DoS With Block Gas Limit

Description

When smart contracts are deployed or functions inside them are called, the execution of these actions always requires a certain amount of gas, based of how much computation is needed to complete them. The Ethereum network specifies a block gas limit and the sum of all transactions included in a block cannot exceed the threshold.

Programming patterns that are harmless in centralized applications can lead to Denial-of-Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit. Modifying an array of unknown size, that increases in size over time, can lead to such a Denial-of-Service condition.

Remediation

Caution is advised when you expect to have large arrays that grow over time. Actions that require looping across the entire data structure should be avoided.

If you absolutely must loop over an array of unknown size, then you should plan for it to potentially take multiple blocks, and therefore require multiple transactions.

Example:

Code:

```
pragma solidity ^0.4.25;

contract DosOneFunc {

    address[] listAddresses;

    function ifillArray() public returns (bool){
        if(listAddresses.length<1500) {

            for(uint i=0;i<350;i++) {
                listAddresses.push(msg.sender);
            }
            return true;

        } else {
            listAddresses = new address[](0);
            return false;
        }
    }
}
```

Explanation:

Since listAddresses.length is unknown, it will be run 1,500 times, resulting in more gas consumption during contract execution. This results in the DOS Gas Limit vulnerability.