^0.8.20 (contracts/ItpStakingV1.sol#2) necessitates a version too recent
to be trusted. Consider deploying with 0.8.18.
solc-0.8.26 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Parameter ItpStakingV1.setRewardsRatePerLockMultiplierBps(uint256[])._rewardsRatePerLock
MultiplierBps (contracts/ItpStakingV1.sol#393) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Slither:. analyzed (15 contracts with 88 detectors), 25 result(s) found
```

0x Guard