

Integer Overflow and Underflow

Description

An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type. For instance, if a number is stored in the uint8 type, it means that the number is stored in a 8 bits unsigned number ranging from 0 to 2^8-1 . In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented with a given number of bits – either larger than the maximum or lower than the minimum representable value.


Remediation

It is recommended to use vetted safe math libraries for arithmetic operations consistently throughout the smart contract system.

Example:

Code:

```
1 pragma solidity ^0.4.21;
2
3 contract TokenSaleChallenge {
4     mapping(address => uint256) public balanceOf;
5     uint256 constant PRICE_PER_TOKEN = 1 ether;
6
7     function TokenSaleChallenge(address _player) public payable {
8         require(msg.value == 1 ether);
9     }
10
11     function isComplete() public view returns (bool) {
12         return address(this).balance < 1 ether;
13     }
14
15     function buy(uint256 numTokens) public payable {
16         require(msg.value == numTokens * PRICE_PER_TOKEN);
17
18         balanceOf[msg.sender] += numTokens;
19     }
20
21     function sell(uint256 numTokens) public {
22         require(balanceOf[msg.sender] >= numTokens);
23
24         balanceOf[msg.sender] -= numTokens;
25         msg.sender.transfer(numTokens * PRICE_PER_TOKEN);
26     }
27 }
```



Explanation:

In the buy and sell functions, the `balanceof[msg.sender]` value will underflow and overflow if the upper and lower limits of `uint256` ($2^{256} + 1$) are exceeded.

By exploiting the vulnerability, an attacker may increase the balance.