TECHFUND Audit Score

Smart contract Audit Report

Astar Farm v0.1.0

Completed on 2022-06-15

Score **POSITIVE**

Risk level		
	Critical	0
	High	0
	Medium	0
	Low	1
	Note	1

Risk level detail

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

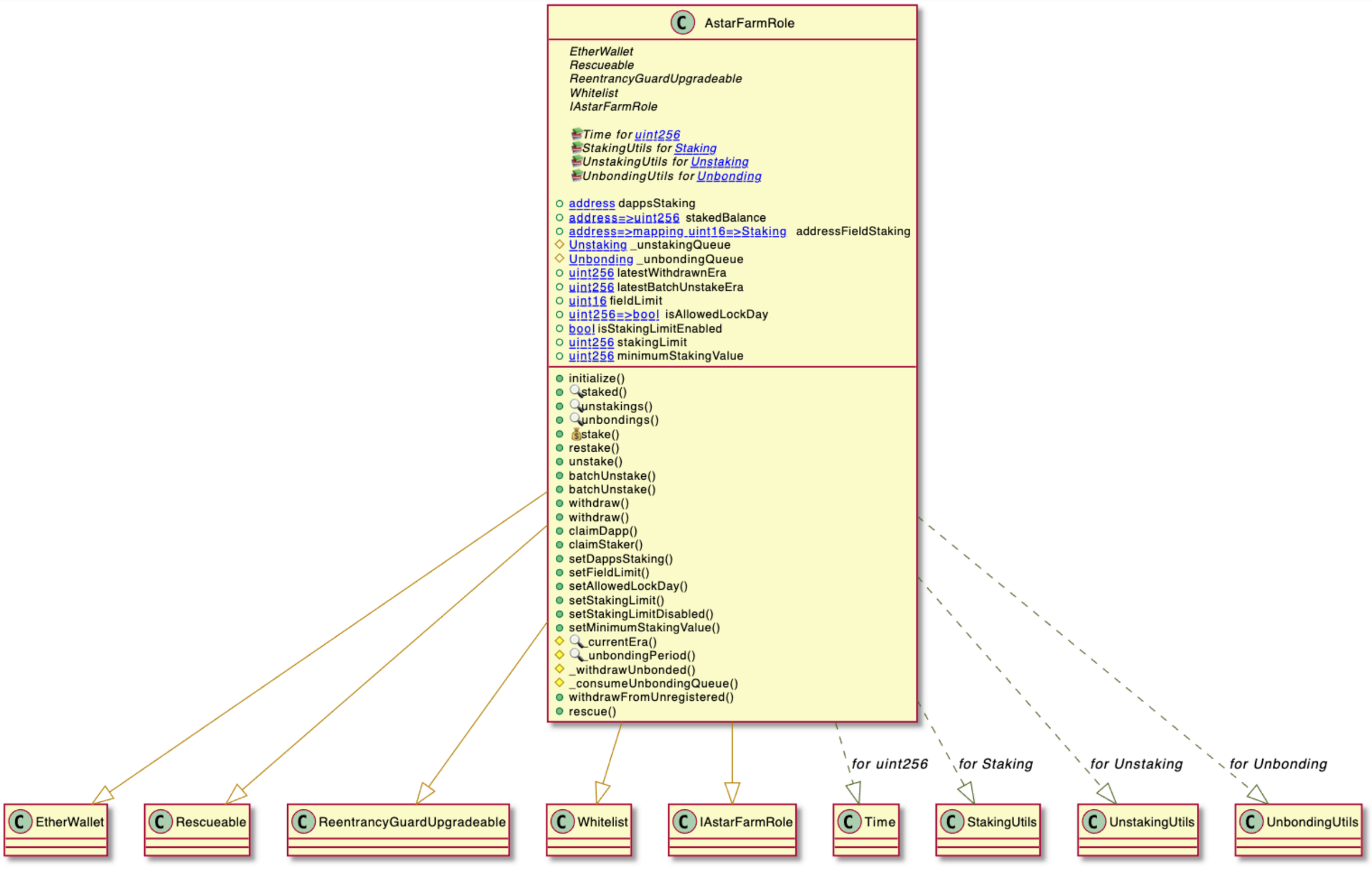
The tester arrives at the likelihood and impact estimates, they can now combine them to get a final severity rating for this risk. Note that if they have good business impact information, they should use that instead of the technical impact information.

https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

Vulnerability Review

	Number of warnings
Compiler Version	0
State Variable Default Visibility	0
Multiple calls in same transaction	1
High Gas cost (In-efficient Struct packing)	1
Integer Overflow / Underflow	0
Parity Multisig Bug	0
Callstack Depth Attack	0
Production Node modules security	0
Development Node modules security	0
Re-Entrancy	0
Double Withdrawal	0

Call Graph


AstarFarmRole
EtherWallet
Rescueable
ReentrancyGuardUpgradeable
Whitelist
IAstarFarmRole
Time for uint256
StakingUtils for Staking
UnstakingUtils for Unstaking
UnbondingUtils for Unbonding
address dappsStaking
address=>uint256 stakedBalance
address=>mapping uint16=>Staking addressFieldStaking
Unstaking _unstakingQueue
Unbonding _unbondingQueue
uint256 latestWithdrawnEra
uint256 latestBatchUnstakeEra
uint16 fieldLimit
uint256=>bool isAllowedLockDay
bool isStakingLimitEnabled
uint256 stakingLimit
uint256 minimumStakingValue
initialize()
staked()
unstakings()
unbondings()
stake()
restake()
unstake()
batchUnstake()
batchUnstake()
withdraw()
withdraw()
claimDapp()
claimStaker()
setDappsStaking()
setFieldLimit()
setAllowedLockDay()
setStakingLimit()
setStakingLimitDisabled()
setMinimumStakingValue()
_currentEra()
_unbondingPeriod()
_withdrawUnbonded()
_consumeUnbondingQueue()
withdrawFromUnregistered()
rescue()
for uint256
for Staking
for Unstaking
for Unbonding
EtherWallet
Rescueable
ReentrancyGuardUpgradeable
Whitelist
IAstarFarmRole
Time
StakingUtils
UnstakingUtils
UnbondingUtils

Multiple calls in same transaction

1

This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently.
It is recommended to follow call best practices:

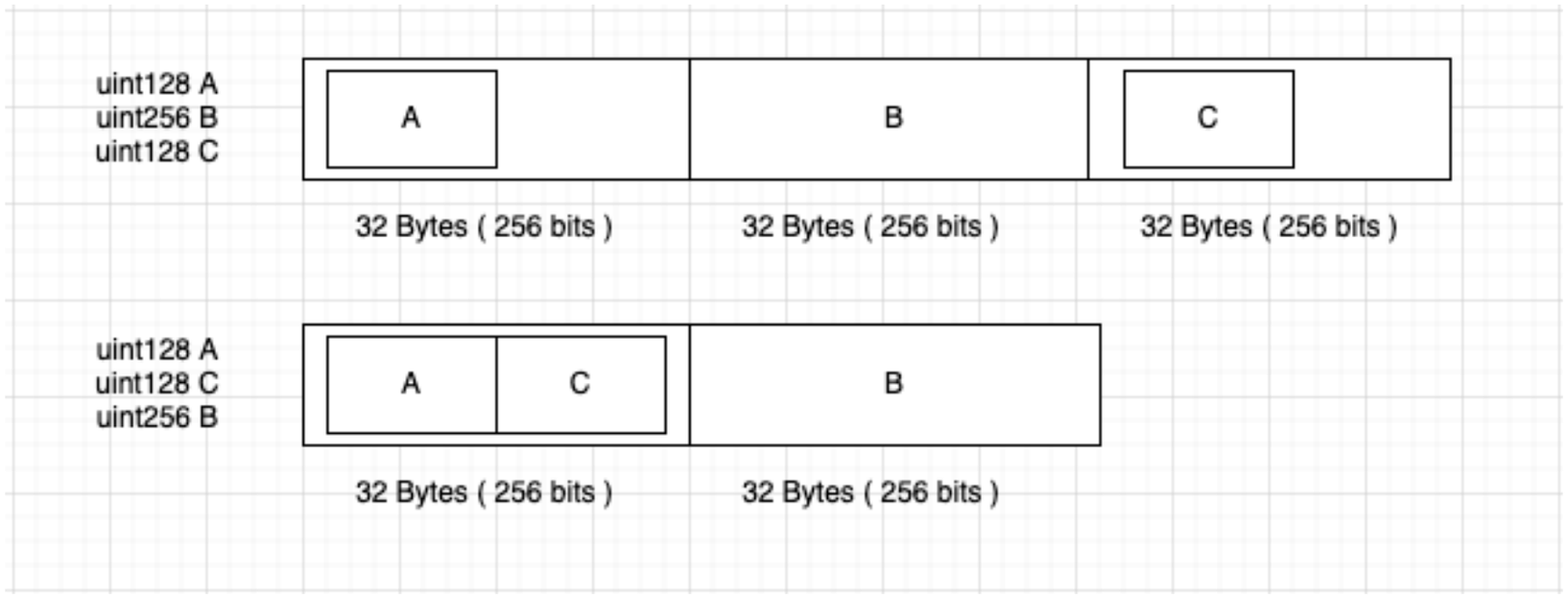
1. Avoid combining multiple calls in a single transaction, especially when calls are executed as part of a loop
2. Always assume that external calls can fail

Since it is just a part of test file , it is currently listed under note.

```
contract Airdrop is EtherWallet {
    function etherBulkTransfer(Entity[] calldata entities) external payable {
        for (uint256 i = 0; i < entities.length; i++) {
            (bool success, ) = payable(entities[i].addr).call{value: entities[i].amount}("");
            require(success, "Failed to send.");
        }
    }
}
```

High Gas cost (In-efficient Struct packing) 1

Every-time we make a variable in our solidity code, the EVM stores it in a storage slot of 32 bytes (256 bits). That means that every time we have a uint (which is read as a uint256) we have packed a storage slot fully.
For example we can see in the following diagram that schema 2 is much more efficient in saving Gas as compared to schema 1.



High Gas cost (In-efficient Struct packing) 1

Similarly in the Staking struct we can see that just restructuring the structure can cause to saving GAS, hence we suggest to use a tightly packed structure for the Staking.

```
4   enum StakingStatus {
5       NotStaked,
6       Staked,
7       Unstaking
8   }
9
10  struct Staking {
11      address staker;
12      uint256 era;
13      uint16 fieldIndex;
14      uint256 lockEra;
15      uint256 value;
16      StakingStatus status;
17  }
18
19  Staking s = Staking({
20      era : 0,
21      lockEra: 0,
22      value: 0,
23      staker: 0x0000000000000000000000000000000000005001,
```

tructDefinition Staking 3 reference(s)

0 listen on all transactions Search with transaction hash or address

```
creation of test pending...

[vm] from: 0x5B3...eddC4 to: test.(constructor) value: 0 wei data: 0x608...e0033 logs: 0 hash: 0x78f...eff5d
status                  true Transaction mined and execution succeed
transaction hash        0x78f01730c60dc58d185205b84d35be635d926aa7b589862ac6d306fcf71eff5d
from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                      test.(constructor)
gas                     144265 gas
transaction cost        125447 gas
```

```
4   enum StakingStatus {
5       NotStaked,
6       Staked,
7       Unstaking
8   }
9
10  struct Staking {
11      StakingStatus status;
12      address staker;
13      uint16 fieldIndex;
14      uint256 era;
15      uint256 lockEra;
16      uint256 value;
17  }
18
19  Staking s = Staking({
20      era : 0,
21      lockEra: 0,
22      value: 0,
23      staker: 0x0000000000000000000000000000000000005001,
```

tructDefinition Staking 3 reference(s)

0 listen on all transactions Search with transaction hash or address

```
[vm] from: 0x5B3...eddC4 to: test.(constructor) value: 0 wei data: 0x608...e0033 logs: 0 hash: 0x7ac...b4c68
status                  true Transaction mined and execution succeed
transaction hash        0x7ac431c1c852682f7cc68903bd982d938b5f0401ad6f28997dd8c224a5fb4c68
from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                      test.(constructor)
gas                     116908 gas
transaction cost        101659 gas
```