Dusk Smart Contract Audit

Date: September 8, 2020

This document may contain confidential information about IT systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.
The report containing confidential information can be used internally by the customer, or it can disclose publicly after all vulnerabilities are fixed – upon the decision of the customer.

Scope and Code Revision Date

Link	https://github.com/dusk-network/prestaking-contract.git
Date	08.09.2020

Introduction

This report presents the findings of the security assessment of Dusk's smart contract and its code review conducted between September 2 2020 – September 8 2020.

Scope

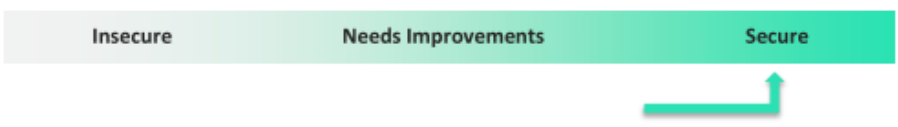
The scope of the project is DUSK smart contract, which can be found here:
https://github.com/dusk-network/prestaking-contract.git

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the widely known vulnerabilities that considered (the full list includes them but is not limited to them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Unchecked external call – Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, Dusk smart contracts are secure. Overall code quality is good.



Our team performed an analysis of code functionality, manual audit and automated checks with Slither and remixed IDE (see Appendix A pic 1-2). All issues found during automated investigation manually reviewed and application vulnerabilities presented in the Audit overview section. A general overview presented in the AS-IS section and all encountered matters can found in the Audit overview section.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss.
High	High-level vulnerabilities are difficult to exploit. However, they also have a significant impact on smart contract execution, e.g. public access to crucial functions.
Medium	Medium-level vulnerabilities are essential to fix; however, they can't lead to tokens loss.
Low	Low-level vulnerabilities are mostly related to outdated or unused code snippets.
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can generally be ignored.

AS-IS overview

Prestaking contract consists of the next smart contracts:

1. SafeMath, IERC20, SafeERC20, Ownable, contracts – standard OpenZeppelin smart contracts for tokens known as etoken2.
2. Prestaking contract – implementation of OpenZeppelin Ownable, SafeERC20, SafeMath smart contract

Contracts from point 1 were compared to original OpenZeppelin templates and we did not find logic differences. They are considered secure.

PrestakingProvisioner contract functional implementation:

1. Prestaking contract inherits Ownable
2. deactivate,returnStake,withdrawStake, withdrawReward,startWithdrawReward,startWithdrawStake,stake, startWithdrawStakeAfterDeactivation

PrestakingProvisioner contract init function was called with following parameters:

- token – DUSK token address

latestVersion was set at the moment of review.

Note: Contract tests in production are out-of-scope of the current security review.

Audit overview

Critical

No critical vulnerabilities found.

High

No high vulnerabilities found.

Medium

No medium severity vulnerabilities found.

Low

No low vulnerabilities found

Lowest / Code style / Best Practice

1. The contract is targeting Solidity compiler version 0.6.12.

   We recommend to use a more recent version of the compiler. The latest version is 0.7.1

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality presented in As-is overview section of the report.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found one best practice that can be applied. No other vulnerabilities were found.

Disclaimer

The smart contracts given for audit have analyzed following the best industry practices at the date of this report, concerning: cybersecurity vulnerabilities and issues in smart contract source code, the details of which disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

The audit doesn't make warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the system, bug free status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is essential to note that you should not rely on this report only. We recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee specific security of the audited smart contracts.

Appendix A. Automated Tools Reports

Pic 1. Slither automated report:

```
INFO:Detectors:
PrestakingProvisioner.distributeRewards(PrestakingProvisioner.Staker,uint256) (PrestakingProvisioner.sol#728-737) performs a multiplication on the result of a division:
        -numDays = comparisonTime.sub(staker.lastUpdated).div(86400) (PrestakingProvisioner.sol#729)
        -reward = staker.dailyReward.mul(numDays) (PrestakingProvisioner.sol#734)
PrestakingProvisioner.distributeRewards(PrestakingProvisioner.Staker,uint256) (PrestakingProvisioner.sol#728-737) performs a multiplication on the result of a division:
        -numDays = comparisonTime.sub(staker.lastUpdated).div(86400) (PrestakingProvisioner.sol#729)
        -staker.lastUpdated = staker.lastUpdated.add(numDays.mul(86400)) (PrestakingProvisioner.sol#736)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
PrestakingProvisioner.distributeRewards(PrestakingProvisioner.Staker,uint256) (PrestakingProvisioner.sol#728-737) uses a dangerous strict equality:
        - numDays == 0 (PrestakingProvisioner.sol#730)
PrestakingProvisioner.onlyActive() (PrestakingProvisioner.sol#562-565) uses a dangerous strict equality:
        - require(bool)(deactivationTime == 0) (PrestakingProvisioner.sol#563)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

```
INFO:Detectors:
PrestakingProvisioner.returnStake(address) (PrestakingProvisioner.sol#601-616) uses timestamp for comparisons
        Dangerous comparisons:
        - deactivationTime != 0 (PrestakingProvisioner.sol#606)
PrestakingProvisioner.stake(uint256) (PrestakingProvisioner.sol#622-641) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(staker.amount == 0,Address already known) (PrestakingProvisioner.sol#625)
PrestakingProvisioner.startWithdrawReward() (PrestakingProvisioner.sol#646-656) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(staker.cooldownTime == 0,A withdrawal call has already been triggered) (PrestakingProvisioner.sol#649)
        - require(bool,string)(staker.endTime == 0,Stake already withdrawn) (PrestakingProvisioner.sol#650)
PrestakingProvisioner.withdrawReward() (PrestakingProvisioner.sol#661-672) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp.sub(cooldownTime) >= 604800 (PrestakingProvisioner.sol#666)
PrestakingProvisioner.startWithdrawStake() (PrestakingProvisioner.sol#677-688) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(staker.startTime.add(2592000) <= blockTimestamp,Stakes can only be withdrawn 30 days after initial lock up) (PrestakingProvisioner.sol#680)
        - require(bool,string)(staker.endTime == 0,Stake withdrawal already in progress) (PrestakingProvisioner.sol#681)
        - require(bool,string)(staker.cooldownTime == 0,A withdrawal call has been triggered - please wait for it to complete before withdrawing your stake) (PrestakingProvisioner.sol#682)
PrestakingProvisioner.startWithdrawStakeAfterDeactivation() (PrestakingProvisioner.sol#698-709) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(staker.startTime.add(2592000) <= blockTimestamp,Stakes can only be withdrawn 30 days after initial lock up) (PrestakingProvisioner.sol#701)
        - require(bool,string)(staker.endTime == 0,Stake withdrawal already in progress) (PrestakingProvisioner.sol#702)
        - require(bool,string)(staker.cooldownTime == 0,A withdrawal call has been triggered - please wait for it to complete before withdrawing your stake) (PrestakingProvisioner.sol#703)
PrestakingProvisioner.withdrawStake() (PrestakingProvisioner.sol#714-722) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp.sub(endTime) >= 604800 (PrestakingProvisioner.sol#719)
PrestakingProvisioner.distributeRewards(PrestakingProvisioner.Staker,uint256) (PrestakingProvisioner.sol#728-737) uses timestamp for comparisons
        Dangerous comparisons:
        - numDays == 0 (PrestakingProvisioner.sol#730)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (PrestakingProvisioner.sol#26-35) uses assembly
        - INLINE ASM None (PrestakingProvisioner.sol#33)
Address._functionCallWithValue(address,bytes,uint256,string) (PrestakingProvisioner.sol#119-140) uses assembly
        - INLINE ASM None (PrestakingProvisioner.sol#132-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Pragma version0.6.12 (PrestakingProvisioner.sol#3) necessitates versions too recent to be trusted. Consider deploying with 0.5.11
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (PrestakingProvisioner.sol#53-59):
        - (success) = recipient.call{value: amount}() (PrestakingProvisioner.sol#57)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (PrestakingProvisioner.sol#119-140):
        - (success,returndata) = target.call{value: weiValue}(data) (PrestakingProvisioner.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter PrestakingProvisioner.returnStake(address)._staker (PrestakingProvisioner.sol#601) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
PrestakingProvisioner.stake(uint256) (PrestakingProvisioner.sol#622-641) uses literals with too many digits:
        - amount > 100000000000000000000000000 || amount < 100000000000000000000000000 (PrestakingProvisioner.sol#627)
PrestakingProvisioner.stake(uint256) (PrestakingProvisioner.sol#622-641) uses literals with too many digits:
        - staker.dailyReward = amount.mul(100020).div(100000).sub(amount) (PrestakingProvisioner.sol#636)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (PrestakingProvisioner.sol#491-493)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (PrestakingProvisioner.sol#510-513)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (PrestakingProvisioner.sol#519-523)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-as-external
```

Pic 2. RemixIDE automated report:


Static Analysis raised 36 warning(s) that requires your attention.
Click here to show the warning(s).
Address
Context
IERC20
Ownable
PrestakingProvisioner
SafeERC20
SafeMath

Potential Violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more

Potential Violation of Checks-Effects-Interaction pattern in SafeERC20.safeIncreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more

Potential Violation of Checks-Effects-Interaction pattern in SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more

browser/PrestakingProvisioner.sol:33:9:CAUTION: The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more

browser/PrestakingProvisioner.sol:132:17:CAUTION: The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more

browser/PrestakingProvisioner.sol:558:42:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

browser/PrestakingProvisioner.sol:591:28:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

browser/PrestakingProvisioner.sol:605:31:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

browser/PrestakingProvisioner.sol:632:31:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

[redacted]

browser/PrestakingProvisioner.sol:648:31:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

browser/PrestakingProvisioner.sol:666:13:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

browser/PrestakingProvisioner.sol:679:31:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

browser/PrestakingProvisioner.sol:700:31:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

browser/PrestakingProvisioner.sol:719:13:use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more

browser/PrestakingProvisioner.sol:57:28:use of "call": the use of low level "call" should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more

Gas requirement of function PrestakingProvisioner.returnStake(address) high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Gas requirement of function PrestakingProvisioner.stake(uint256) high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Gas requirement of function PrestakingProvisioner.startWithdrawReward() high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Gas requirement of function PrestakingProvisioner.startWithdrawStake() high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Gas requirement of function PrestakingProvisioner.startWithdrawStakeAfterDeactivation() high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

[redacted]

Gas requirement of function PrestakingProvisioner.transferOwnership(address) high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Gas requirement of function PrestakingProvisioner.withdrawReward() high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Gas requirement of function PrestakingProvisioner.withdrawStake() high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Address.isContract() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more

SafeERC20._callOptionalReturn() : Potentially should be constant but is not. Note: Modifiers are currently not considered by this static analysis.
more

PrestakingProvisioner.withdrawReward() : Potentially should be constant but is not. Note: Modifiers are currently not considered by this static analysis.
more

PrestakingProvisioner.distributeRewards() : Potentially should be constant but is not. Note: Modifiers are currently not considered by this static analysis.
more

Address.sendValue(): Defines a return type but never explicitly returns a value.

SafeERC20.safeApprove(): Defines a return type but never explicitly returns a value.

SafeERC20._callOptionalReturn(): Defines a return type but never explicitly returns a value.

Ownable.transferOwnership(): Defines a return type but never explicitly returns a value.

PrestakingProvisioner.returnStake(): Defines a return type but never explicitly returns a value.

PrestakingProvisioner.stake(): Defines a return type but never explicitly returns a value.

PrestakingProvisioner.removeUser(): Defines a return type but never explicitly returns a value.

Use assert(x) if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use require(x) if x can be false, due to e.g. invalid input or a failing external component.
more

browser/PrestakingProvisioner.sol:748:9:Using delete on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the length property.
more