DoS with Failed Call

Description

External calls can fail accidentally or deliberately, which can cause a DoS condition in the contract. To minimize the damage caused by such failures, it is better to isolate each external call into its own transaction that can be initiated by the recipient of the call. This is especially relevant for payments, where it is better to let users withdraw funds rather than push funds to them automatically (this also reduces the chance of problems with the gas limit).

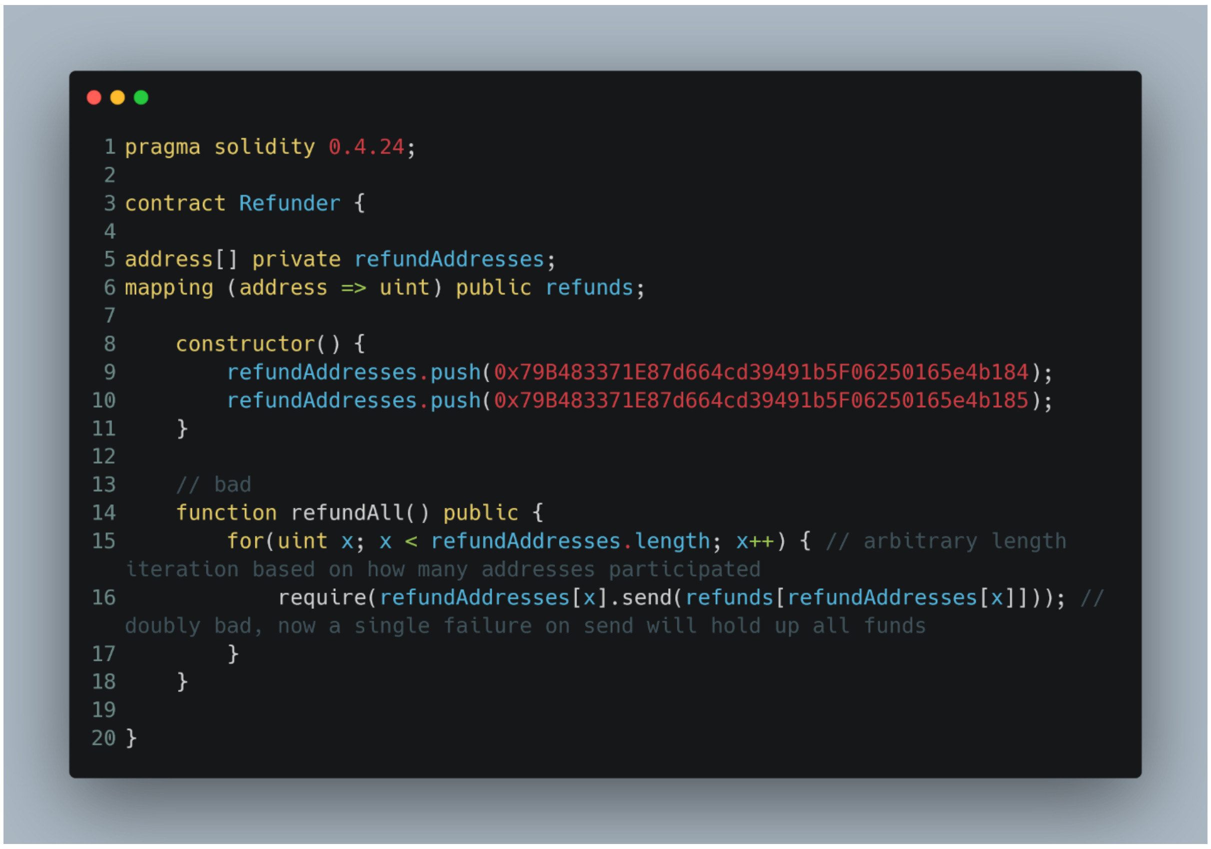
Remediation

It is recommended to follow call best practices:

- Avoid combining multiple calls in a single transaction, especially when calls are executed as part of a loop
- Always assume that external calls can fail
- Implement the contract logic to handle failed calls

Example:

Code:



```
pragma solidity 0.4.24;

contract Refunder {

address[] private refundAddresses;
mapping (address => uint) public refunds;

    constructor() {
        refundAddresses.push(0x79B483371E87d664cd39491b5F06250165e4b184);
        refundAddresses.push(0x79B483371E87d664cd39491b5F06250165e4b185);
    }

    // bad
    function refundAll() public {
        for(uint x; x < refundAddresses.length; x++) { // arbitrary length iteration based on how many addresses participated
            require(refundAddresses[x].send(refunds[refundAddresses[x]])); // doubly bad, now a single failure on send will hold up all funds
        }
    }

}
```

Explanation:

The transaction will be permanently blocked if the length of "refundAddresses" is undefined. The preceding transaction will utilize the gas to complete the transaction, and the subsequent transaction will fail because the gas limit has been reached which leads to block gas limit vulnerability.

Reference: https://blog.finxter.com/denial-of-service-dos-attack-on-smart-contracts/